finaltest

zza

2025-07-02

1

```
matrix <- matrix(c(9, 4, 12, 2,
                   5, 0, 7, 9,
                   2, 6, 8, 0,
                   9, 2, 9, 11),
                 nrow = 4, byrow = TRUE)


inverse_matrix <- solve(matrix)


verification <- matrix %*% inverse_matrix
all.equal(verification, diag(4))
```

```
## [1] TRUE
```

2

a

```
xVec <- sample(0:999, 250, replace=T)
yVec <- sample(0:999, 250, replace=T)

## a
newVec <- yVec[2:250] - xVec[1:249]
newVec
```

```
##   [1] -192   48 -755  169  487  285 -296 -422   15  225  363 -335  294   35   -8
##  [16]  395 -730 -465 -583  165  156   98  725  350   54  662 -307   13 -199  397
##  [31]  831  248 -445  723 -534  -43 -415  -53 -465 -793 -722  214   75 -241   50
##  [46] -787  272  245 -374    0 -789 -143  -92  781 -270  154 -539  281  570  -54
##  [61] -686  262 -354  816 -653   -5  415  366  189 -149 -139  381  -63  -45  612
##  [76]  193   -6  332 -587 -710 -238 -110  770 -178  470 -375 -904  -63 -680  109
##  [91] -269  571 -578 -323  178  145 -115   87  435  409  323  467  161  290  350
## [106] -190  -69  228  231 -791 -532 -268 -387   -2  522  223  961   44 -112 -124
## [121]  563 -693 -430  743  721  406 -675  492 -778   34   46 -356 -412  581 -747
## [136]  -61  -19   76 -524  512  305  176   52  -56  302 -112 -557  107   41  148
## [151]  569 -116 -521  302 -529 -257  -75  128 -633   31  411  108  213  668 -308
## [166]  395  -92 -496 -429 -446  -93   86   -7  114 -499  199  -11   79 -423 -733
## [181]  686  414  189  969  584  903  -75 -344  902  176 -747 -535  -32 -235  218
## [196]   75 -509  535 -605 -571 -830  -47  425  -24   54 -624  340  -13  526 -140
## [211] -803  798  123 -525 -349   44  -50 -711  631   36  268  615 -329  240  368
## [226]  913 -331  422 -319  849 -180 -211 -184  751 -285  459 -110  149 -514  -96
## [241]  -35 -541  178  -15 -241 -754  142 -179  200
```

b

```
values_greater_600 <- yVec[yVec > 600]
values_greater_600
```

```
##  [1] 714 988 927 885 722 909 610 914 882 903 754 838 874 761 692 680 870 958 663
## [20] 912 929 904 828 999 800 908 997 973 989 960 767 645 996 606 853 681 801 977
## [39] 737 837 709 755 867 824 765 622 975 810 923 794 713 718 909 910 827 813 683
## [58] 857 899 638 853 916 749 982 935 933 664 970 762 980 971 788 964 885 745 808
## [77] 942 642 830 873 854 757 690 929 981 788 614 940
```

c

```
indices_greater_600 <- which(yVec > 600)
indices_greater_600
```

```
##  [1]   2   6  11  14  16  17  22  24  27  29  31  32  33  35  43  44  49  55  60
## [20]  65  67  69  72  73  74  76  77  84  86 100 101 102 103 105 109 110 116 118
## [39] 120 121 122 125 126 132 135 137 139 141 142 145 146 151 152 159 162 165 167
## [58] 168 172 175 177 178 182 185 186 187 188 190 191 199 204 205 208 209 210 211
## [77] 213 218 220 221 223 225 226 227 231 235 237 250
```

d

```
sorted_xVec <- xVec[order(yVec)]
sorted_xVec
```

```
##   [1] 626 380 571  72 478  90 199 625 506 209 191 357 793 884 222 524 778  30
##  [19] 622 554 849 288  19 596 606 739 176  41 970 485  50 877 184  12 416 125
##  [37] 683 144  38 278 325 228 782  63 277 191 216 890 263 373 975 284 734 888
##  [55] 811 759 501  93 155 741 570 592  68 989 568 888   5  37 184 869 752 947
##  [73] 675 777 149 699 591 624 279 546  13   9 654 523 525 692 510  82 934 372
##  [91] 303 298 220 681  96 850 909 517 280 230 702  16 483 145 433 203 531 762
## [109] 984 992 101 445  34 441 454 737 127 939 704 298 538 906 899 246 177 524
## [127] 519 930 289 730 736 880 450 132 958 316 480 816 652 797 525  97 264 944
## [145] 239 535 610 967 625  93 899 189 848 296 726 686 272 694 984 520 740 579
## [163] 226 464 438 490 797 728 529 491 572 737 896 949  16 605 767 630 207 340
## [181] 514 961 948  24   7 146 322 836 868 875 322 507 498 411 620 168 934 618
## [199] 399 904  48 618 837 146 626 450 927 693 996 172 494 275 579 469 458 219
## [217] 311 242 277 804 741 708 844 983 133  26 166 191  55 347 739  30 606 311
## [235] 310 358 898 586 812 656 991  31 821 253 351 311 840 339  42 863
```

e

```
selected_elements <- yVec[seq(1, 250, by = 3)]
selected_elements
```

```
##  [1] 471 114 596 388 275 722 106 610 483 162 754 134 480 114 692 500 870   4 958
## [20] 142 437 345 929 466 999 908 541 521 478  35 331 116 586 960 996 576 853  64
## [39] 278 977 837 124 578  70 548 128 975 923 794 240 718 463 369 211 156  91 500
## [58] 899 638 916 155 290 933 970 342 252 980 145 788 964 808 434 308 830 854 690
## [77] 512  73 788 328 150 327 126 940
```

3

a

```
data(state)
state.x77 <- as_tibble(state.x77, rownames = 'State')
```

```r
low_income_states <- state.x77 %>% filter(Income < 4300)
avg_income_low <- mean(low_income_states$Income)
cat(" 收入低于 4300 的州的平均收入: ", avg_income_low)
```

```
## 收入低于4300的州的平均收入:  3830.6
```

b

```r
highest_income_state <- state.x77 %>%
  arrange(desc(Income)) %>%
  slice(1) %>%
  pull(State)
cat(" 收入最高的州: ", highest_income_state)
```

```
## 收入最高的州:  Alaska
```

c

```r
state.x77 <- state.x77 %>%
  mutate(Population_size = ifelse(Population <= 4500, "S", "L"))
```

d

```r
grouped_stats <- state.x77 %>%
  group_by(Population_size) %>%
  summarize(avg_income = mean(Income), avg_illiteracy = mean(Illiteracy))
grouped_stats
```

```
## # A tibble: 2 x 3
##   Population_size avg_income avg_illiteracy
##   <chr>                <dbl>          <dbl>
## 1 L                    4608.           1.2
## 2 S                    4355.           1.16
```

4

a

```
simulate_uniform <- function(n) {

  tibble(
    X1 = runif(n, min = 0, max = 1),
    X2 = runif(n, min = 0, max = 1)
  )
}
```

b

```
calculate_proportions <- function(observations) {
  if (!all(c("X1", "X2") %in% colnames(observations))) {
    stop(" 输入数据必须包含 X1 和 X2 列")
  }

  n <- nrow(observations)


  dist_to_edges <- pmin(
    observations$X1,
    1 - observations$X1,
    observations$X2,
    1 - observations$X2
  )
  prop_edges <- mean(dist_to_edges < 0.25)


  vertices <- matrix(c(0, 0, 0, 1, 1, 0, 1, 1), ncol = 2, byrow = TRUE)


  dist_to_vertices <- apply(vertices, 1, function(vertex) {
    sqrt(
      (observations$X1 - vertex[1])^2 +
      (observations$X2 - vertex[2])^ 2
```

```
    )
  })


  min_dist_vertices <- apply(dist_to_vertices, 1, min)
  prop_vertices <- mean(min_dist_vertices < 0.25)


  return(list(
    proportion_edges = prop_edges,
    proportion_vertices = prop_vertices
  ))
}



## 验证两个函数
set.seed(123)
obs <- simulate_uniform(10000)

# 计算比例
results <- calculate_proportions(obs)


cat(" 到最近边距离 <0.25 的比例: ", results$proportion_edges, "\n")
```

```
## 到最近边距离<0.25的比例:  0.7493
```

```
cat(" 到最近顶点距离 <0.25 的比例: ", results$proportion_vertices)
```

```
## 到最近顶点距离<0.25的比例:  0.1948
```

5

a

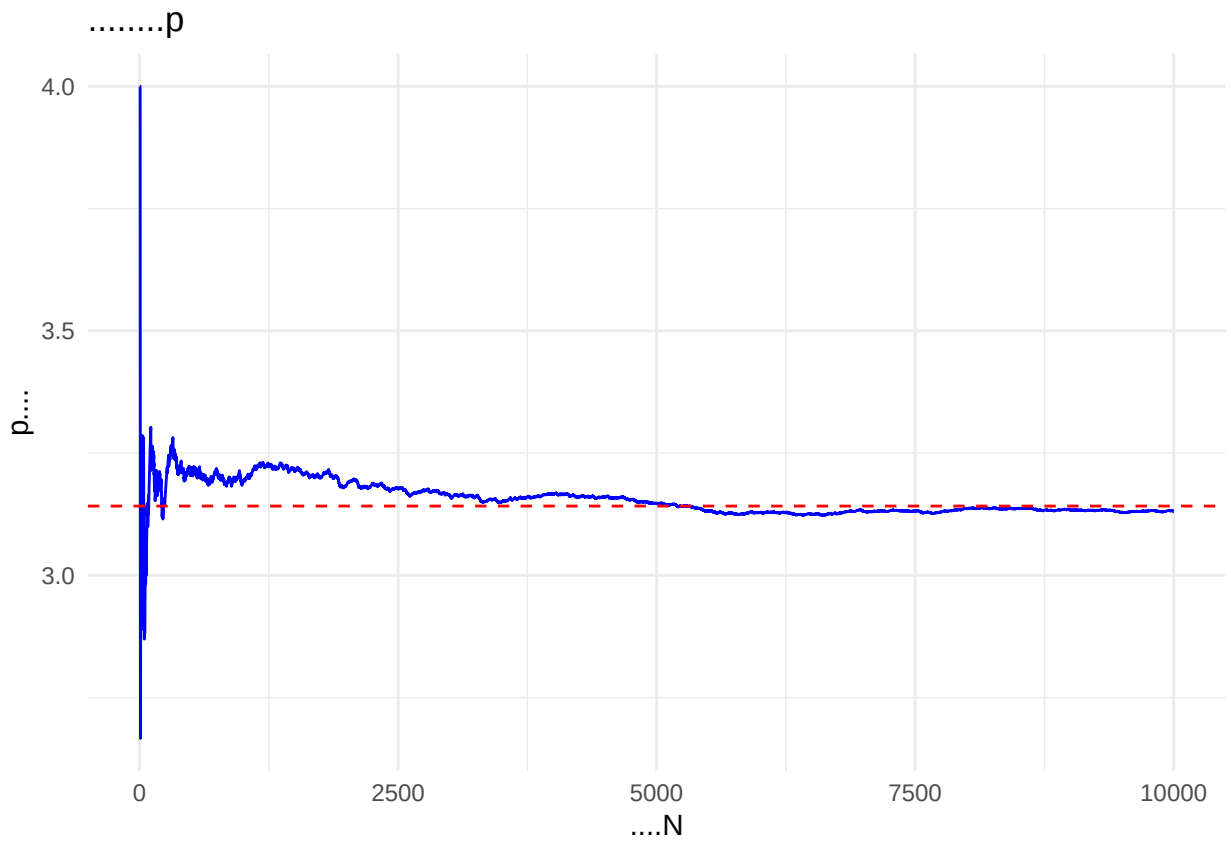
```
n <- 10000
set.seed(1)
```

```r
points <- tibble("x" = runif(n), "y" = runif(n))

points <- points |>
mutate(inside = map2_dbl(.x = x, .y = y, ~ifelse(.x**2 + .y**2 < 1, 1, 0))) |>
rowid_to_column("N")



points <- points |>
  mutate(cumulative_inside = cumsum(inside),
         pi_estimate = 4 * cumulative_inside / N)

# 查看后几行的估计值
tail(points[, c("N", "inside", "cumulative_inside", "pi_estimate")])
```

```
## # A tibble: 6 x 4
##       N inside cumulative_inside pi_estimate
##   <int>  <dbl>             <dbl>       <dbl>
## 1  9995      1              7824        3.13
## 2  9996      0              7824        3.13
## 3  9997      1              7825        3.13
## 4  9998      1              7826        3.13
## 5  9999      1              7827        3.13
## 6 10000      1              7828        3.13
```

b

```r
library(ggplot2)
ggplot(points, aes(x = N, y = pi_estimate)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = pi, color = "red", linetype = "dashed") +
  labs(title = " 蒙特卡洛模拟估计 ",
       x = " 箭头数量 N",
       y = " 的估计值") +
  theme_minimal()
```



6

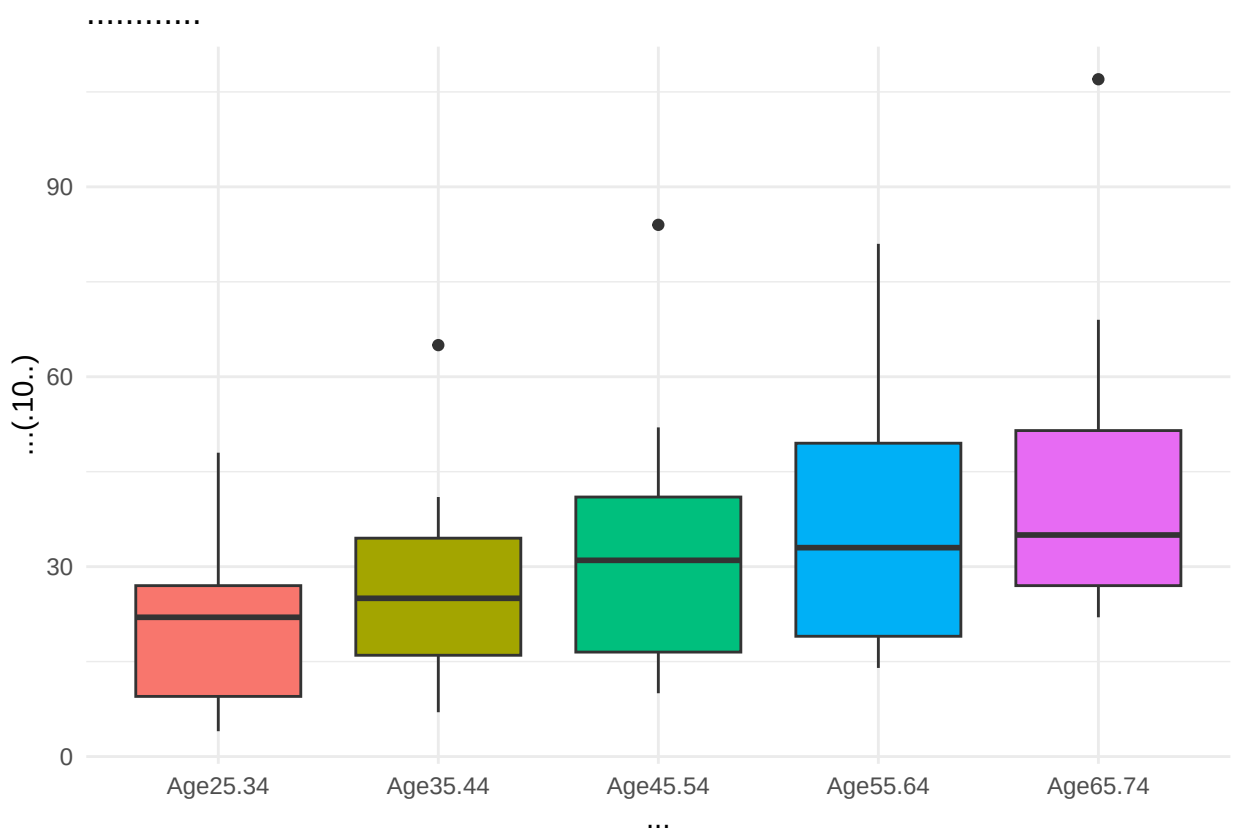
a

```
suicrates <- tibble(
  Country = c('Canada', 'Israel', 'Japan', 'Austria', 'France', 'Germany',
              'Hungary', 'Italy', 'Netherlands', 'Poland', 'Spain', 'Sweden',
              'Switzerland', 'UK', 'USA'),
  Age25.34 = c(22, 9, 22, 29, 16, 28, 48, 7, 8, 26, 4, 28, 22, 10, 20),
  Age35.44 = c(27, 19, 19, 40, 25, 35, 65, 8, 11, 29, 7, 41, 34, 13, 22),
  Age45.54 = c(31, 10, 21, 52, 36, 41, 84, 11, 18, 36, 10, 46, 41, 15, 28),
  Age55.64 = c(34, 14, 31, 53, 47, 49, 81, 18, 20, 32, 16, 51, 50, 17, 33),
  Age65.74 = c(24, 27, 49, 69, 56, 52, 107, 27, 28, 28, 22, 35, 51, 22, 37)
)


library(tidyr)
suicrates_long <- suicrates %>%
```

```
  pivot_longer(cols = -Country,
               names_to = "AgeGroup",
               values_to = "SuicideRate")
```

b

```
library(ggplot2)
ggplot(suicrates_long, aes(x = AgeGroup, y = SuicideRate, fill = AgeGroup)) +
  geom_boxplot() +
  labs(title = " 不同年龄组的自杀率箱线图",
       x = " 年龄组",
       y = " 自杀率（每 10 万人）") +
  theme_minimal() +
  theme(legend.position = "none")
```



从图中可以看出年龄越大，自杀率整体水平越高，且数据波动越大。

65-74 岁组不仅中位数最高，离散程度也最大，需重点关注该年龄段的自杀预防干预；

25-34 岁组数据集中，自杀率相对稳定且偏低。

7

a

```
#data(LaborSupply)
LaborSupply <- read_csv("LaborSupply.csv")
```

```
## Rows: 5320 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (7): lnhr, lnwg, kids, age, disab, id, year
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
labor <- LaborSupply %>%
  mutate(
    hour = exp(lnhr),
    wage = exp(lnwg),
    .before = kids
  ) %>%
  select(-lnhr, -lnwg)

states <- labor %>%
  group_by(year) %>%
  summarise(
    avg_hours = mean(hour),
    sd_hours = sd(hour),
    n = n()
  )

# 输出结果
states
```

```
## # A tibble: 10 x 4
##     year avg_hours sd_hours     n
##    <dbl>     <dbl>    <dbl> <int>
##  1  1979     2202.     502.   532
##  2  1980     2182.     454.   532
```

```
##  3  1981     2185.     460.   532
##  4  1982     2145.     442.   532
##  5  1983     2124.     550.   532
##  6  1984     2149.     492.   532
##  7  1985     2203.     515.   532
##  8  1986     2195.     482.   532
##  9  1987     2219.     529.   532
## 10  1988     2222.     478.   532
```

b

```
age_hours_1982 <- labor %>%
  filter(year == 1982) %>%
  group_by(age) %>%
  summarise(avg_hours = mean(hour, na.rm = TRUE)) %>%
  arrange(desc(avg_hours))


max_age_group <- age_hours_1982$age[1]

cat("1982 年工作时长最长的年龄组为", max_age_group, " 岁，平均工作时长为",
    round(age_hours_1982$avg_hours[1], 2), " 小时")
```

```
## 1982年工作时长最长的年龄组为 46 岁，平均工作时长为 2373.46 小时
```

c

```
id_years <- labor %>%
  group_by(id) %>%
  summarise(
    n_years = n_distinct(year),
    first_year = min(year),
    last_year = max(year)
  )


is_balanced <- (length(unique(id_years$n_years)) == 1)
```

```r
labor <- labor %>%
  left_join(id_years[, c("id", "n_years")], by = "id")

cat(" 面板数据平衡性判断: ", ifelse(is_balanced, " 平衡", " 不平衡"))
```

```
## 面板数据平衡性判断:  平衡
```

d

```r
id_no_kids <- labor %>%
  group_by(id) %>%
  summarise(
    no_kids = as.integer(all(kids == 0))  # 1= 全程无子女, 0= 有子女
  )


labor <- labor %>%
  left_join(id_no_kids, by = "id")

prop_no_kids <- mean(id_no_kids$no_kids)
cat(" 全程无子女的个体占比: ", round(prop_no_kids * 100, 2), "%")
```

```
## 全程无子女的个体占比:  8.08 %
```

e

```r
wage_compare_1980 <- labor %>%
  filter(year == 1980) %>%
  group_by(no_kids) %>%
  summarise(
    avg_wage = mean(wage, na.rm = TRUE),
    sd_wage = sd(wage, na.rm = TRUE),
    count = n()
  )


wage_compare_1980
```

```
## # A tibble: 2 x 4
##   no_kids avg_wage sd_wage count
##     <int>    <dbl>   <dbl> <int>
## 1       0     14.5    6.69   489
## 2       1     15.9    6.71    43
```