Vilniaus universitetas

Matematikos ir informatikos fakultetas

STATISTINIS MODELIAVIMAS

Praktinės užduoties Nr. **1—21** ataskaita

Užduotį atliko: **Matas Amšiejus**

Duomenų mokslo antras kursas, 2 grupė

2021 05 12

Turinys

Užduotis 1-21

1. Sugeneruokite pseudoatsitiktinių skaičių sekas tiesiniu kongruentiniu metodu su maksimaliu periodu, kai modulis $m = 656$ ir $m = 810$. Daugiklius a parinkite taip, kad galingumai būtų didžiausi. Prieaugio c parinkimui naudokitės gretimų narių koreliacija (teoriniai testai).
2. Gautas sekas patikrinkite dviem testais. Pirma su maksimumo testu, kai maksimumas išrenkamas iš 6 skaičių. Kitą testą pasirinkite patys.
3. Naudokite sugeneruotą geresniąją pseudoatsitiktinių skaičių seką sumodeliuokite 2 atsitiktinius dydžius, vieną turintį t skirstinį su 3 laisvės laipsniais, o kitą pasirinkite patys.
4. Naudodami sugeneruotą geresniąją pseudoatsitiktinių skaičių seką ir pasirinkdami tankius (tolygiai pasiskirsčiusio intervale [0,4] atsitiktinio dydžio ir kitą savo nuožiūra) suskaičiuokite intergralą:
$$\int_0^4 e^x dx$$
5. Sugeneruokite Markovo grandinę, kuri nebūtų gimimo-mirties procesas ir kurios vienintelis stacionarus skirstinys būtų
$$\pi = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}, 0\right).$$
Naudokite geresniu algoritmu gautus pseudoatsitiktinius skaičius.

1 užduotis

Sugeneruokite pseudoatsitiktinių skaičių sekas tiesiniu kongruentiniu metodu su maksimaliu periodu, kai modulis $m = 656$ ir $m = 810$. Daugiklius a parinkite taip, kad galingumai būtų didžiausi. Prieaugio c parinkimui naudokitės gretimų narių koreliacija (teoriniai testai).

Teorija

Norint generuoti sekas, kurių formulė $X_k \equiv (aX_{k-1} + c) mod\ m$, mums reikės rasti daugiklį a ir prieaugį c. a ir c, kartu su salygos reikalavimais, dar turi tenkinti:

- DBD(c,m)=1, t.y. c ir m negali turėti bendrų pirminių daliklių;
- Jei p|m, tai p|(a-1);
- Jei 4|m, tai 4|(a-1).

Taip pat atkreipsiu dėmesį, kad $a \approx \sqrt{m}$, nes tada gaunasi mažiausia c gretimų narių koreliacijos paklaida per daug nepadidinus pačios koreliacijos. Tam, kad rasti didžiausią galingumą, reikia rasti skaičių s, kuriuo keliant $b=a-1$, gausime m kartotinį (t.y. $b^s \equiv 0 mod\ m$), kur s – galingumas. Ieškant geriausio c, naudosime gretimų narių koreliacijos formulę $C \approx \frac{1}{a}\left(1 - 6\frac{c}{m} + 6\left(\frac{c}{m}\right)^2\right)$ ir

$\frac{c}{m} \approx \frac{1}{2} \pm \frac{1}{6}\sqrt{3}$.

```
Galimi a su galiomis:
a = 165 su galia = 2
a = 329 su galia = 2
a = 493 su galia = 2
Galingiausias a buvo 165 su galia 2
```

Sprendimas

Daugiklio a parinkimas

```
Galimi a su galiomis:
a = 31 su galia = 4
a = 61 su galia = 4
a = 91 su galia = 2
a = 121 su galia = 4
a = 151 su galia = 4
a = 181 su galia = 2
a = 211 su galia = 4
a = 241 su galia = 4
a = 271 su galia = 2
a = 301 su galia = 4
a = 331 su galia = 4
a = 361 su galia = 2
a = 391 su galia = 4
a = 421 su galia = 4
a = 451 su galia = 2
a = 481 su galia = 4
a = 511 su galia = 4
a = 541 su galia = 2
a = 571 su galia = 4
a = 601 su galia = 4
a = 631 su galia = 2
a = 661 su galia = 4
a = 691 su galia = 4
a = 721 su galia = 2
a = 751 su galia = 4
a = 781 su galia = 4
Galingiausias a buvo 31 su galia 4
```

Visų pirma ieškojau tinkamo daugiklio a. Tam pasirašiau kodą, kuris paimtų visus skaičius nuo 1 iki m-1 ir atmestų tuos, kurie netenkina aukščiau nurodytų sąlygų. Pirma tikrinau jei pirminis skaičius dalija m, tai jis turi dalinti ir a-1. Nesunku įsitikinti, kad kai m=656, pirminiai dalinantys skaičiai yra 2 ir 41, o kai m=810 – 2, 3, 5. Tada tikrinau, kad visi likę skaičiai tenkintų sąlygą jei 4 dalina m, tai 4 dalins ir a-1. Mano atveju šis teiginys galiojo tik m=656. Galiausiai tikrinau visų likusių a galingumą. Kėliau turimus a-1 laispniu tol, kol jie pasidalindavo iš m ir fiksavau laipsnį. Pirmame paveikslėjyje m=656, antrame m=810. Taigi nors galingumai ir buvo panašūs tarp skaičių, tačiau aš pasirinkau mažiausius a, nes jie atitiko sąlygą (ar bent jau buvo arčiausiai) $a \approx \sqrt{m}$. Taigi pasirinkti a buvo 165 (m=656) ir 31 (m=810).

Prieaugio c parinkimas

Šiuo atveju pirma ieškosime tiksliausio (mažiausios koreliacijos) prieaugio, tik tada tikrinsime, ar DBD(c,m) = 1.

Kai m = 656, gauname lygtį $\frac{c}{656} \approx \frac{1}{2} \pm \frac{1}{6}\sqrt{3}$. Perkėlus m į kitą pusę gauname 2 spredinius (jau suapvalinta): 517 ir 138. Tačiau prisiminus, kad DBD(c,m)=1, 138 netinkamas, nes dalinasi iš 2.

Taigi c=517. Išsprendus $\frac{c}{810} \approx \frac{1}{2} \pm \frac{1}{6}\sqrt{3}$ antrai sekai, gauname 638 ir 171. Tačiau abu skaičiai netenkina reikalavimų. 638 dalinasi iš 2, 639 – iš 3. Tad imame 637. 171 dalinasi iš 3, 170 – iš 2 ir 5, 172 – 2, taigi lieka arba 169, arba 173. Tada ieškome, kurio c – 637, 169 ar 173 koreliacija mažiausia. Įstatome:

- $C \approx \frac{1}{31}\left(1 - 6\frac{637}{810} + 6\left(\frac{637}{810}\right)^2\right) \approx -0.000251$
- $C \approx \frac{1}{31}\left(1 - 6\frac{169}{810} + 6\left(\frac{169}{810}\right)^2\right) \approx 0.000301$
- $C \approx \frac{1}{31}\left(1 - 6\frac{173}{810} + 6\left(\frac{173}{810}\right)^2\right) \approx -0.000251$

Taigi gauname, kad 637 ir 173 koreliacija yra beveik vienoda, tad tiesiog paimsime 637.

Sekų generavimas

Taigi gauname pseudoatsitiktinę seką $X_k \equiv (165X_{k-1} + 517) mod\ 656$, o X_0 imsime 0. Seka:

0 517 542 75 428 289 314 503 200 61 86 275 628 489 514 47 400 261 286 475 (1)

172 33 58 247 600 461 486 19 372 233 258 447 144 5 30 219 572 433 458 647 (2)

344 205 230 419 116 633 2 191 544 405 430 619 316 177 202 391 88 605 630 163 (3)

516 377 402 591 288 149 174 363 60 577 602 135 488 349 374 563 260 121 146 335 (4)

32 549 574 107 460 321 346 535 232 93 118 307 4 521 546 79 432 293 318 507 (5)

204 65 90 279 632 493 518 51 404 265 290 479 176 37 62 251 604 465 490 23 (6)

376 237 262 451 148 9 34 223 576 437 462 651 348 209 234 423 120 637 6 195 (7)

548 409 434 623 320 181 206 395 92 609 634 167 520 381 406 595 292 153 178 367 (8)

64 581 606 139 492 353 378 567 264 125 150 339 36 553 578 111 464 325 350 539 (9)

236 97 122 311 8 525 550 83 436 297 322 511 208 69 94 283 636 497 522 55 (10)

408 269 294 483 180 41 66 255 608 469 494 27 380 241 266 455 152 13 38 227 (11)

580 441 466 655 352 213 238 427 124 641 10 199 552 413 438 627 324 185 210 399 (12)

96 613 638 171 524 385 410 599 296 157 182 371 68 585 610 143 496 357 382 571 (13)

268 129 154 343 40 557 582 115 468 329 354 543 240 101 126 315 12 529 554 87 (14)

440 301 326 515 212 73 98 287 640 501 526 59 412 273 298 487 184 45 70 259 (15)

612 473 498 31 384 245 270 459 156 17 42 231 584 445 470 3 356 217 242 431 (16)

128 645 14 203 556 417 442 631 328 189 214 403 100 617 642 175 528 389 414 603 (17)

300 161 186 375 72 589 614 147 500 361 386 575 272 133 158 347 44 561 586 119 (18)

472 333 358 547 244 105 130 319 16 533 558 91 444 305 330 519 216 77 102 291 (19)

644 505 530 63 416 277 302 491 188 49 74 263 616 477 502 35 388 249 274 463 (20)

160 21 46 235 588 449 474   7 360 221 246 435 132 649 18 207 560 421 446 635 (21)

332 193 218 407 104 621 646 179 532 393 418 607 304 165 190 379 76 593 618 151 (22)

504 365 390 579 276 137 162 351 48 565 590 123 476 337 362 551 248 109 134 323 (23)

 20 537 562 95 448 309 334 523 220 81 106 295 648 509 534 67 420 281 306 495 (24)

192 53 78 267 620 481 506 39 392 253 278 467 164 25 50 239 592 453 478 11 (25)

364 225 250 439 136 653 22 211 564 425 450 639 336 197 222 411 108 625 650 183 (26)

536 397 422 611 308 169 194 383 80 597 622 155 508 369 394 583 280 141 166 355 (27)

 52 569 594 127 480 341 366 555 252 113 138 327 24 541 566 99 452 313 338 527 (28)

224 85 110 299 652 513 538 71 424 285 310 499 196 57 82 271 624 485 510 43 (29)

396 257 282 471 168 29 54 243 596 457 482 15 368 229 254 443 140   1 26 215 (30)

568 429 454 643 340 201 226 415 112 629 654 187 540 401 426 615 312 173 198 387 (31)

84 601 626 159 512 373 398 587 284 145 170 359 56 573 598 131 484 345 370 559 (32)

256 117 142 331 28 545 570 103 456 317 342 531 228 89 114 303   0

Viso sekoje 33 eilutės, iš kurių 32 po 20 elementų, o paskutinė su 16. Sumoje gaunasi 656. Gale 0, toliau seka kartojasi.

Kita mūsų seka pagal formulę $X_k \equiv (31X_{k-1} + 637) mod\ 810$:

0 173 676 69 692 565 678 131 184 207 110 343 276 629 232 75 68 661 414 47 (1)

 10 483 566 709 282   5 328 621 794 487 690 503 376 489 752 805 18 731 154 87 (2)

440 43 696 689 472 225 668 631 294 377 520 93 626 139 432 605 298 501 314 187 (3)

300 563 616 639 542 775 708 251 664 507 500 283 36 479 442 105 188 331 714 437 (4)

760 243 416 109 312 125 808 111 374 427 450 353 586 519 62 475 318 311 94 657 (5)

290 253 726 809 142 525 248 571 54 227 730 123 746 619 732 185 238 261 164 397 (6)

330 683 286 129 122 715 468 101 64 537 620 763 336 59 382 675 38 541 744 557 (7)

430 543 806 49 72 785 208 141 494 97 750 743 526 279 722 685 348 431 574 147 (8)

680 193 486 659 352 555 368 241 354 617 670 693 596 19 762 305 718 561 554 337 (9)

 90 533 496 159 242 385 768 491   4 297 470 163 366 179 52 165 428 481 504 407 (10)

640 573 116 529 372 365 148 711 344 307 780 53 196 579 302 625 108 281 784 177 (11)

800 673 786 239 292 315 218 451 384 737 340 183 176 769 522 155 118 591 674   7 (12)

390 113 436 729 92 595 798 611 484 597 50 103 126 29 262 195 548 151 804 797 (13)

580 333 776 739 402 485 628 201 734 247 540 713 406 609 422 295 408 671 724 747 (14)

650 73   6 359 772 615 608 391 144 587 550 213 296 439 12 545 58 351 524 217 (15)

420 233 106 219 482 535 558 461 694 627 170 583 426 419 202 765 398 361 24 107 (16)

250 633 356 679 162 335 28 231 44 727 30 293 346 369 272 505 438 791 394 237 (17)

230 13 576 209 172 645 728 61 444 167 490 783 146 649 42 665 538 651 104 157 (18)

180 83 316 249 602 205 48 41 634 387 20 793 456 539 682 255 788 301 594 767 (19)

460 663 476 349 462 725 778 801 704 127 60 413 16 669 662 445 198 641 604 267 (20)

350 493 66 599 112 405 578 271 474 287 160 273 536 589 612 515 748 681 224 637 (21)

480 473 256   9 452 415 78 161 304 687 410 733 216 389 82 285 98 781 84 347 (22)

400 423 326 559 492 35 448 291 284 67 630 263 226 699 782 115 498 221 544 27 (23)

200 703 96 719 592 705 158 211 234 137 370 303 656 259 102 95 688 441 74 37 (24)

510 593 736 309 32 355 648 11 514 717 530 403 516 779 22 45 758 181 114 467 (25)

 70 723 716 499 252 695 658 321 404 547 120 653 166 459 632 325 528 341 214 327 (26)

590 643 666 569 802 735 278 691 534 527 310 63 506 469 132 215 358 741 464 787 (27)

270 443 136 339 152 25 138 401 454 477 380 613 546 89 502 345 338 121 684 317 (28)

280 753 26 169 552 275 598 81 254 757 150 773 646 759 212 265 288 191 424 357 (29)

710 313 156 149 742 495 128 91 564 647 790 363 86 409 702 65 568 771 584 457 (30)

570 23 76 99   2 235 168 521 124 777 770 553 306 749 712 375 458 601 174 707 (31)

220 513 686 379 582 395 268 381 644 697 720 623 46 789 332 745 588 581 364 117 (32)

560 523 186 269 412 795 518 31 324 497 190 393 206 79 192 455 508 531 434 667 (33)

600 143 556 399 392 175 738 371 334 807 80 223 606 329 652 135 308   1 204 17 (34)

700   3 266 319 342 245 478 411 764 367 210 203 796 549 182 145 618 701 34 417 (35)

140 463 756 119 622 15 638 511 624 77 130 153 56 289 222 575 178 21 14 607 (36)

360 803 766 429 512 655 228 761 274 567 740 433 636 449 322 435 698 751 774 677 (37)

100 33 386 799 642 635 418 171 614 577 240 323 466 39 572 85 378 551 244 447 (38)

260 133 246 509 562 585 488 721 654 197 610 453 446 229 792 425 388 51 134 277 (39)

660 383 706 189 362 55 258 71 754 57 320 373 396 299 532 465   8 421 264 257 (40)

40 603 236 199 672 755 88 471 194 517   0

Sekoje 40 eilučių po 20 elementų ir paskutinė su 10. Sumoje 810 elementų. Valio, sekos sugeneruotos. (Toliau pirmoji seka prasidės nuo 517, o antroji, nuo 173 (pasislinks per vieną), nes klišai išsaugojau seką kompiuterio atmintyje nenorėdamas, kad dubliuotūsi 0 (dabar 0 gale)).

2 užduotis

Gautas sekas patikrinkite dviem testais. Pirma su maksimumo testu, kai maksimumas išrenkamas iš 6 skaičių. Kitą testą pasirinkite patys.

Maksimumo testas

Visų pirma reikia atkreipti dėmesį, kad dauguma testų naudoja ne sveikųjų skaičių, o tolygiai pasiskirsčiusių skaičių intervale [0;1] seką. Taigi pirma konvertuojame abi sekas, dalindami kiekvieną jų elementą iš tos sekos *m.*

Pirma seka (m=656)

Skirstome į grupeles po 6 elementus ir ieškome jų maksimumo ($V_j = \max\left(U_{jk}, U_{jk+1}, \dots, U_{jk+k-1}\right), 0 \le j < n$). Taigi viso naujų elementų bus 656/6=109,33 grupės, t.y. bus 109 grupės iš 6 elementų maksimumo ir viena iš 2 elementų maksimumo. Pasitelkiame kodą. Gauname seką:

0.826219; 0.957317; 0.783537; 0.914634; 0.740854; 0.871951; 0.986281; 0.964939; 0.943598; 0.960366; 0.900915; 0.917683; 0.858232; 0.875; 0.815549; 0.832317; 0.772866; 0.963415; 0.730183; 0.920732; 0.6875; 0.992378; 0.971037; 0.949695; 0.966463; 0.907012; 0.923781; 0.864329; 0.881098; 0.821646; 0.838415; 0.778963; 0.969512; 0.736281; 0.926829; 0.693598; 0.884146; 0.998476; 0.977134; 0.955793; 0.972561; 0.91311; 0.929878; 0.870427; 0.887195; 0.827744; 0.844512; 0.97561; 0.801829; 0.932927; 0.759146; 0.890244; 0.716463; 0.983232; 0.96189; 0.978659; 0.919207; 0.935976; 0.876524; 0.893293; 0.833841; 0.85061; 0.791159; 0.981707; 0.748476; 0.939024; 0.705793; 0.896341; 0.989329; 0.967988; 0.984756; 0.925305; 0.942073; 0.882622; 0.89939; 0.839939; 0.856707; 0.797256; 0.987805; 0.754573; 0.945122; 0.71189; 0.902439; 0.669207; 0.995427; 0.974085; 0.990854; 0.931402; 0.948171; 0.888719; 0.905488; 0.846037; 0.862805; 0.993902; 0.820122; 0.951219; 0.777439; 0.908537; 0.734756; 0.865854; 0.980183; 0.996951; 0.9375; 0.954268; 0.894817; 0.911585; 0.852134; 0.868902; 0.809451; 0.46189;

Dabar galime taikyti KS (Kolmogorovo-Smirnovo) arba ω^2 kriterijų. Renkuosi KS. Taigi mums reikės tikrinti, ar $\sqrt{n}D_n \le x_\alpha$, kur n – elementų skaičius (mūsų atveju 110), x_α yra KS kriterijaus α kritinė reikšmė (paprastai α=0.05), o D_n – statistika. Iškart galime gauti, kad $x_{0.05} = 0.053$ (iš internetinės lentelės). Tereikia rasti D_n. Tam, kad statistiką būtų lengviau rasti, surūšiuojame mūsų tolygiai pasiskirsčiusių dydžių vektorių didėjimo tvarka, o tada ieškome $D_n^+ = max_{1\le i\le n}\left(\frac{i}{n} - F(X_i^*)\right)$ ir $D_n^- = max_{1\le i\le n}\left(F(X_i^*) - \frac{i-1}{n}\right)$ (kur $F(X_i^*)$ maksimumų surūšiuota seka) ir iš jų gauname $D_n = \max(D_n^+, D_n^-)$. Taigi surūšiuojame maksimumų vektorių, tada, iš programos, gauname, kad $D_n^+ = 0.151852$, o $D_n^- = 0.67852$. Taigi $D_n = 0.67852$, o $\sqrt{n}D_n = \sqrt{110} * 0.67852 = 7.116$. Deja gauname, kad pirmosios sekos hipotezė, kad ji tolygiai pasiskirsčiusi intervale [0;1], statistiškai skiriasi su patikimumo lygmeniu $\beta = 0.95$, nes $7.116 \ge 0.053$.

Antra seka (m=810)

Kaip ir pirmoje sekoje pirma skirstome į grupes po 6 ir išrenkame jų maksimumą (viso bus 810/6=135 elementų / grupių). Gaunama seka:

0.914815; 0.864198; 0.916049; 0.987654; 0.993827; 0.977778; 0.946914; 0.722222; 0.885185; 0.769136; 0.330864; 0.955556; 0.87037; 0.865432; 0.862963; 0.923457; 0.883951; 0.933333; 0.848148; 0.797531; 0.849383; 0.920988;

0.953086; 0.939506; 0.880247; 0.655556; 0.818519; 0.702469; 0.976543; 0.888889; 0.803704; 0.995062; 0.935802; 0.85679; 0.817284; 0.934568; 0.781482; 0.730864; 0.782716; 0.991358; 0.88642; 0.938272; 0.964198; 0.588889; 0.751852; 0.635802; 0.992593; 0.822222; 0.985185; 0.928395; 0.869136; 0.790123; 0.97037; 0.867901; 0.965432; 0.664198; 0.983951; 0.924691; 0.948148; 0.871605; 0.940741; 0.975309; 0.685185; 0.569136; 0.925926; 0.980247; 0.918519; 0.861728; 0.802469; 0.723457; 0.988889; 0.801235; 0.898765; 0.597531; 0.95679; 0.858025; 0.966667; 0.804938; 0.874074; 0.954321; 0.960494; 0.98642; 0.97284; 0.91358; 0.851852; 0.795062; 0.735802; 0.65679; 0.922222; 0.734568; 0.969136; 0.530864; 0.890123; 0.967901; 0.9; 0.738272; 0.807407; 0.887654; 0.893827; 0.919753; 0.997531; 0.846914; 0.785185; 0.728395; 0.669136; 0.94321; 0.855556; 0.961728; 0.902469; 0.464198; 0.823457; 0.901235; 0.998765; 0.996296; 0.740741; 0.820988; 0.958025; 0.981481; 0.930864; 0.982716; 0.718518; 0.661728; 0.602469; 0.959259; 0.788889; 0.951852; 0.835802; 0.397531; 0.75679; 0.937037; 0.932099; 0.92963; 0.990123; 0.950617; 0.891358;

Vėl naudosiu KS kriterijų su tais pačiais parametrais. Skirsis n – dabar ji lygi 135, taip pat $x_{0.05} = 0.0477$. Surūšiuojame maksimumus ir ieškome didžiausio D_n^+ ir D_n^-. Iš kodo gauname, kad $D_n^+ = 0.0469136$, o $D_n^- = 0.607407$. Taigi $D_n = 0.607407$. $\sqrt{n}D_n = \sqrt{135} * 0.607407 = 7.0574$. Vėlgi gauname, kad $7.0574 \geq 0.0477$, tad hipotezė, kad seka tolygiai pasiskirsčiusi intervale nėra tenkinama.

Abi sekos

Nors abi sekos netenkina maksimumo testo, antroji seka yra nežymiai geresnė už pirmąją.

Kėlinių testas
Teorija

Kėlinių testui reikia vėl skaidyti seką po į n grupes po k elementų ir kiekvienoje iš jų $(U_{jk}, U_{jk+1}, \dots, U_{jk+k-1}), 0 \leq j < n$. Tada gali būti k! variantų. Paskaičiuojamem kiek kartų kiekvienas dėstinys kartojasi. Tada pritaikomas χ^2 kriterijus su r=k! ir tikimybėmis $p_i = 1/k!$.

χ^2 kriterijaus testas remiasi Pirsono teorema:

$$\chi_n^2 = \sum_{i=1}^{r} \frac{(v_i - np_i)^2}{np_i}$$

Seka tenkins testą, jei bus teisinga nelygybė $k_n^2 < \chi^2(r-1, \alpha)$. Kitu atveju hipotezę atmesime.

Sprendimas

Pirma seka (m=656)

Renkamės k=3. Tada $n = 656/3 \equiv 218$ (atmečiau paskutinę grupę, nes ji buvo nepilna). Visi galimi grupių išsidėstymo variantai yra 6:

$U_{3j} < U_{3j+1} < U_{3j+2}$, arba $U_{3j} < U_{3j+2} < U_{3j+1}$, arba $U_{3j+1} < U_{3j} < U_{3j+2}$, arba

$U_{3j+1} < U_{3j+2} < U_{3j}$, arba $U_{3j+2} < U_{3j} < U_{3j+1}$, arba $U_{3j+2} < U_{3j+1} < U_{3j}$.

Jų dažnius iš eilės pavadinsime $v_1, v_2, \dots, v_6$. Taigi pasitelkdami kodą (ir konvertuotus į tolygius duomenis iš maksimumo testo) randame, kad:

$v_1 = 56, v_2 = 21, v_3 = 14, v_4 = 64, v_5 = 44, v_6 = 19,$ n=218. Duomenis statome į Chi kvadrato kriterijų su r=3!=6 ir p=1/k=1/3. Skaičiuojame statistiką:

$$k_{218}^2 = \sum_{i=1}^{r=6} \frac{\left(v_i - 218 * \frac{1}{6}\right)^2}{218 * \frac{1}{6}} = \sum_{i=1}^{6} \frac{(v_i - 36.333)^2}{36.333} = 10.645 + 6.471 + 13.728 + 21.067 + 1.618 +$$

$+8.269 = 61.798$. Dabar lyginame su $\chi^2_{\alpha=0.05}(5) \approx 11.071$ ir matome, kad seka netenkina hipotezės, nes 61.798 nėra mažiau nei 11.071. Taigi seka, imant po 3, nėra tolygiai pasiskirsčiusi, galime matyti, kad dominuoja dėstiniai v_1, v_5.

Antra seka (m=810)

Kaip ir pirmos sekos atveju k, α, r, p bus tokie patys. Randame dėstinių dažnius:

$v_1 = 44, v_2 = 64, v_3 = 37, v_4 = 46, v_5 = 30, v_6 = 49, n = 270$. Statome į formulę:

$$k_{270}^2 = \sum_{i=1}^{r=6} \frac{\left(v_i - 270 * \frac{1}{6}\right)^2}{270 * \frac{1}{6}} = \sum_{i=1}^{6} \frac{(v_i - 45)^2}{45} = 0.022 + 8.022 + 1.422 + 0.022 + 5 + 0.356 = 14.844$$

Ir vėl seka neatitinka hipotezės, nes 11.071 nėra mažiau už 14.844. Tačiau ši seka yra žymiai labiau tolydi imant po 3 nei pirmoji seka. Taip pat ji buvo geresnė atliekant ir maksimumo testą, todėl tolimesniame darbe naudosiu antrą seką (m=810).

3 užduotis

Naudokite sugeneruotą geresniąją pseudoatsitiktinių skaičių seką sumodeliuokite 2 atsitiktinius dydžius, vieną turintį t skirstinį su 3 laisvės laipsniais, o kitą pasirinkite patys.

T (stjudento) skirstinys

Teorija

T skirstinio generavimas

Tegul $Y_1 \in N(0,1)$, o Y_2 nepriklauso nuo Y_1 ir pasiskirstęs pagal χ^2 skirstinį su v laisvės laipsnių. Tada a.d. $X = Y_1\sqrt{\frac{v}{Y_2}}$ turi t pasiskirstymą su v laisvės laispnių.

Standartinio normaliojo skirstinio generavimas

Tegul U_1 ir U_2 – nepriklausomi ir tolygiai pasiskirstę intervale [0, 1) atsitiktiniai dydžiai. Tuomet atsitiktiniai dydžiai $V_1 = 2U_1 - 1$ ir $V_2 = 2U_2 - 1$ yra nepriklausomi ir tolygiai pasiskirstę intervale [−1, 1). Pažymėkime $S = V_1^2 + V_2^2$. Jeigu S ≥ 1, imame naujus U_1 ir U_2 bei generuojame S iš naujo. Imame tik 0 ≤ S < 1. Tada

$$X_i = V_i * \sqrt{\frac{-2\ln(S)}{S}} \quad (formulė\ 1)$$

yra nepriklausomi ir normaliai pasiskirstę atsitiktiniai dydžiai (X_1, X_2, … $\in N(0,1)$).

Chi kvadrato skirstinio generavimas

Modeliuodami χ^2 skirstinį turėsime naudoti formulę $X = 2(Y_1 + \cdots + Y_k) + Z^2$, nes duotasis laisvės laipsnis yra nelyginis (3), tai k=1. Y_i yra nepriklausomai eksponentiškai pasiskirstę (su vidurkiu 1) a.d., o Z $\in N(0,1)$ ir nepriklauso nuo Y_i.

Eksponentinio dydžio generavimas

Jei ekspnentinio dydžio vidurkis EX=1, tai X = − lnU. Reikšmės, kai U = 0 reikia išmesti arba pakeisti į U = 1.

Sprendimas

Tam, kad gautume t skirstinį, mums reikia atsekti kitus skirstinius, kurių reikauja šio a.d. modeliavimas. Taigi pagal apibrėžimą mums reikės standartiškai normaliai pasiskirsčiusio a.d. ir pagal χ^2 pasiskirsčiusio a.d. Pradėkime nuo standartinio normaliojo a.d. modeliavimo. Pirma teks padalinti turimą tolygią seką į 3 dalis tam, kad a.d., generuojami iš gautos sekos, būtų nepriklausomi (normalieji, chi kvadrat (chi kvadratui reikės 2 sekų, nes jam reikės eksponentinio bei normaliojo a.d. (kuris turi nepriklausyti nuo kito normaliojo a.d.))). Tai turėsime 3 sekas po 810/3=270. Pirmą dalį skiriame st. norm. a.d. Pirmomas 540 elementų skiriame norm. a.d.:

0.78642; 0.165432; 0.914815; 0.145679; 0.302469; 0.162963; 0.838272; 0.772839; 0.744444; 0.864198; 0.576543; 0.659259; 0.223457; 0.71358; 0.907407; 0.916049; 0.183951; 0.488889; 0.941975; 0.987654; 0.403704; 0.301235; 0.124691; 0.651852; 0.993827; 0.595062; 0.233333; 0.0197531; 0.398765; 0.148148; 0.379012; 0.535802; 0.396296; 0.0716049; 0.00617284; 0.977778; 0.0975309; 0.809877; 0.892593; 0.45679; 0.946914; 0.140741; 0.149383; 0.417284; 0.722222; 0.175309; 0.220988; 0.637037; 0.534568; 0.358025; 0.885185; 0.22716; 0.828395; 0.466667; 0.253086; 0.632099; 0.381481; 0.612346; 0.769136; 0.62963; 0.304938; 0.239506; 0.211111; 0.330864; 0.0432099; 0.125926; 0.690123; 0.180247; 0.374074; 0.382716; 0.650617; 0.955556; 0.408642; 0.454321; 0.87037; 0.767901; 0.591358; 0.118519; 0.460494; 0.0617284; 0.7; 0.48642; 0.865432; 0.614815; 0.845679; 0.00246914; 0.862963; 0.538272; 0.47284; 0.444444; 0.564198; 0.276543; 0.359259; 0.923457; 0.41358; 0.607407; 0.616049; 0.883951; 0.188889; 0.641975; 0.687654; 0.103704; 0.00123457; 0.824691; 0.351852; 0.693827; 0.295062; 0.933333; 0.719753; 0.0987654; 0.848148; 0.0790123; 0.235802; 0.0962963; 0.771605; 0.706173; 0.677778; 0.797531; 0.509877; 0.592593; 0.15679; 0.646914; 0.840741; 0.849383; 0.117284; 0.422222; 0.875309; 0.920988; 0.337037; 0.234568; 0.0580247; 0.585185; 0.927161; 0.528395; 0.166667; 0.953086; 0.332099; 0.0814815; 0.312346; 0.469136; 0.32963; 0.00493827; 0.939506; 0.911111; 0.0308642; 0.74321; 0.825926; 0.390123; 0.880247; 0.0740741; 0.082716; 0.350617; 0.655556; 0.108642; 0.154321; 0.57037; 0.467901; 0.291358; 0.818519; 0.160494; 0.761728; 0.4; 0.18642; 0.565432; 0.314815; 0.545679; 0.702469; 0.562963; 0.238272; 0.17284; 0.144444; 0.264198; 0.976543; 0.0592593; 0.623457; 0.11358; 0.307407; 0.316049; 0.583951; 0.888889; 0.341975; 0.387654; 0.803704; 0.701235; 0.524691; 0.0518519; 0.393827; 0.995062; 0.633333; 0.419753; 0.798765; 0.548148; 0.779012; 0.935802; 0.796296; 0.471605; 0.406173; 0.377778; 0.497531; 0.209877; 0.292593; 0.85679; 0.346914; 0.540741; 0.549383; 0.817284; 0.122222; 0.575309; 0.620988; 0.037037; 0.934568; 0.758025; 0.285185; 0.62716; 0.228395; 0.866667; 0.653086; 0.0320988; 0.781482; 0.0123457; 0.169136; 0.0296296; 0.704938; 0.639506; 0.611111; 0.730864; 0.44321; 0.525926; 0.0901235; 0.580247; 0.774074; 0.782716; 0.0506173; 0.355556; 0.808642; 0.854321; 0.27037; 0.167901; 0.991358; 0.518519; 0.860494; 0.461728; 0.1; 0.88642; 0.265432; 0.0148148; 0.245679; 0.402469; 0.262963; 0.938272; 0.87284; 0.844444; 0.964198; 0.676543; 0.759259; 0.323457; 0.81358; 0.00740741; 0.0160494; 0.283951; 0.588889; 0.0419753;

0.0876543; 0.503704; 0.401235; 0.224691; 0.751852; 0.0938272; 0.695062; 0.333333; 0.119753; 0.498765; 0.248148; 0.479012; 0.635802; 0.496296; 0.171605; 0.106173; 0.0777778; 0.197531; 0.909877; 0.992593; 0.55679; 0.0469136; 0.240741; 0.249383; 0.517284; 0.822222; 0.275309; 0.320988; 0.737037; 0.634568; 0.458025; 0.985185; 0.327161; 0.928395; 0.566667; 0.353086; 0.732099; 0.481481; 0.712346; 0.869136; 0.72963; 0.404938; 0.339506; 0.311111; 0.430864; 0.14321; 0.225926; 0.790123; 0.280247; 0.474074; 0.482716; 0.750617; 0.0555556; 0.508642; 0.554321; 0.97037; 0.867901; 0.691358; 0.218519; 0.560494; 0.161728; 0.8; 0.58642; 0.965432; 0.714815; 0.945679; 0.102469; 0.962963; 0.638272; 0.572839; 0.544444; 0.664198; 0.376543; 0.459259; 0.0234568; 0.51358; 0.707407; 0.716049; 0.983951; 0.288889; 0.741975; 0.787654; 0.203704; 0.101235; 0.924691; 0.451852; 0.793827; 0.395062; 0.0333333; 0.819753; 0.198765; 0.948148; 0.179012; 0.335802; 0.196296; 0.871605; 0.806173; 0.777778; 0.897531; 0.609877; 0.692593; 0.25679; 0.746914; 0.940741; 0.949383; 0.217284; 0.522222; 0.975309; 0.0209877; 0.437037; 0.334568; 0.158025; 0.685185; 0.0271605; 0.628395; 0.266667; 0.0530864; 0.432099; 0.181481; 0.412346; 0.569136; 0.42963; 0.104938; 0.0395062; 0.0111111; 0.130864; 0.84321; 0.925926; 0.490123; 0.980247; 0.174074; 0.182716; 0.450617; 0.755556; 0.208642; 0.254321; 0.67037; 0.567901; 0.391358; 0.918519; 0.260494; 0.861728; 0.5; 0.28642; 0.665432; 0.414815; 0.645679; 0.802469; 0.662963; 0.338272; 0.27284; 0.244444; 0.364198; 0.0765432; 0.159259; 0.723457; 0.21358; 0.407407; 0.416049; 0.683951; 0.988889; 0.441975; 0.487654; 0.903704; 0.801235; 0.624691; 0.151852; 0.493827; 0.0950617; 0.733333; 0.519753; 0.898765; 0.648148; 0.879012; 0.0358025; 0.896296; 0.571605; 0.506173; 0.477778; 0.597531; 0.309877; 0.392593; 0.95679; 0.446914; 0.640741; 0.649383; 0.917284; 0.222222; 0.675309; 0.720988; 0.137037; 0.0345679; 0.858025; 0.385185; 0.727161; 0.328395; 0.966667; 0.753086; 0.132099; 0.881481; 0.112346; 0.269136; 0.12963; 0.804938; 0.739506; 0.711111; 0.830864; 0.54321; 0.625926; 0.190123; 0.680247; 0.874074; 0.882716; 0.150617; 0.455556; 0.908642; 0.954321; 0.37037; 0.267901; 0.091358; 0.618519; 0.960494; 0.561728; 0.2; 0.98642; 0.365432; 0.114815; 0.345679; 0.502469; 0.362963; 0.0382716; 0.97284; 0.944444; 0.0641975; 0.776543; 0.859259; 0.423457; 0.91358; 0.107407; 0.116049; 0.383951; 0.688889; 0.141975; 0.187654; 0.603704; 0.501235; 0.324691; 0.851852; 0.193827; 0.795062; 0.433333; 0.219753; 0.598765; 0.348148; 0.579012; 0.735802; 0.596296; 0.271605; 0.206173; 0.177778; 0.297531; 0.00987654; 0.0925926; 0.65679; 0.146914; 0.340741; 0.349383; 0.617284; 0.922222; 0.375309; 0.420988; 0.837037; 0.734568; 0.558025; 0.0851852; 0.427161; 0.0283951; 0.666667;

Ieškodami N(0,1) (Y_1), raskime S. Tam reikia $V_1 = 2U_1 - 1$ ir $V_2 = 2U_2 - 1$, tai gauname:

$V_1 = 2 * 0.78642 - 1 = 0.57284; V_2 = 2 * 0.165432 - 1 = -0.669136; S = V_1^2 + V_2^2 = 0.7759$

S tenkina sąlygą, tad tęsiame darbą. Modeliuojame a.d. pagal standartinį normalųjį skirstinį:

0.463268; -0.541145; 0.67094; -0.573095; -0.319495; -0.545139; 0.547136; 0.441303; 0.395375; 0.589069; 0.123804; 0.257593; -0.447293; 0.345454; 0.658959; 0.672937; -0.511192; -0.0179716; 0.714871; 0.788754; -0.155754; -0.321492; -0.607041; 0.245612; 0.798738; 0.153757; -0.431319; -0.776773; -0.163741; -0.569101; -0.195691; 0.0579085; -0.167735; -0.692905; -0.798738; 0.772779; -0.650972; 0.501208; 0.634997; -0.0698896; 0.722858; -0.581082; -0.567104; -0.133789; 0.359432; -0.52517; -0.451287; 0.22165; 0.0559117; -0.229637; 0.623016; -0.441303; 0.531161; -0.0539148; -0.399369; 0.213662; -0.191697; 0.181713; 0.435312; 0.209669; -0.315502; -0.421334; -0.467262; -0.273568; -0.738833; -0.605044; 0.307514; -0.517183; -0.203678; -0.1897; 0.243615; 0.736836; -0.147767; -0.0738833; 0.599054; 0.433315; 0.147767; -0.617025; -0.063899; -0.70888; 0.323489; -0.0219653; 0.591066; 0.185707; 0.559117; -0.804729; 0.587072; 0.0619022; -0.0439306; -0.089858; 0.103836; -0.361429; -0.22764; 0.684918; -0.139779; 0.173726; 0.187703; 0.621019; -0.503205; 0.229637; 0.30352; -0.640987; -0.806726; 0.52517; -0.239621; 0.313505; -0.331476; 0.700893; 0.355438; -0.648975; 0.56311; -0.680924; -0.427325; -0.652968; 0.439306; 0.333473; 0.287546; 0.48124; 0.0159748; 0.149763; -0.555123; 0.237625; 0.551129; 0.565107; -0.619022; -0.125801; 0.607041; 0.680924; -0.263584; -0.429322; -0.714871; 0.137782; 0.690908; 0.0459274; -0.539148; 0.732842; -0.271571; -0.676931; -0.30352; -0.0499211; -0.275565; -0.800735; 0.710877; 0.664949; -0.758801; 0.393379; 0.527167; -0.177719; 0.615028; -0.688912; -0.674934; -0.241618; 0.251602; -0.633; -0.559117; 0.11382; -0.051918; -0.337467; 0.515186; -0.549132; 0.423331; -0.161744; -0.507199; 0.105833; -0.299527; 0.0738833; 0.327483; 0.101839; -0.423331; -0.529164; -0.575091; -0.381397; 0.770782; -0.712874; 0.199685; -0.625013; -0.311508; -0.29753; 0.135786; 0.629006; -0.255596; -0.181713; 0.491224; 0.325486; 0.0399369; -0.724855; -0.171729; 0.800735; 0.215659; -0.129795; 0.483237; 0.077877; 0.451287; 0.704886; 0.479243; -0.0459274; -0.15176; -0.197688; -0.00399367; -0.469259; -0.33547; 0.577088; -0.247609; 0.0658959; 0.0798738; 0.513189; -0.611035;

0.121808; 0.195691; -0.748817; 0.70289; 0.417341; -0.347451; 0.205675; -0.439306; 0.593063; 0.247609; -0.756804; 0.455281; -0.788754; -0.535155; -0.760798; 0.331476; 0.225643; 0.179716; 0.37341; -0.0918549; 0.0419338; -0.662953; 0.129795; 0.4433; 0.457278; -0.726852; -0.233631; 0.499211; 0.573095; -0.371413; -0.537151; 0.794744; 0.0299527; 0.583079; -0.0619022; -0.646978; 0.625013; -0.379401; -0.78476; -0.41135; -0.157751; -0.383394; 0.70888; 0.603047; 0.55712; 0.750814; 0.285549; 0.419338; -0.285549; 0.507199; -0.796741; -0.782763; -0.349448; 0.143773; -0.74083; -0.666946; 0.00599055; -0.159748; -0.445296; 0.407356; -0.656962; 0.315502; -0.269574;

Taigi jau turime t skirstiniui reikaling1ą Y_1.Toliau mums reikėtų χ^2 skirstinio (Y_2), tačiau jo modeliavimui reikia standartinio normaliojo a.d. ir eksponentinių dydžių su vidurkiu 1. Iš likusios st. nomraliajam dydžiui skirtos sekos dalies (nuo 271 iki 540) modeliuojame dar vieną pagal st. normlauįį skirstinį pasiskirsčiusį a.d. Mums reikės naujo S. Kadangi dabar lentelėje nelabai patogu ieškoti 271 ir 272 skaitmenų, pasitelksiu kodą. Taigi kodas rado, kad $S(U_{271}, U_{272}) = 0.578357$. S atitinka reikalavimus, tad galime ramia galva generuoti dar vieną a.d. pagal N(0,1) pasiskirsčiusią seką. Gauname:

-1.04648; -0.00339764; -0.693122; -0.0577602; 0.373742; -0.010193; -0.903777; -1.08385; -1.162; -0.832426; 1.12802; 1.35566; 0.156292; -1.24694; -0.713508; -0.689724; 0.0475673; 0.886788; -0.618374; -0.49266; 0.65235; 0.370345; -0.11552; 1.33528; -0.475672; 1.17899; 0.183473; -0.404321; 0.638759; -0.0509648; 0.584397; 1.0159; 0.631964; -0.26162; -0.441695; -0.519841; -0.190269; -0.981923; -0.75428; 0.798449; -0.604783; -0.0713508; -0.0475672; 0.689724; -1.22316; 0.0237835; 0.149497; 1.29451; 1.0125; 0.526637; -0.774666; 0.166485; -0.930958; 0.825631; 0.237836; 1.28092; 0.591192; 1.22655; -1.09405; 1.27412; 0.380538; 0.200462; 0.122316; 0.451888; -0.339766; -0.112123; -1.3115; 0.0373743; 0.570806; 0.59459; 1.33188; -0.580999; 0.665941; 0.791654; -0.815438; -1.09744; 1.16879; -0.132509; 0.808642; -0.288801; -1.28431; 0.879993; -0.829028; 1.23335; -0.883391; -0.451888; -0.835824; 1.02269; 0.842619; 0.764473; 1.09405; 0.302391; 0.530035; -0.669338; 0.679531; 1.21296; 1.23675; -0.778063; 0.0611578; 1.3081; -1.31829; -0.17328; -0.455286; -0.941151; 0.509649; -1.3013; 0.353356; -0.642157; -1.22995; -0.186871; -0.876595; -0.241234; 0.190269; -0.193666; -1.08725; -1.26733; -1.34547; -1.0159; 0.944548; 1.17219; -0.0271813; 1.32169; -0.896981; -0.873198; -0.135906; 0.703315; -0.801847; -0.676134; 0.468877; 0.186871; -0.298994; 1.15181; -0.659145; 0.995513; 0; -0.587795; 0.455286; -0.234438; 0.400923; 0.832426; 0.448491; -0.445093; -0.625169; -0.703315; -0.373742; -1.1654; -0.937753; 0.614976; -0.788256; -0.254824; -0.231041; 0.506251; 1.34547; -0.15969; -0.0339765; 1.11103; 0.829028; 0.343163; -0.958139; -0.0169883; -1.11443; 0.642157; 0.0543625; 1.09744; 0.407719; 1.04308; -1.27752; 1.09065; 0.197064; 0.0169883; -0.0611578; 0.268415; -0.523239; -0.295596; 1.25713; -0.146099; 0.387333; 0.411116; 1.14841; -0.764473; 0.482467; 0.608181; -0.998911; -1.28092; 0.985321; -0.315982; 0.625169; -0.472274; 1.28431; 0.69652; -1.0125; 1.04988; -1.06686; -0.635362; -1.0193; 0.839221; 0.659145; 0.580999; 0.910572; 0.118918; 0.346561; -0.852812; 0.496058; 1.02949; 1.05327; -0.961537; -0.122316; 1.12462; 1.25034; -0.356754; -0.638759; -1.12462; 0.326175; 1.26733; 0.169883; -0.825631; 1.33868; -0.370345; -1.06007; -0.424707; 0.00679528; -0.37714; -1.27072; 1.3013; 1.22316; -1.19937; 0.761075; 0.988718; -0.210655; 1.13821; -1.08045; -1.05667; -0.31938; 0.519841; -0.98532; -0.859607; 0.285403; 0.00339772; -0.482467; 0.968332; -0.842619; 0.81204; -0.183473; -0.771268; 0.271813; -0.417912; 0.21745; 0.648952; 0.265017; -0.628567; -0.808642; -0.886788; -0.557216; -1.34887; -1.12123; 0.431502; -0.97173; -0.438298; -0.414514; 0.322777; 1.162; -0.343163; -0.21745; 0.92756; 0.645555; 0.15969; -1.14161; -0.200462; -1.2979; 0.458684;

Teliko gauti eksponentiškai pasiskirsčiusį a.d. Kadangi X=-lnU, galime iškart modeliuoti eksponentinį a.d. Trečia tolygios sekos dalis (541, 810) atrodo taip:

0.453086; 0.832099; 0.581481; 0.812346; 0.969136; 0.82963; 0.504938; 0.439506; 0.411111; 0.530864; 0.24321; 0.325926; 0.890123; 0.380247; 0.574074; 0.582716; 0.850617; 0.155556; 0.608642; 0.654321; 0.0703704; 0.967901; 0.791358; 0.318519; 0.660494; 0.261728; 0.9; 0.68642; 0.0654321; 0.814815; 0.045679; 0.202469; 0.062963; 0.738272; 0.67284; 0.644444; 0.764198; 0.476543; 0.559259; 0.123457; 0.61358; 0.807407; 0.816049; 0.0839506; 0.388889; 0.841975; 0.887654; 0.303704; 0.201235; 0.0246914; 0.551852; 0.893827; 0.495062; 0.133333; 0.919753; 0.298765; 0.0481481; 0.279012; 0.435802; 0.296296; 0.971605; 0.906173; 0.877778; 0.997531; 0.709877; 0.792593;

0.35679; 0.846914; 0.0407407; 0.0493827; 0.317284; 0.622222; 0.0753086; 0.120988; 0.537037; 0.434568; 0.258025; 0.785185; 0.12716; 0.728395; 0.366667; 0.153086; 0.532099; 0.281481; 0.512346; 0.669136; 0.52963; 0.204938; 0.139506; 0.111111; 0.230864; 0.94321; 0.0259259; 0.590123; 0.0802469; 0.274074; 0.282716; 0.550617; 0.855556; 0.308642; 0.354321; 0.77037; 0.667901; 0.491358; 0.0185185; 0.360494; 0.961728; 0.6; 0.38642; 0.765432; 0.514815; 0.745679; 0.902469; 0.762963; 0.438272; 0.37284; 0.344444; 0.464198; 0.176543; 0.259259; 0.823457; 0.31358; 0.507407; 0.516049; 0.783951; 0.0888889; 0.541975; 0.587654; 0.0037037; 0.901235; 0.724691; 0.251852; 0.593827; 0.195062; 0.833333; 0.619753; 0.998765; 0.748148; 0.979012; 0.135802; 0.996296; 0.671605; 0.606173; 0.577778; 0.697531; 0.409877; 0.492593; 0.0567901; 0.546914; 0.740741; 0.749383; 0.017284; 0.322222; 0.775309; 0.820988; 0.237037; 0.134568; 0.958025; 0.485185; 0.82716; 0.428395; 0.0666667; 0.853086; 0.232099; 0.981481; 0.212346; 0.369136; 0.22963; 0.904938; 0.839506; 0.811111; 0.930864; 0.64321; 0.725926; 0.290123; 0.780247; 0.974074; 0.982716; 0.250617; 0.555556; 0.00864198; 0.054321; 0.47037; 0.367901; 0.191358; 0.718518; 0.0604938; 0.661728; 0.3; 0.0864198; 0.465432; 0.214815; 0.445679; 0.602469; 0.462963; 0.138272; 0.0728395; 0.0444444; 0.164198; 0.876543; 0.959259; 0.523457; 0.0135802; 0.207407; 0.216049; 0.483951; 0.788889; 0.241975; 0.287654; 0.703704; 0.601235; 0.424691; 0.951852; 0.293827; 0.895062; 0.533333; 0.319753; 0.698765; 0.448148; 0.679012; 0.835802; 0.696296; 0.371605; 0.306173; 0.277778; 0.397531; 0.109877; 0.192593; 0.75679; 0.246914; 0.440741; 0.449383; 0.717284; 0.0222222; 0.475309; 0.520988; 0.937037; 0.834568; 0.658025; 0.185185; 0.52716; 0.128395; 0.766667; 0.553086; 0.932099; 0.681481; 0.912346; 0.0691358; 0.92963; 0.604938; 0.539506; 0.511111; 0.630864; 0.34321; 0.425926; 0.990123; 0.480247; 0.674074; 0.682716; 0.950617; 0.255556; 0.708642; 0.754321; 0.17037; 0.0679012; 0.891358; 0.418519; 0.760494; 0.361728; 0;

Modeliojame eksponentinį a.d.:

0.791672; 0.183804; 0.542176; 0.207829; 0.0313505; 0.186776; 0.683319; 0.822104; 0.888892; 0.633249; 1.41383; 1.12109; 0.116395; 0.966935; 0.554997; 0.540055; 0.161793; 1.86075; 0.496525; 0.424157; 2.65398; 0.0326252; 0.234005; 1.14407; 0.414767; 1.34045; 0.105361; 0.376266; 2.72674; 0.204794; 3.08612; 1.59717; 2.76521; 0.303444; 0.396248; 0.439367; 0.268929; 0.741197; 0.581142; 2.09186; 0.488444; 0.213927; 0.20328; 2.47753; 0.944462; 0.172005; 0.119173; 1.1917; 1.60328; 3.7013; 0.594476; 0.112243; 0.703073; 2.0149; 0.08365; 1.2081; 3.03347; 1.2765; 0.830566; 1.2164; 0.028806; 0.0985252; 0.130362; 0.00247218; 0.342664; 0.232446; 1.03061; 0.166157; 3.20053; 3.00815; 1.14796; 0.474458; 2.58616; 2.11207; 0.621688; 0.833403; 1.3547; 0.241836; 2.06231; 0.316912; 1.0033; 1.87675; 0.630926; 1.26769; 0.668756; 0.401768; 0.635577; 1.58505; 1.96965; 2.19722; 1.46593; 0.0584664; 3.65251; 0.527424; 2.52265; 1.29436; 1.26331; 0.596715; 0.156004; 1.17557; 1.03755; 0.260884; 0.403615; 0.710582; 3.98898; 1.02028; 0.0390232; 0.510826; 0.950831; 0.267315; 0.663948; 0.29346; 0.102621; 0.270546; 0.824916; 0.986607; 1.06582; 0.767445; 1.73419; 1.34993; 0.194244; 1.1597; 0.678441; 0.661553; 0.243409; 2.42037; 0.612535; 0.531616; 5.59842; 0.10399; 0.322009; 1.37891; 0.521167; 1.63444; 0.182322; 0.478434; 0.00123535; 0.290154; 0.021211; 1.99655; 0.00371059; 0.398085; 0.50059; 0.548566; 0.360209; 0.891899; 0.708073; 2.86839; 0.603464; 0.300105; 0.288505; 4.05798; 1.13251; 0.254494; 0.197247; 1.43954; 2.00569; 0.0428817; 0.723225; 0.189757; 0.847709; 2.70805; 0.158894; 1.46059; 0.0186921; 1.54954; 0.996591; 1.47129; 0.0998885; 0.174941; 0.20935; 0.0716419; 0.441284; 0.320307; 1.23745; 0.248145; 0.0262679; 0.0174351; 1.38383; 0.587787; 4.75112; 2.91284; 0.754235; 0.999941; 1.65361; 0.330564; 2.80521; 0.4129; 1.20397; 2.44854; 0.764789; 1.53798; 0.808156; 0.506719; 0.770108; 1.97854; 2.6195; 3.11352; 1.80669; 0.131769; 0.0415939; 0.647301; 4.29914; 1.57307; 1.53225; 0.725772; 0.23713; 1.41892; 1.246; 0.351398; 0.50877; 0.856393; 0.0493459; 1.22476; 0.110863; 0.628609; 1.14021; 0.35844; 0.802631; 0.387116; 0.179363; 0.36198; 0.989924; 1.18361; 1.28093; 0.922483; 2.2084; 1.64718; 0.278669; 1.39872; 0.819299; 0.79988; 0.332283; 3.80666; 0.743791; 0.652029; 0.0650325; 0.180841; 0.418513; 1.6864; 0.64025; 2.05264; 0.265703; 0.592241; 0.0703165; 0.383486; 0.0917363; 2.67168; 0.072969; 0.502629; 0.617101; 0.671168; 0.460665; 1.06941; 0.85349; 0.00992565; 0.733455; 0.394415; 0.381676; 0.0506437; 1.36432; 0.344405; 0.281937; 1.76978; 2.6897; 0.115009; 0.871034; 0.273787; 1.01686; 0

Dabar galime modeliuoti pagal χ^2 pasiskirsčiusį a.d. Naudosime 2 paskutines sekas. Jo formulė musų atveju $X = 2 * Y + Z^2$, kur Z bus narys iš antros sekos (N(0,1)), o Y iš trečios – eksponentinės. Taigi modeliuojame ir gauname:

2.67846; 0.36762; 1.56477; 0.418995; 0.202384; 0.373656; 2.18345; 2.81894; 3.12802; 1.95943; 4.10009; 4.08; 0.257217; 3.48873; 1.61909; 1.55583; 0.325849; 4.5079; 1.37544; 1.09103; 5.73353; 0.202406; 0.481354; 4.07112;

1.0558; 4.07091; 0.244384; 0.916008; 5.8615; 0.412186; 6.51375; 4.22639; 5.9298; 0.675332; 0.987592; 1.14897; 0.57406; 2.44657; 1.73122; 4.82125; 1.34265; 0.432945; 0.408823; 5.43077; 3.38504; 0.344575; 0.260695; 4.05915; 4.23173; 7.67995; 1.78906; 0.252203; 2.27283; 4.71147; 0.223866; 4.05694; 6.41645; 4.05743; 2.85807; 4.05618; 0.202421; 0.237235; 0.275685; 0.209147; 0.800769; 0.477463; 3.78124; 0.33371; 6.72687; 6.36985; 4.06982; 1.28648; 5.6158; 4.85085; 1.90831; 2.87119; 4.07548; 0.50123; 4.77851; 0.717229; 3.65607; 4.52789; 1.94914; 4.05653; 2.11789; 1.00774; 1.96976; 4.216; 4.6493; 4.97887; 4.12879; 0.208373; 7.58596; 1.50286; 5.50706; 4.05999; 4.05617; 1.79881; 0.315749; 4.06227; 3.81299; 0.551794; 1.01452; 2.30693; 8.23771; 3.73395; 0.202907; 1.43402; 3.41444; 0.56955; 2.09632; 0.645114; 0.241444; 0.578598; 2.83195; 3.57933; 3.94194; 2.56694; 4.36055; 4.07389; 0.389227; 4.06626; 2.16146; 2.08558; 0.505289; 5.33539; 1.86803; 1.52039; 11.4167; 0.2429; 0.733416; 4.08448; 1.47681; 4.25993; 0.364643; 1.30237; 0.209756; 0.63527; 0.203162; 4.68604; 0.208565; 0.994278; 1.39202; 1.59178; 0.8601; 3.14195; 2.29553; 6.11498; 1.82828; 0.665145; 0.630391; 8.37224; 4.07532; 0.534489; 0.395649; 4.11347; 4.69866; 0.203525; 2.36448; 0.379802; 2.93738; 5.82847; 0.320744; 4.12557; 0.203619; 4.1871; 3.62524; 4.13209; 0.238611; 0.350172; 0.422441; 0.21533; 1.15635; 0.727992; 4.05528; 0.517635; 0.202563; 0.203887; 4.0865; 1.75999; 9.73502; 6.19557; 2.50629; 3.64063; 4.27807; 0.760972; 6.00126; 1.04884; 4.05741; 5.38222; 2.55474; 4.1782; 2.75451; 1.41712; 2.57918; 4.66136; 5.67347; 6.56459; 4.44251; 0.27768; 0.203292; 2.02189; 8.84435; 4.20599; 4.17388; 2.3761; 0.489221; 4.10262; 4.05534; 0.830069; 1.42555; 2.97757; 0.205082; 4.05564; 0.250585; 1.93888; 4.07247; 0.854035; 2.72901; 0.954608; 0.358772; 0.866195; 3.59459; 4.0606; 4.05798; 3.28346; 4.99603; 4.27192; 0.601714; 4.09297; 2.80598; 2.71631; 0.76657; 7.88356; 2.45844; 2.04298; 0.21152; 0.361694; 1.0698; 4.31047; 1.99051; 4.7647; 0.565069; 1.77934; 0.214515; 0.941623; 0.230757; 5.7645; 0.216172; 1.40035; 1.8881; 2.12873; 1.23182; 3.95828; 2.96413; 0.206046; 2.41117; 0.980935; 0.935174; 0.205473; 4.07887; 0.806571; 0.611159; 4.39993; 5.79614; 0.255519; 3.04535; 0.587759; 3.71828; 0.210391;

Pagaliau galime modeliuoti a.d. pasiskirsčiusį pagal t. Prisiminkime formulę: $X = Y_1\sqrt{\frac{v}{Y_2}}$. Mūsų atveju Y_1 yra pirmosios sekos pagal N(0,1) elementas, Y_2 – paskutiniosios – pagal Chi kvadrat. $v = 3$(pagal sąlygą). Taigi generuojame ir gauname a.d. pagal t seką:

0.490287; -1.54588; 0.929007; -1.5335; -1.23009; -1.54466; 0.641333; 0.455254; 0.3872; 0.72889; 0.105901; 0.220884; -1.52758; 0.320345; 0.896982; 0.934445; -1.55109; -0.0146609; 1.05577; 1.30793; -0.112665; -1.23771; -1.51547; 0.21084; 1.3464; 0.131993; -1.5112; -1.40574; -0.117143; -1.53534; -0.132805; 0.0487886; -0.119307; -1.46041; -1.39212; 1.24871; -1.48814; 0.555009; 0.835903; -0.0551307; 1.08052; -1.52961; -1.53623; -0.0994372; 0.338373; -1.5496; -1.5309; 0.190551; 0.0470765; -0.143524; 0.806766; -1.52203; 0.610244; -0.043022; -1.46198; 0.183734; -0.131078; 0.15625; 0.44599; 0.180317; -1.2146; -1.4983; -1.5414; -1.03609; -1.43006; -1.51662; 0.273911; -1.55067; -0.136019; -0.130186; 0.209159; 1.1252; -0.108002; -0.0581029; 0.751106; 0.442929; 0.126779; -1.50954; -0.0506301; -1.44979; 0.293031; -0.0178792; 0.733289; 0.159702; 0.665444; -1.38847; 0.724513; 0.0522176; -0.0352886; -0.0697513; 0.088511; -1.37139; -0.143154; 0.967698; -0.103168; 0.149335; 0.161427; 0.801996; -1.55108; 0.197342; 0.269225; -1.49459; -1.38726; 0.598885; -0.144605; 0.281009; -1.27457; 1.01376; 0.333169; -1.48944; 0.673637; -1.46839; -1.5063; -1.48684; 0.452153; 0.305296; 0.250849; 0.520252; 0.0132503; 0.128517; -1.54116; 0.204105; 0.649293; 0.677763; -1.50833; -0.0943327; 0.769284; 0.956493; -0.135117; -1.50879; -1.44582; 0.118082; 0.984735; 0.0385418; -1.54645; 1.11225; -1.02704; -1.47104; -1.16635; -0.0399431; -1.04511; -1.3909; 1.0436; 0.912866; -1.41714; 0.38439; 0.602654; -0.124479; 0.787834; -1.46307; -1.47237; -0.144634; 0.215871; -1.49967; -1.5396; 0.0972021; -0.041485; -1.29564; 0.580306; -1.54333; 0.42782; -0.116041; -1.55117; 0.0902483; -1.14971; 0.0625389; 0.297907; 0.0867741; -1.50105; -1.54885; -1.53255; -1.42359; 1.2415; -1.44714; 0.171749; -1.50466; -1.19881; -1.14129; 0.116343; 0.821221; -0.141888; -0.126446; 0.537433; 0.295464; 0.0334434; -1.43922; -0.121418; 1.35424; 0.18544; -0.0969031; 0.523657; 0.0659896; 0.470968; 1.0256; 0.516863; -0.0368448; -0.110356; -0.13364; -0.00328185; -1.54241; -1.28871; 0.70295; -0.144209; 0.0556525; 0.0677165; 0.576642; -1.51312; 0.104161; 0.168313; -1.42357; 1.01966; 0.41891; -1.32889; 0.176894; -1.52002; 0.73771; 0.212519; -1.41842; 0.477351; -1.39827; -1.5475; -1.41587; 0.302823; 0.193949; 0.154523; 0.356928; -0.0711787; 0.0351409; -1.48029; 0.111122; 0.45837; 0.480563; -1.43791; -0.144122; 0.551462; 0.694472; -1.39876; -1.54699; 1.33087; 0.0249881; 0.715824; -0.049119; -1.49073; 0.811559; -1.41883; -1.40075; -1.48318; -0.113802; -1.42826; 1.03756; 0.760149; 0.661377; 1.17171; 0.248593; 0.421867; -1.08958; 0.565751; -1.39334; -1.40199; -1.33526; 0.123301; -1.42875; -1.47766; 0.00494658; -0.114928; -1.5258; 0.404312; -1.48423; 0.283394; -1.01795;

Kadangi t skirstiniui teko prigeneruoti daug sekų, daugiau sekų nebegeneruosiu.

4 užduotis

Naudodami sugeneruotą geresniąją pseudoatsitiktinių skaičių seką ir pasirinkdami tankius (tolygiai pasiskirsčiusio intervale [0,4] atsitiktinio dydžio ir kitą savo nuožiūra) suskaičiuokite intergralą:

$$\int_0^4 e^x dx$$

Teorija

$$\mathrm{I} \approx \frac{1}{N}\sum_{j=1}^{N}\frac{g(\xi_j)}{p_\xi(\xi_j)},$$

kur I – tankis, N – a.d. kiekis, $g(\xi_j)$ – norima suskaičiuoti funkcija nuo a.d., $p_\xi(\xi_j)$ – a.d. tankio funkcija.

Sprendimas

Integralas tolygiai pasiskirsčiusio a.d.

Sau kaip atskaitos tašką galime suskaičiuoti paprasto integralo nuo duotos funkcijos vertę:

$$\int_0^4 e^x dx = e^x|_0^4 = 53.5982.$$

Pirma mums reikia rasti tankio funkciją. Tarkime

$$p_\xi(\xi_j) = \frac{1}{4-0} = \frac{1}{4}.$$

Jeigu atsitiktinis dydis U yra tolygiai pasiskirstęs intervale (0, 1), tai reikėtų imti $\xi = 4U$. Tai

$$\mathrm{I} \approx \frac{1}{809}\sum_{j=1}^{809}\frac{e^{4U_j}}{\frac{1}{4}} = \frac{4}{809}\sum_{j=1}^{809} e^{4U_j} = 53.5269$$

Puiku, gavome artimą įprastam integralui MK būdu suskaičiuotą integralą (809, ne 810, nes pas mus sugeneruotoje sekoje yra vienas 0 (jį išmesime)).

Integralas tiesiškai pasiskirsčiusio a.d.

Tarkime

$$p_\xi(\xi_j) = \frac{x}{8}, \qquad \int_0^4 \frac{x}{8} dx = \frac{1}{8} * \frac{x^2}{2}|_0^4 = \frac{16}{16} = 1$$

Tada

$$\int_0^{\xi} \frac{x}{8} dx = \frac{x^2}{16}|_0^{\xi} = \frac{\xi^2}{16} = U \to \xi^2 = 16U \to \xi = 4\sqrt{U}.$$

Taigi

$$\mathrm{I} \approx \frac{1}{809} \sum_{j=1}^{809} \frac{e^{4\sqrt{U_j}}}{\frac{4\sqrt{U_j}}{8}} = \frac{2}{809} \sum_{j=1}^{809} \frac{e^{4\sqrt{U_j}}}{\sqrt{U_j}} = 53.489.$$

Išvada

Gauname, kad pirmasis integralas yra arčiau originaliojo negu antrasis. Vadinasi pirmosios sekos tankio grafikas yra artimesnis funkcijos grafikui.

5 užduotis

Sugeneruokite Markovo grandinę, kuri nebūtų gimimo-mirties procesas ir kurios vienintelis stacionarus skirstinys būtų

$$\pi = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}, 0\right).$$

Naudokite geresniu algoritmu gautus pseudoatsitiktinius skaičius.

Teorija

1 apibrėžimas

Tegu $(X_0, X_1, ...)$ yra Markovo grandinė su būsenų aibe $S = \{s_1, s_2, ...\}$ ir perėjimo matrica P. Vektorius – eilutė $\pi = (\pi_1, \pi_{2,} ...)$ yra vadinamas MG stacionariuoju skirstiniu, jei:

1) $\pi_i \geq 0, \forall i = 1, ..., k$ ir $\sum_{i=1}^{k} \pi_i = 1$

2) $\pi P = \pi$, t.y. $\sum_{i=1}^{k} \pi_i P_{ij} = \pi_j, \forall j = 1, ..., k$

2 apibrėžimas

Tegul $(X_0, X_1, ...)$ yra Markovo grandinė su baigtine būsenų aibe S ir perėjimo matrica $\mathbb{P}$. Jei π yra šios Markovo grandinės apgręžiamas skirstinys, tai jis yra ir stacionarus šios grandinės skirstinys.

3 apibrėžimas

Tegul $(X_0, X_1, \dots)$ yra Markovo grandinė su baigtine būsenų aibe S ir perėjimo matrica $\mathbb{P}$. Tikimybinis skirstinys π aibėje S yra vadinamas apgręžiamu, jei $\forall\, i, j = 1, \dots, k$ teisinga lygybė: $\pi_i P_{ij} = \pi_j P_{ji}$.

Teorema

Tegul $(X_0, X_1, \dots)$ yra Markovo grandinė su baigtine būsenų aibe S ir perėjimo matrica $\mathbb{P}$. Be to, tegul matrica $\mathbb{P}$ tenkina savybes, tada ji yra gimimo – mirties procesas:

1. $P_{ij} > 0$, jei $|i - j| = 1$,
2. $P_{ij} = 0$, jei $|i - j| \geq 2$.

Sprendimas

Pirma ieškosime perėjimų matricos. Žinome, kad ji negali būti gimimo-mirties procesas bei turi tenkinti 1 ir 3 apibrėžimus. Taigi iš 3 apibrėžimo (apgręžiamumo) gauname:

$$\frac{P_{ij}}{P_{ji}} = \frac{\pi_j}{\pi_i}, \qquad tai:$$

$$\frac{P_{12}}{P_{21}} = \frac{1/3}{1/2} = \frac{2}{3}, \qquad \frac{P_{13}}{P_{31}} = \frac{1/6}{1/2} = \frac{1}{3}, \quad \frac{P_{14}}{P_{41}} = \frac{0}{1/2} = 0, \quad \frac{P_{23}}{P_{32}} = \frac{1/6}{1/3} = \frac{1}{2}$$

$$\frac{P_{24}}{P_{42}} = 0, \quad \frac{P_{34}}{P_{43}} = 0$$

Iš to bei iš pirmo apibrėžimo antro teiginio matome (j=4 stulpelio suma turi būti = 0) matome, kad P_{14}, P_{24}, P_{34} bus 0. Reikia pasirūpinti, kad bent $P_{13}, P_{31}, P_{41}, P_{42} \neq 0$ (kad nebūtų gimimo – mirties procesas). Taigi iš to ką gavome, mūsų matrica priliminariai atrodo taip:

$$\mathbb{P} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & 0 \\ 1.5P_{12} & P_{22} & P_{23} & 0 \\ 3P_{13} & 2P_{23} & P_{33} & 0 \\ P_{41} & P_{42} & P_{43} & P_{44} \end{pmatrix}$$

Galime rasti daug matricų variantų, aš tiesiog ieškau žiūrėdamas, kas tiktų trečioje eilutėje (nes ten 2 gana dideli daugikliai, kurie sumažina variantų skaičių). Pasirenku, kad $P_{13} = 0.1, P_{23} = 0.1$, tada $P_{33} = 1 - 0.5 = 0.5$. Jau turime trečią stulpelį, galime patikrinti, ar galioja pirmo apibrėžimo 2 sąlyga:

$$0.3 * \frac{1}{2} + 0.2 * \frac{1}{3} + 0.5 * \frac{1}{6} + P_{43} * 0 = \frac{1}{6} = \pi_{j=3}$$

Gauname teisingai, todėl ramia širdimi galime skaičiuoti toliau. Daugiau nerašysiu, kaip skaičiavau, nes sprendimas analogiškas. Tik paminėsiu, kad 4 eilutėje mes neturime jokių apribojimų (išskyrus, kad nelygūs 0), todėl parenkame bet ką. Taigi:

$$\mathbb{P} = \begin{pmatrix} 0.7 & 0.2 & 0.1 & 0 \\ 0.3 & 0.6 & 0.1 & 0 \\ 0.3 & 0.2 & 0.5 & 0 \\ 0.1 & 0.4 & 0.3 & 0.2 \end{pmatrix}$$

Paskirkime bet kokį pradinį skirstinį $\mu^{(0)} = (0.1, 0.2, 0.3, 0.4)$. Apsibrėžkime inicijuojančią funkciją ψ: [0,1] → S:

$$\psi(x) = \begin{cases} 1, x \in [0, 0.1) \\ 2, x \in [0.1, 0.3) \\ 3, x \in [0.3, 0.6) \\ 4, x \in [0.6, 1] \end{cases}$$

Toliau sukonstruojame atnaujinimo funkciją $\phi: S \times [0,1] \to S$, ji leidžia gauti X_{n+1} turint X_n:

$$\phi(s_1, x) = \begin{cases} 1, x \in [0, 0.7) \\ 2, x \in [0.7, 0.9) \\ 3, x \in [0.9, 1) \\ 4, x \in \emptyset \end{cases}$$

$$\phi(s_2, x) = \begin{cases} 1, x \in [0, 0.3) \\ 2, x \in [0.3, 0.9) \\ 3, x \in [0.9, 1) \\ 4, x \in \emptyset \end{cases}$$

$$\phi(s_3, x) = \begin{cases} 1, x \in [0, 0.3) \\ 2, x \in [0.3, 0.5) \\ 3, x \in [0.5, 1) \\ 4, x \in \emptyset \end{cases}$$

$$\phi(s_1, x) = \begin{cases} 1, x \in [0, 0.1) \\ 2, x \in [0.1, 0.5) \\ 3, x \in [0.5, 1) \\ 4, x \in \emptyset \end{cases}$$

Dabar galima generuoti MG su būsenų aibe $S = \{1,2,3,4\}$, pradiniu skirstiniu $\mu^{(0)} = (0.1, 0.2, 0.3, 0.4)$ ir perėjimų matrija $\mathbb{P}$. Naudojame antrą, geresnę seką (m=810) konvertuotą į intevalą [0,1). Taigi gauname

4; 2; 3; 1; 1; 1; 2; 2; 2; 2; 2; 2; 1; 2; 3; 3; 1; 1; 3; 3; 2; 2; 1; 1; 3; 3; 1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 1; 2; 2; 2; 3; 1; 1; 1; 2; 1; 1; 1; 1; 1; 2; 1; 2; 2; 1; 1; 1; 1; 2; 2; 2; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 2; 2; 2; 2; 2; 1; 1; 1; 1; 1; 2; 2; 2; 1; 2; 2; 2; 2; 2; 1; 1; 3; 2; 2; 2; 2; 1; 1; 1; 1; 1; 2; 2; 2; 1; 3; 3; 1; 2; 1; 1; 1; 2; 2; 2; 2; 2; 2; 1; 1; 2; 2; 1; 1; 2; 3; 2; 1; 1; 1; 3; 3; 1; 3; 2; 1; 1; 1; 1; 1; 3; 3; 1; 2; 2; 2; 2; 1; 1; 1; 1; 1; 1; 1; 1; 1; 2; 1; 2; 2; 1; 1; 1; 1; 2; 2; 1; 1; 1; 1; 3; 1; 1; 1; 1; 1; 1; 2; 2; 2; 2; 2; 2; 1; 1; 3; 3; 2; 2; 2; 2; 3; 3; 2; 2; 2; 2; 1; 1; 2; 2; 2; 2; 2; 1; 1; 1; 1; 3; 3; 1; 1; 1; 2; 2; 1; 2; 1; 1; 1; 2; 2; 2; 2; 2; 2; 1; 1; 2; 2; 1; 1; 2; 2; 1; 3; 3; 3; 2; 1; 2; 1; 1; 1; 1; 1; 3; 3; 3; 3; 3; 3; 2; 2; 1; 1; 1; 1; 1; 1; 1; 1; 1; 2; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 3; 3; 1; 1; 1; 1; 2; 1; 1; 2; 2; 2; 3; 2; 3; 3; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2; 1; 1; 2; 1; 1; 1; 2; 1; 1; 1; 3; 3; 3; 1; 1; 1; 2; 2; 3; 3; 3; 1; 3; 3; 3; 3; 3; 2; 2; 1; 1; 2; 2; 3; 1; 2; 2; 1; 1; 3; 2; 2; 2; 1; 2; 1; 3; 1; 1; 1; 2; 2; 2; 2; 2; 2; 1; 2; 3; 3; 1; 1; 3; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 2; 3; 2; 3; 1; 1; 1; 2; 1; 1; 1; 1; 1; 3; 1; 2; 2; 1; 1; 1; 1; 2; 2; 2; 1; 1; 1; 1; 1; 2; 1; 1; 1; 1; 3; 2; 2; 3; 3; 3; 1; 1; 1; 2; 2; 2; 2; 2; 1; 2; 2; 2; 2; 2; 2; 2; 3; 2; 2; 2; 3; 1; 1; 2; 1; 1; 2; 2; 2; 2; 3; 3; 1; 2; 1; 1; 1; 2; 2; 2; 2; 2;

2; 1; 1; 2; 2; 1; 1; 3; 3; 2; 1; 1; 1; 3; 3; 1; 3; 2; 1; 1; 1; 1; 1; 3; 3; 1; 2; 2; 2; 3; 1; 1; 1; 1; 1; 1; 1; 1; 2; 1; 2; 2; 1; 1; 1; 1; 2; 2; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 2; 2; 2; 2; 2; 1; 1; 1; 1; 1; 2; 2; 2; 3; 3; 3; 2; 2; 2; 1; 1; 2; 2; 2; 2; 2; 1; 1; 1; 1; 3; 3; 2; 2; 1; 2; 2; 1; 2; 1; 1; 1; 2; 2; 2; 2; 2; 2; 1; 1; 2; 2; 1; 1; 2; 2; 2; 1; 1; 1; 2; 2; 1; 3; 1; 1; 1; 1; 1; 3; 3; 3; 3; 3; 3; 2; 2; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 2; 1; 2; 2; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 1; 1; 1; 1; 1; 1; 2; 2; 2; 2; 2; 2; 1; 1; 3; 3; 2; 2; 2; 2; 3; 3; 2; 2; 2; 2; 1; 1; 2; 2; 2; 2; 2; 1; 1; 1; 1; 3; 3; 1; 1; 1; 2; 2; 3; 3; 3; 1; 3; 3; 3; 3; 2; 2; 1; 1; 2; 2; 1; 1; 2; 2; 1; 1; 3; 2; 2; 2; 1; 2; 1; 3; 1; 1; 1; 3; 3; 3; 3; 3; 3; 1; 2; 3; 3; 1; 1; 1; 1; 1; 1; 1; 2; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 2; 3; 3; 1; 1; 1; 1; 2; 1; 1; 2; 2; 2; 3; 1; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2; 1; 1; 1; 1; 2; 1; 1; 1; 2; 1; 1; 1; 3; 3; 3; 1; 1; 1; 2; 2; 3; 3; 3; 1; 3; 3; 3; 3; 3; 2; 2; 3; 2; 2; 2; 3; 1; 2; 2; 1; 1; 2; 2; 2; 2; 1;

Išvados

Buvo penkios užduotys. Pirmoji užduotis parodė, kaip sugeneruoti optimalias galingiausias pseudoatsitiktinių skaičių sekas naudojant tiesinį kongruentinį metodą. Gautas dvi sekas (mažesnę ir didesnę, konvertuotas į [0;1) intervalą) antroje užuotyje tikrinau dviem testais: maksimumo ir kėlinių testu. Testų rezultatai nebuvo geri, tačiau jie parodė, kad antroji (didesnė) seka yra geresnė. Tada trečioje užduotyje mes generavome atsitiktinius dydžius iš mūsų gautos sekos. Reikėjo sugeneruoti t skirstinį, tačiau jam dar reikėjo chi kvadrat, normaliojo ir eksponentinio skirstinių. Ketvirtoje užduotyje skaičiavome intergralą Markovo būdu. Pastebėjome, kad abu skaičiavimo variantai (kai mūsų dydis tolygiai arba tiesiškai pasiskirstęs) yra gana tikslūs, tačiau pirmasis tikslesnis. Galiausiai iš mūsų konvertuotos jau antroje užduotyje geresnės sekos generavome markovo grandinę su stacionariu skirstiniu.

Priedai

Main funkcija

```
int m1 = 656, m2 = 810;
    vector<int> aVariantai1; //avariantai - vektorius skirtas laikyti visas tinkamas a reiksmes kazkokiam moduliui (siuo atveju 656)
    vector<int> aVariantai2; //skirtas 810 moduliui
    vector<int> pirminiai;
    vector<int> cmasyvas;
    vector<int> seka1, seka2;
    vector<float> tolyg1, tolyg2;

    pirminiai.push_back(2);
    pirminiai.push_back(3);
    //sukuriame pirminiu skaiciu vektoriu (mechaniskai irasome maksimalia verte)
    for (int i = 5; i <= 810; i++) {
        if (arPirminis(i)) {
            pirminiai.push_back(i);
        }
    }

    //atrenkame visus a variantus kiekvienam m:
    aparinkimas(m1, pirminiai, aVariantai1);
    aparinkimas(m2, pirminiai, aVariantai2);

    //renkame geriausia a pagal galia:
    int a1 = galingumas(m1, aVariantai1);
    cout << endl;

    int a2 = galingumas(m2, aVariantai2);
    cout << endl;

    //renkame c pagal maziausios koreliacijos teorini testa
    int c1 = cparinkimas(pirminiai, cmasyvas, m1, a1);
    cout << "c1 = " << c1 << endl;

    int c2 = cparinkimas(pirminiai, cmasyvas, m2, a2);
```

```
    cout << "c2 = " << c2 << endl;

    //spausdiname sekas
    sekos(a1, c1, m1, seka1);
    sekos(a2, 637, m2, seka2);

    //konvertuojame sekas i tolygias
    konvertavimas(m1, seka1, tolyg1);
    konvertavimas(m2, seka2, tolyg2);

    //testai
    //maksimumo
    maksimumas(m1, 6, tolyg1);
    maksimumas(m2, 6, tolyg2);
    //keliniu
    keliniai(m1, 3, tolyg1);
    keliniai(m2, 3, tolyg2);

    //a.d. generavimas
    adgeneravimas(m2, 3, tolyg2);

    //integralu skaiciavimas
    integralas(tolyg2);

    //markovo grandine
    markovas(tolyg2);
```

Pirma užduotis

```
void aparinkimas(int m, vector<int> pirminiai, vector<int> &amasyvas) {
    //atrenkame tik tuos a, kur jei p|m => p|a-1
    //tam pirma reikia atrinkti pirminius skaicius, kurie dalina m

    //pirma uzpildome vektoriu visais a variantais (galimais ir negalimais)
    for (int i = 2; i < m; i++) {
        amasyvas.push_back(i);
    }

    vector<int> temp;

    for(int sk : pirminiai){
        //tikriname, ar m dalinasi is kazkokio pirminio skaiciaus sk
        if (m % sk == 0) {
            //jei taip, tai visi a-1, esantys amasyve (avariantai1, avariantai2) turi dalintis is to pirminio
sk
            for (int aSk : amasyvas) {
                if ((aSk - 1) % sk == 0) {
                    //jei tenkina salyga, tinkama a keliame i laikina vektoriu, kuri uzpilde padarome
naujuoju avariantai vektoriumi
                    temp.push_back(aSk);
                }
            }
            amasyvas = temp;
            temp.clear();
        }
    }

    //toliau tikrinsime 4|m => 4|(a-1) (kodas beveik analogiskas ankstesniam)
    if (m % 4 == 0) {
        for (int aSk : amasyvas) {
            if ((aSk - 1) % 4 == 0) {
                temp.push_back(aSk);
            }
        }
        amasyvas = temp;
        temp.clear();
    }
}

//galingumo ieskojimas
int galingumas(int m, vector <int> amasyvas) {
    int index = 0;
    int iMax = 0;
    int galia = 0;
```

```
    int gMax = 0;
    long long bkelimas = 0;
    cout << "Galimi a su galiomis:" << endl;
    //einame per galimu a masyva (nustatytas anksciau)
    for (int aSk : amasyvas) {
        int n = 0;
        while (galia == 0) {
            n++;
            //keliame b laipsniu n tol, kol gauname kad jis yra m1 ar m2 kartotinis
            bkelimas = pow(aSk-1, n);
            if (bkelimas % m == 0) {
                galia = n;
                cout << "a = " << aSk << " su galia = " << galia << endl;
            }
        }
        //randame is visu a geriausia a, vis pakeisdami, jei atsiranda stipresnis. Jei lygus, nieko nedarom
        if (galia > gMax) {
            gMax = galia;
            iMax = index;
        }
        index++;
        galia = 0;
    }
    cout << "Galingiausias a buvo " << amasyvas[iMax] << " su galia " << gMax << endl;
    return amasyvas[iMax];
}

void sekos(int a, int c, int m, vector<int>& seka) {
    cout << "Seka (" << a << "X + " << c << ") mod " << m << ":" << endl;
    int x = 0;
    int counter = 2;
    cout << setw(3) << x << " ";
    do{
        x = (a * x + c) % m;
        seka.push_back(x);
        cout << setw(3) <<x << " ";
        if (counter%20 == 0) cout << "("<<counter/20 <<")"<< endl;
        counter++;
    } while (x != 0);
    cout << endl;
}

bool arPirminis(int n)
{
    if (n % 2 == 0 || n % 3 == 0)
        return false;

    for (int i = 5; i * i <= n; i = i + 6)
        if (n % i == 0 || n % (i + 2) == 0)
            return false;

    return true;
}
```

Antra užduotis

```
void konvertavimas(int m, vector<int> seka, vector<float>& tolyg) {
    float temp;
    for (int sk : seka) {
        temp = sk * 1.0 / (m * 1.0);
        tolyg.push_back(temp);
    }
}

void maksimumas(int m, int k, vector<float> tolyg) {
    float gr;
    float maxi = 0, temp;
    vector<float> maksimumai;
    gr = ceil(m * 1.0 / (k * 1.0));
    int pradz, pab;
    for (int i = 0; i < gr; i++) {
        pradz = i*k;
        if (pradz + k - 1 < m) {
            pab = pradz + k - 1;
```

```
        }
        else pab = m - 1;
        for (int j = pradz; j <= pab; j++) {
            temp = tolyg[j];
            if (temp > maxi) maxi = temp;
        }
        maksimumai.push_back(maxi);
        maxi = 0;
    }


    //surusiuojame vektoriu didejimo tvarka
    sort(maksimumai.begin(), maksimumai.end());

    maxi = -100;
    //ieskome maksimumo
    for (int i = 1; i <= gr; i++) {
        temp = (i*1.0) / gr - maksimumai[i];
        if (temp > maxi) maxi = temp;
    }
    cout << "m = " << m << "D vertes:" << endl;
    cout << "maksi: " << maxi << "; ";

    float maximinus = -100;
    for (int i = 1; i <= gr; i++) {
        temp = maksimumai[i] - ((i * 1.0) - 1) / gr;
        if (temp > maximinus) maximinus = temp;
    }
    cout << "maksiminus: " << maximinus << endl;

    maksimumai.clear();
}

void keliniai(int m, int k, vector<float> tolyg) {
    float gr;
    gr = floor(m * 1.0 / (k * 1.0));
    float temp[3];
    int pradz, pab;
    int v1 = 0, v2 = 0, v3 = 0, v4 = 0, v5 = 0, v6 = 0;
    int pos = 0;
    for (int i = 0; i < gr; i++) {
        pradz = i * k;
        pab = pradz + k - 1;

        for (int j = pradz; j <= pab; j++) {
            temp[pos] = tolyg[j];
            pos++;
        }

        //cia tikrink jau tuos if 6 turbut
        if      (temp[0] < temp[1] && temp[1] < temp[2]) v1++;
        else if (temp[0] < temp[2] && temp[2] < temp[1]) v2++;
        else if (temp[1] < temp[0] && temp[0] < temp[2]) v3++;
        else if (temp[1] < temp[2] && temp[2] < temp[0]) v4++;
        else if (temp[2] < temp[0] && temp[0] < temp[1]) v5++;
        else if (temp[2] < temp[1] && temp[1] < temp[0]) v6++;

        pos = 0;
    }
    cout << v1 << " " << v2 << " " << v3 << " " << v4 << " " << v5 << " " << v6 << endl;
}
```

Trečia užduotis

```
void adgeneravimas(int m, int v, vector<float> tolyg) {
    int mtrecdalis = floor((m * 1.0) / 3);
    float V, x, s;
    vector<float> norm1, norm2, eksp, chi, stud;
    s = 0.7759;
    //norm seka skirta Y1
    for (int i = 0; i < mtrecdalis; i++) {
        V = 2 * tolyg[i] - 1;
        x = V * sqrt((-2 * log(s)) / s);
        norm1.push_back(x);
```

```
    }

    float V1, V2;
    s = pow(tolyg[mtrecdalis] * 2 - 1, 2) + pow(tolyg[mtrecdalis + 1] * 2 - 1, 2);
    cout << "s=" << s << endl;
    //norm seka skirta Y2
    for (int i = mtrecdalis; i < mtrecdalis*2; i++) {
        V = 2 * tolyg[i] - 1;
        x = V * sqrt((-2 * log(s)) / s);
        norm2.push_back(x);
    }

    //eksp seka skirta Y2
    for (int i = mtrecdalis*2; i < m-1; i++) {//m-1, nes paskutinis skaicius 0 (zr. zemiau)
        x = -1 * log(tolyg[i]);
        eksp.push_back(x);
    }
    eksp.push_back(0);//nes paskutinis narys pas mane sekoje yra 0, tai vietoje jo keiciu i 1, tada ln(1)
gaunasi 0

    //chi kvadrat seka Y2
    for (int i = 0; i < mtrecdalis; i++) {
        x = 2 * eksp[i] + norm2[i] * norm2[i];
        chi.push_back(x);
    }

    //stud t seka
    for (int i = 0; i < mtrecdalis; i++) {
        x = norm1[i] * sqrt(v / chi[i]);
        stud.push_back(x);
    }

    norm1.clear();
    norm2.clear();
    eksp.clear();
    chi.clear();
    stud.clear();
}
```

Ketvirta užduotis

```
void integralas(vector<float> tolyg) {
    float integ = 0;
    //tolygiai tolydziai pasiskirstes integralas
    for (auto& ad : tolyg) {
        if (ad != 0) {
            integ = integ + pow(2.71828, 4 * ad);
        }
    }
    integ = (4.0 / 809.0) * integ;
    //cout << "Tolygaus [0;4] integralas I = " << integ << endl;

    //tiesiskai pasiskirstes integralas
    integ = 0;
    for (auto& ad : tolyg) {
        if (ad != 0) {
            integ = integ + (pow(2.71828, 4 * sqrt(ad))) / sqrt(ad);
        }
    }
    integ = (2.0 / 809.0) * integ;
    //cout << "Tiesiskai pasiskirsciusio a.d. integralas I = " << integ << endl;
}
```

Penkta užduotis

```
void markovas(vector<float>& tolyg) {
    vector<float> MG;
    //inicijuojanti funkcija:
    if (tolyg[0] < 0.1) MG.push_back(1);
    else if (tolyg[0] < 0.3) MG.push_back(2);
    else if (tolyg[0] < 0.6) MG.push_back(3);
    else MG.push_back(4);

    int busena;
```

```
    //atnaujinimo funkcija:
    for (int i = 1; i < tolyg.size(); i++) {
        busena = MG.back();
        if (busena == 1) {
            if (tolyg[i] < 0.7) MG.push_back(1);
            else if (tolyg[i] < 0.9) MG.push_back(2);
            else MG.push_back(3);
        }

        else if (busena == 2) {
            if (tolyg[i] < 0.3) MG.push_back(1);
            else if (tolyg[i] < 0.9) MG.push_back(2);
            else MG.push_back(3);
        }

        else if (busena == 3) {
            if (tolyg[i] < 0.3) MG.push_back(1);
            else if (tolyg[i] < 0.5) MG.push_back(2);
            else MG.push_back(3);
        }

        else {
            if (tolyg[i] < 0.1) MG.push_back(1);
            else if (tolyg[i] < 0.5) MG.push_back(2);
            else if (tolyg[i] < 0.8) MG.push_back(3);
            else MG.push_back(4);
        }
    }

    for (auto& bus : MG) {
        cout << bus << "; ";
    }
    MG.clear();
}
```