Solutions for Final2024

Lai Wei

2024-07-08

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

1.

```
A <- matrix(c(9, 5, 2, 9, 4, 0, 6, 2, 12, 7, 8, 9, 2, 9, 0, 11), nrow = 4)
b <- c(7, 18, 1, 0)
solution <- solve(A, b)
solution
```

```
## [1] -4.2028571 -6.2285714  5.8471429 -0.2128571
```

2.

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

a. Create the vector

```
zVec <- yVec[seq(2, 250, by = 2)] - xVec[seq(1, 249, by = 2)]
zVec
```

```
##   [1]  481  -20 -657  507 -968  560  316  -61  194  -72  -72 -827  126  727 -740
##  [16] -165  459  237 -578 -209  -59 -199  479 -187  287  651  638  737  514  844
##  [31]  245 -668 -634   95  -47 -368   41 -556 -583  -67 -260 -147 -435   73  408
##  [46]  403  602  -69 -330  681  113  584 -301 -532 -454 -161   51 -469 -529  349
##  [61] -247 -201  587 -166  214  -25   75 -141  603  613 -496 -661  102  300 -374
##  [76] -444 -292   71 -218  122 -282  -21   65 -150 -650 -740  286 -607 -483  304
##  [91]  228 -332 -461  450  389 -597 -730  128 -887 -264  298 -332 -124  690  -20
## [106]  -43  113 -314 -361 -455  394 -205  434 -655 -499 -260  109  298    9   30
## [121]  335   37  471 -683 -333
```

b. Pick out the values in yVec which are > 600

```
yVec_greater_than_600 <- yVec[yVec > 600]
yVec_greater_than_600
```

```
##  [1] 961 839 696 677 853 835 706 827 968 768 901 741 617 931 753 832 931 679 713
## [20] 806 993 704 927 975 875 632 881 719 773 623 671 607 632 666 815 676 824 932
## [39] 723 923 678 943 901 758 632 630 787 835 966 691 632 826 791 824 714 722 715
## [58] 918 820 682 669 809 999 973 868 923 980 774 937 760 674 906 730 865 867 628
## [77] 939 881 884 724 726 656 886 813 821 612 970 804 800 607 705 606 810
```

c. Index positions in yVec of the values which are > 600

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

```
##  [1]   5   7   8  11  19  20  22  25  26  28  33  34  35  39  40  41  46  51  52
## [20]  54  56  57  59  60  61  63  65  67  68  71  75  82  83  85  88  89  92  93
## [39]  94  97  99 100 101 103 104 106 109 112 115 126 132 133 138 139 140 141 148
## [58] 164 166 168 169 173 174 177 179 180 185 187 188 189 190 193 202 205 208 210
## [77] 212 213 214 215 217 219 221 222 225 233 235 237 239 240 246 247 249
```

d. Sort the numbers in xVec in the order of increasing values in yVec

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

```
##   [1] 634 162 935 628 988 679 712 188 991 614 144 488 450 966 177 528 863 657
##  [19] 187 363 236 787 507  67 808 452 772 389 957 104 950 373 939 755 617 996
##  [37] 203 454 106 340 991 149 196  35  78 724 927 648 272 849 551 173 928 830
##  [55] 367 946 344  47 224 586 127 567 234  12 251  23  97 179 786 411  81 829
##  [73] 454 756  82 973 452 832 877  67 570 695  27 392 158 473 742 481 588 941
##  [91] 227 974 321 883 481 833 393 985 437 757 764 295 255 989 999 204 662 509
## [109] 415 660 405 192 620  90 972 553 585 168 421 246 416 501 976 726 867 303
## [127] 814 483 135 778 356 775 140 256 267 855 982 168 446 990 931 376 961  41
## [145]  82 642 598 884 722 876 399 432 696 903  25 387 631 746 749 589 314 243
## [163] 437 412 171 922 546 420  79 932 441 786 889 447  25   8 262  62 492  44
## [181] 816   5 608 899 924  15 807 678 686 777 772 934 433 807 631  48 285 258
## [199] 147 487 590 795 577  25 331 713 537 530  23 371  12 274 101 199 842 576
## [217] 132 126 189 907 132 325 619 226 884 771 508 419 282 151 829 670 548 203
## [235] 131 962 266 121 181 195 419 829 980 161   6 830 774 979 578 909
```

e. Elements in yVec at index positions 1, 4, 7, …

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

```
##  [1] 591 153 839  31 533 376 853 706 827 768 401 741 178 753  51 931 224 713 517
## [20] 519 875 254 719 587 439 333 493 607 666 815 456 723 923 943 758 630 787 835
## [39] 966 460 579 387 284 249 826 114 824 190 185 715 463 189  86 595 138 820 669
## [58] 250  13 347 244  60 774 674 906 210  89 730 865 867 529 884 726 477  69 446
## [77]  22 570 970  34 145 413 606 204
```

3.

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

a. Select states with income < 4300 and calculate average income

```
states_low_income <- state.x77 %>% filter(Income < 4300)
average_income <- mean(states_low_income$Income)
average_income
```

```
## [1] 3830.6
```

b. Sort by income and select the state with the highest income

```
state_highest_income <- state.x77 %>% arrange(desc(Income)) %>% slice(1)
state_highest_income
```

```
## # A tibble: 1 × 9
##   State  Population Income Illiteracy `Life Exp` Murder `HS Grad` Frost   Area
##   <chr>       <dbl>  <dbl>      <dbl>      <dbl>  <dbl>     <dbl> <dbl>  <dbl>
## 1 Alaska        365   6315        1.5       69.3   11.3      66.7   152 566432
```

c. Add a variable categorizing the size of population

```
state.x77 <- state.x77 %>%
  mutate(Population_Size = ifelse(Income <= 4500, "S", "L"))
```

d. Average income and illiteracy by population size

```
average_values <- state.x77 %>%
  group_by(Population_Size) %>%
  summarise(
    avg_income = mean(Income),
    avg_illiteracy = mean(Illiteracy)
  )
average_values
```

```
## # A tibble: 2 × 3
##   Population_Size avg_income avg_illiteracy
##   <chr>                <dbl>          <dbl>
## 1 L                    4902.          0.981
## 2 S                    3931.          1.38
```

4.

a. Function to simulate n observations from uniform distribution

```
simulate_uniform <- function(n) {
  x <- runif(n, 0, 1)
  y <- runif(n, 0, 1)
  tibble(x = x, y = y)
}
```

b. Function to calculate proportions

```
calculate_proportions <- function(data) {
  dist_to_edge <- pmin(data$x, 1 - data$x, data$y, 1 - data$y)
  dist_to_vertex <- pmin(
    sqrt(data$x^2 + data$y^2),
    sqrt((1 - data$x)^2 + data$y^2),
    sqrt(data$x^2 + (1 - data$y)^2),
    sqrt((1 - data$x)^2 + (1 - data$y)^2)
  )
  prop_edge <- mean(dist_to_edge < 0.25)
  prop_vertex <- mean(dist_to_vertex < 0.25)
  list(prop_edge = prop_edge, prop_vertex = prop_vertex)
}

# Test with n = 1000
set.seed(37)
data <- simulate_uniform(1000)
calculate_proportions(data)
```

```
## $prop_edge
## [1] 0.756
##
## $prop_vertex
## [1] 0.207
```

5.

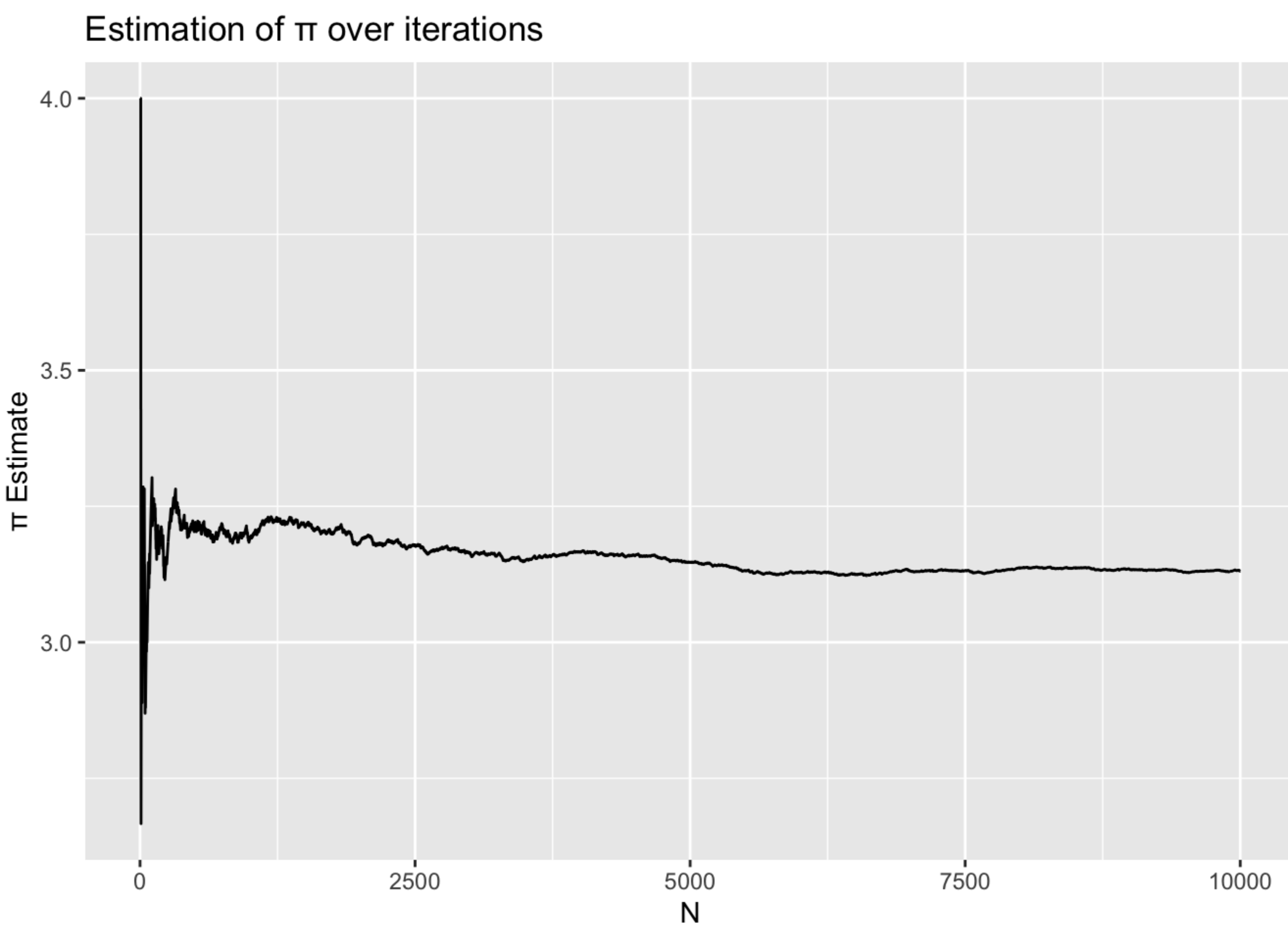
```
n <- 10000
set.seed(1)
points <- tibble(x = runif(n), y = runif(n))
points <- points %>%
  mutate(inside = map2_dbl(.x = x, .y = y, ~ifelse(.x^2 + .y^2 < 1, 1, 0))) %>%
  rowid_to_column("N")
```

a. Compute the estimation of π at each row

```
points <- points %>%
  mutate(pi_estimate = 4 * cumsum(inside) / N)
```

b. Plot the estimates of π against N

```
ggplot(points, aes(x = N, y = pi_estimate)) +
  geom_line() +
  labs(title = "Estimation of π over iterations", x = "N", y = "π Estimate")
```



6.

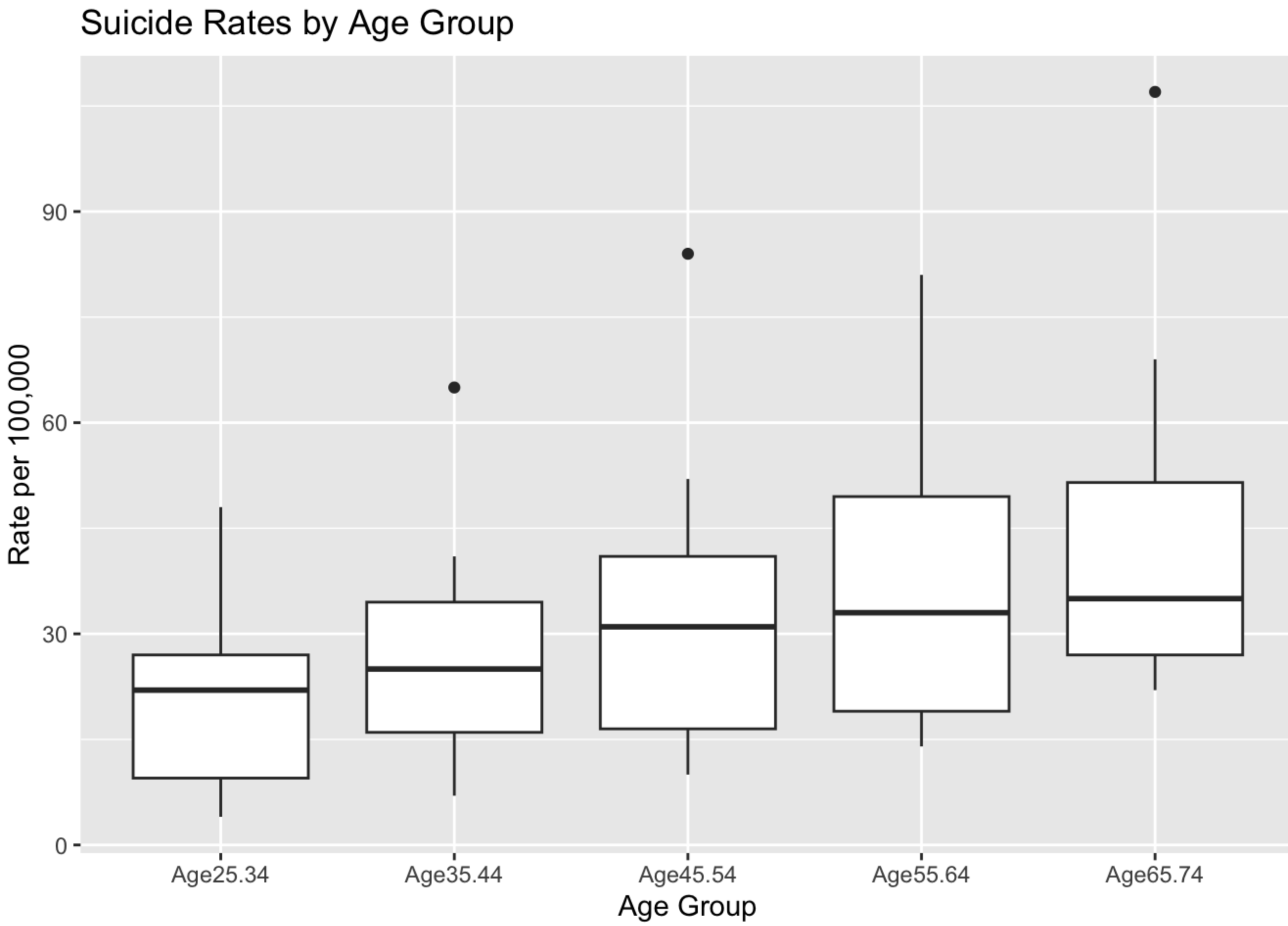
```
suicrates <- tibble(
  Country = c('Canada', 'Israel', 'Japan', 'Austria', 'France', 'Germany', 'Hungary', 'Italy', 'Ne
therlands', 'Poland', 'Spain', 'Sweden', 'Switzerland', 'UK', 'USA'),
  Age25.34 = c(22, 9, 22, 29, 16, 28, 48, 7, 8, 26, 4, 28, 22, 10, 20),
  Age35.44 = c(27, 19, 19, 40, 25, 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)
)
```

a. Transform suicrates into long form

```
suicrates_long <- suicrates %>%
  pivot_longer(cols = starts_with("Age"), names_to = "AgeGroup", values_to = "Rate")
```

b. Construct side-by-side box plots

```
ggplot(suicrates_long, aes(x = AgeGroup, y = Rate)) +
  geom_boxplot() +
  labs(title = "Suicide Rates by Age Group", x = "Age Group", y = "Rate per 100,000")
```



The suicide rate in the age range from 25.34 to 65.74 increases with age, and the difference in suicide rates in different countries increases with age.

7.

```
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.
```

```
# Create hour and wage variables
labor <- LaborSupply %>%
  mutate(hour = exp(lnhr), wage = exp(lnwg), .before = kids) %>%
  dplyr::select(-lnhr, -lnwg)
```

a. Compute average annual hours worked and their standard deviations by year

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

```
## # A tibble: 10 × 3
##     year avg_hours sd_hours
##    <dbl>     <dbl>    <dbl>
##  1  1979     2202.     502.
##  2  1980     2182.     454.
##  3  1981     2185.     460.
##  4  1982     2145.     442.
##  5  1983     2124.     550.
##  6  1984     2149.     492.
##  7  1985     2203.     515.
##  8  1986     2195.     482.
##  9  1987     2219.     529.
## 10  1988     2222.     478.
```

b. Age group that worked the most hours in 1982

```
most_hours_1982 <- labor %>%
  filter(year == 1982) %>%
  group_by(age) %>%
  summarise(avg_hours = mean(hour)) %>%
  arrange(desc(avg_hours)) %>%
  slice(1)
most_hours_1982
```

```
## # A tibble: 1 × 2
##     age avg_hours
##   <dbl>     <dbl>
## 1    46     2373.
```

c. Create n_years variable

```r
labor <- labor %>%
  group_by(id) %>%
  mutate(n_years = n_distinct(year)) %>%
  ungroup()

# Is the panel balanced?
is_balanced <- all(labor %>% count(id) %>% pull(n) == labor$n_years)
is_balanced
```

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

d. Individuals without any kids during the whole period

```r
labor <- labor %>%
  group_by(id) %>%
  mutate(no_kids = ifelse(all(kids == 0), 1, 0)) %>%
  ungroup()
```

e. Average wage, standard deviation, and number of observations for no kids vs kids in 1980

```r
stats_1980 <- labor %>%
  filter(year == 1980) %>%
  group_by(no_kids) %>%
  summarise(
    avg_wage = mean(wage),
    sd_wage = sd(wage),
    n = n()
  )
stats_1980
```

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