Mathematical practice final exam 2025

Zhang Zile,3230104237

2025-07-02

1.

Find the inverse of the following matrix and verify it using the `all.equal()` function.

$$\begin{pmatrix} 9 & 4 & 12 & 2 \\ 5 & 0 & 7 & 9 \\ 2 & 6 & 8 & 0 \\ 9 & 2 & 9 & 11 \end{pmatrix}$$

```r
A <- matrix(c(9, 4, 12, 2,
              5, 0, 7, 9,
              2, 6, 8, 0,
              9, 2, 9, 11),
nrow = 4, byrow = TRUE)
A_inv <- solve(A) #A^-1
product_AA_inv <- A %*% A_inv #A*A^-1
product_A_inv_A <- A_inv %*% A #A^-1*A
is_identity_AA_inv <- all.equal(product_AA_inv, diag(nrow(A)))
is_identity_A_inv_A <- all.equal(product_A_inv_A, diag(ncol(A)))
print(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    9    4   12    2
## [2,]    5    0    7    9
## [3,]    2    6    8    0
## [4,]    9    2    9   11
```

```r
print(A_inv)
```

```
##             [,1]  [,2]        [,3]        [,4]
## [1,]  0.08571429 -0.26 -0.12285714  0.19714286
## [2,] -0.14285714 -0.30  0.17142857  0.27142857
## [3,]  0.08571429  0.29  0.02714286 -0.25285714
## [4,] -0.11428571  0.03  0.04714286  0.08714286
```

```r
print(is_identity_AA_inv)
```

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

```r
print(is_identity_A_inv_A)
```

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

2.

Execute the following lines which create two vectors of random integers which are chosen with replacement from the integers $0, 1, \ldots, 999$. Both vectors have length 250.

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

(a) Create the vector $(y_2 - x_1, \cdots, y_n - x_{n-1})$.

```r
vec <- yVec[2:length(yVec)]-xVec[1:((length(xVec))-1)]
print(vec)
```

```
##   [1]  148 -108    9  239  -61 -458   86 -585  140 -306  285   82   77  -63 -777
##  [16]  462  484  283  245  -44  378  -16  -51 -302 -357 -792   56  744  292  282
##  [31] -992 -799  123  613 -163  -81  110  261  860  565 -128 -760  845  288  827
##  [46] -254 -479 -383  309 -645  -10  220  119  237 -541 -108   26  599  313  469
##  [61]  195  638   91 -318  329 -165   96 -553 -692  650  407 -211 -493  -24 -941
##  [76]   69  350   64 -232  539  326 -809 -433 -712 -470  -16  290 -473 -302  956
##  [91] -543 -624 -591 -244   81   74    2  287  -47 -597  152  -33  746  105  196
## [106] -751 -779  225 -648   54  -11  731   79 -446 -280  -54  617  565  577 -358
## [121] -112  -72 -385  372   89  773 -168 -485  164 -436  366  509  248  359  221
## [136] -699 -221  -71 -245  610   44  120  -72 -308 -363  337  156  -90   39 -419
## [151]  342   -6  135  477 -362  674   45 -342  143  531 -279   88 -599 -837  278
## [166] -166  -48   20  131 -118  665  779 -462 -170 -856   68   49  201 -551 -812
## [181]  191 -495 -214 -146  425 -335  454   38   35 -237 -217  -13  -40  293  484
## [196]  186 -164  287  225  519   35  509  475 -632 -269  599  524 -825 -386   46
## [211] -578   82  -99  819  135   34 -235   56 -259  268  446  169 -466  571  454
## [226] -868  213  355 -378  439 -206 -710  710 -400  325  -33  -66  -77   64 -238
## [241]  -78 -565  229 -277 -317 -112  151  163 -152
```

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

```r
vec_b <- yVec[yVec > 600]
print(vec_b)
```

```
##  [1] 951 782 747 738 666 941 833 914 657 709 938 902 876 756 882 873 877 698
## [19] 975 957 719 653 684 606 913 772 640 630 696 994 963 762 930 878 744 888
## [37] 779 952 630 685 995 640 873 778 907 888 935 676 837 643 945 862 773 663
## [55] 758 784 899 801 734 831 723 850 680 602 755 670 862 785 831 739 860 803
## [73] 656 833 838 838 752 911 806 684 762 875 903 910 673 822 999 939 947 857
## [91] 950 936 967 665 886 916 664 966 633 879 854 874 825 634 867 638 735 917
```

(c) What are the index positions in yVec of the values which are > 600?

```r
which(yVec>600)
```

```
##   [1]   2  14  17  18  23  24  29  30  34  35  39  40  41  42  44  45  46  50
##  [19]  52  53  54  57  59  61  62  63  64  66  68  71  72  73  77  79  80  81
##  [37]  82  87  88  90  91  95  96  98  99 100 104 105 106 111 113 117 118 120
##  [55] 122 125 127 132 133 134 140 141 144 147 148 150 153 154 157 160 161 166
##  [73] 167 170 172 173 175 177 178 179 186 189 190 193 194 195 196 197 200 201
##  [91] 203 204 207 208 215 216 220 221 222 225 226 231 234 236 239 240 247 249
```

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

```r
vec_d <- xVec[order(yVec)]
print(vec_d)
```

```
##   [1] 599 492 987 115 837  66 949  47 987 861 412 173 309 185 623 832 285 639
##  [19] 961 843 301 525 425 985 702 968 916  14 761 236 922 159 776   7 511 589
##  [37] 491 212 337  37 558 943 212  66 323 913 632 389 923 274 534 984 350 596
##  [55] 811 944 705 330 870 266 126 269 344 980  67 847 368 884 435 754 235 837
##  [73] 927  89 189  61 631 620 137 430 872 722 856 790 916 385 157 489 293 999
##  [91] 211 979 917 752 540  85 308 123 297 923 788 335  41 803 441 892 190 677
## [109] 867 106 616 968 837   9 121 721 188 481 551 214 784 995 337 929 435 114
## [127] 265 868 433 184 814 804 637 682 772 501 664  86 714 625 400 600 599 718
## [145] 762 511 282 250 694 386 792 532 372 215  96 567 698 846 992 578 529 641
## [163] 400   4 709  39 934 727 163 329 240 583 106 262 329 349 254 922 405 913
## [181] 632 638 427 549  23 620 871 328 130 113 225 822 483 515 452  93 345 622
## [199] 169 808  59 563 179 898 389 784 156 650 574  50  30 687 868 884 428 976
## [217] 400 585 781 453 948 502 311 831 935 713 757 134  89 285 338 214 571 770
## [235]  42 581 643 494 338 461 581 340 600 973 187 141 737 556 603 753
```

(e) Pick out the elements in yVec at index positions $1, 4, 7, 10, 13, \cdots$

```r
vec_e <- yVec[seq(1, length(yVec), by = 3)]
print(vec_e)
```

```
##  [1] 421  18 141 565 203  60 545 568 341 292 371 657 411 902 153 877 175 975 566
## [20] 398 606 640 597 225 762  43 878 779 283 630 995 246 104 888 304 837 410 521
## [39]  55 773 209  48 899 375 734 327  88 223  92 755 159 785 831 739 569 803  86
## [58] 838 752 806 131 406  92 903 910 999 328 424 138 665 230 231 319 664 451 854
## [77] 590 481  52 200 456 522 735 186
```

3.

For this problem we'll use the (built-in) dataset state.x77.

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

a. Select all the states having an income less than 4300, and calculate the average income of these states.

```r
a <- state.x77 |> filter(Income < 4300)
mean(pull(a, Income))
```

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

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

```r
state_sort<-state.x77 |> arrange(desc(Income))
# print(state_sort)
state_sort |> head(1)
```

```
## # A tibble: 1 x 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 to the data frame which categorizes the size of population: $<= 4500$ is `S`, > 4500 is `L`.

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

d. Find out the average income and illiteracy of the two groups of states, distinguishing by whether the states are small or large.

```r
d_summary <- state.x77 %>%
  group_by(Size) %>%
  summarise(
    avg_income = mean(Income),
    avg_Illiteracy = mean(Illiteracy)
  )
```

4.

a. Write a function to simulate `n` observations of (X_1, X_2) which follow the uniform distribution over the square $[0,1] \times [0,1]$.

```r
simulate <- function(n) {
  data.frame(
    X1 = runif(n, 0, 1),
    X2 = runif(n, 0, 1)
  )
}
```

b. Write a function to calculate the proportion of the observations that the distance between (X_1, X_2) and the nearest edge is less than 0.25, and the proportion of them with the distance to the nearest vertex less than 0.25.

```r
calculate_proportions <- function(data) {
  d_edge <- pmin(data$X1, 1 - data$X1, data$X2, 1 - data$X2)
  prop_edge <- mean(d_edge < 0.25)

  d1 <- sqrt(data$X1^2 + data$X2^2)                    # to (0, 0)
  d2 <- sqrt(data$X1^2 + (1 - data$X2)^2)              # to (0, 1)
  d3 <- sqrt((1 - data$X1)^2 + data$X2^2)              # to (1, 0)
  d4 <- sqrt((1 - data$X1)^2 + (1 - data$X2)^2)        # to (1, 1)
  d_vertex <- pmin(d1, d2, d3, d4)
  prop_vertex <- mean(d_vertex < 0.25)

  c(proportion_edge = prop_edge,
    proportion_vertex = prop_vertex)
}
```

5.

To estimate π with a Monte Carlo simulation, we draw the unit circle inside the unit square, the ratio of the area of the circle to the area of the square will be $\pi/4$. Then shot K arrows at the square, roughly $K * \pi/4$ should have fallen inside the circle. So if now you shoot N arrows at the square, and M fall inside the circle, you have the following relationship $M = N * \pi/4$. You can thus compute π like so: $\pi = 4 * M/N$. The more arrows N you throw at the square, the better approximation of π you'll have.

```r
n <- 10000

set.seed(1)
points <- tibble("x" = runif(n), "y" = runif(n))
```

Now, to know if a point is inside the unit circle, we need to check whether $x^2 + y^2 < 1$. Let's add a new column to the points tibble, called `inside` equal to `1` if the point is inside the unit circle and `0` if not:

```r
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, by computing the cumulative sum of the 1's in the `inside` column and dividing that by the current value of `N` column:

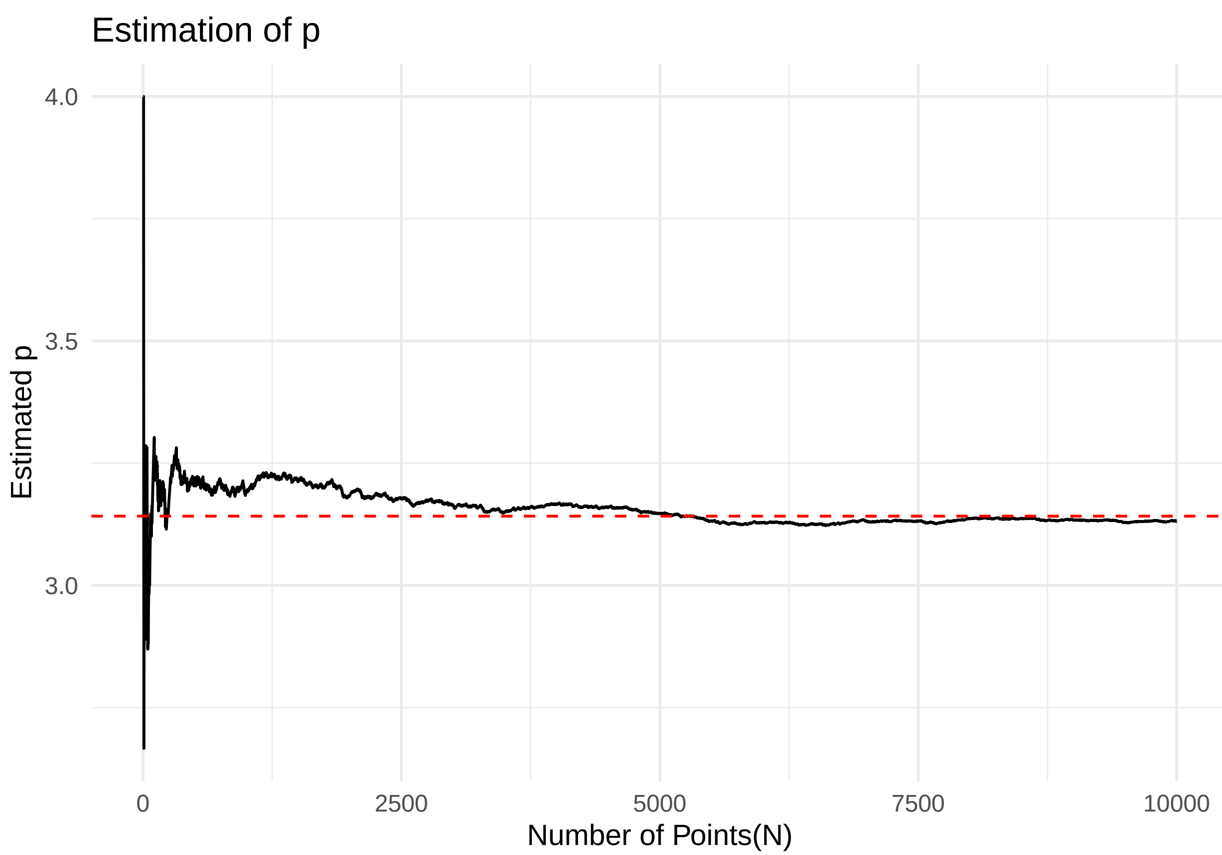
```r
library(tidyverse)
points_with_pi <- points %>%
  mutate(
    cum_inside = cumsum(inside),
    pi_estimate = 4 * cum_inside / N
  )
tail(points_with_pi)
```

```
## # A tibble: 6 x 6
##       N     x     y inside cum_inside pi_estimate
##   <int> <dbl> <dbl>  <dbl>      <dbl>       <dbl>
## 1  9995 0.244 0.691      1       7824        3.13
## 2  9996 0.732 0.788      0       7824        3.13
```

```
## 3  9997 0.499 0.814       1        7825         3.13
## 4  9998 0.503 0.478       1        7826         3.13
## 5  9999 0.568 0.602       1        7827         3.13
## 6 10000 0.653 0.111       1        7828         3.13
```

b. Plot the estimates of π against `N`.

```r
ggplot(points_with_pi, aes(x = N, y = pi_estimate)) +
  geom_line() +
  geom_hline(yintercept = pi, color = "red", linetype = "dashed") +
  labs(title = "Estimation of  ",
       x = "Number of Points(N)",
       y = "Estimated  ") +
  theme_minimal()
```



6.

Mortality rates per 100,000 from male suicides for a number of age groups and a number of countries are given in the following data frame.

```r
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),
```

```r
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))
```

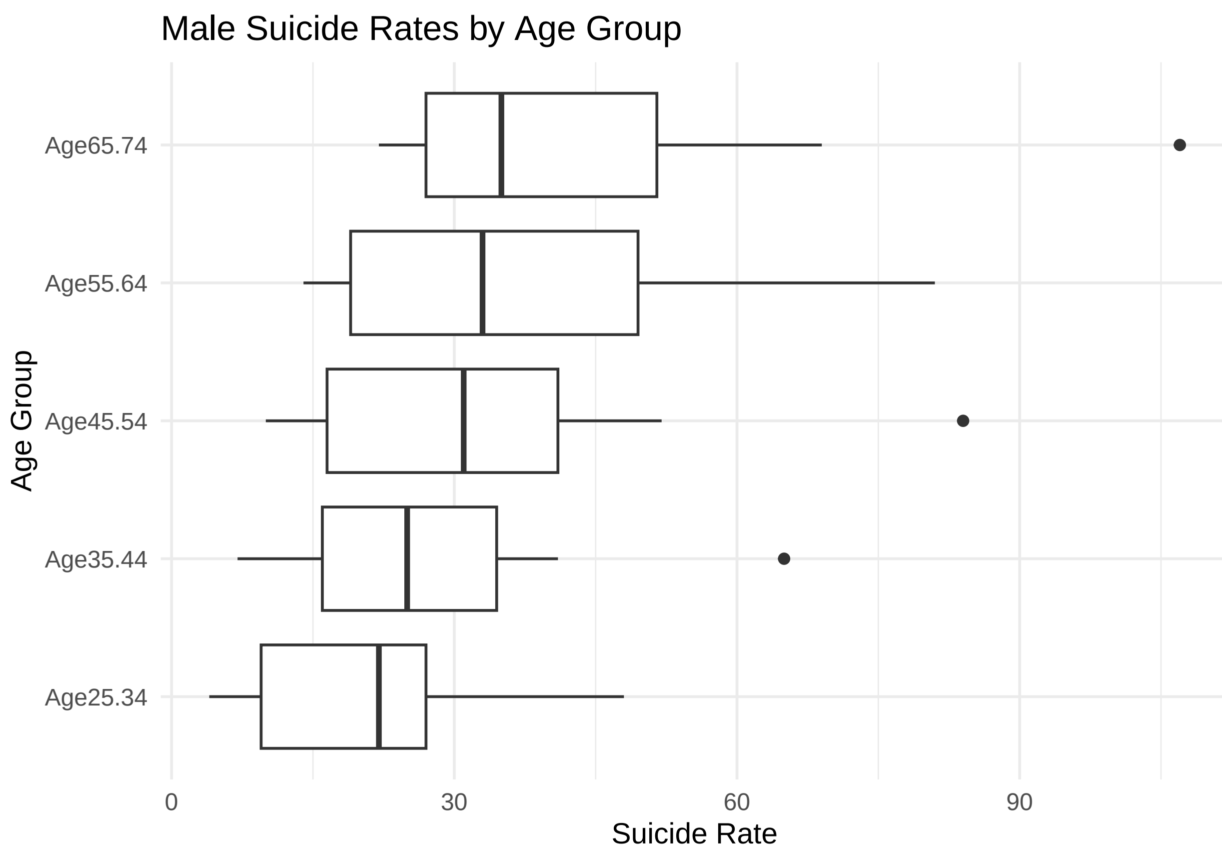
a. Transform `suicrates` into *long* form.

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

```
## # A tibble: 6 x 3
##   Country AgeGroup  Rate
##   <chr>   <chr>    <dbl>
## 1 Canada  Age25.34    22
## 2 Canada  Age35.44    27
## 3 Canada  Age45.54    31
## 4 Canada  Age55.64    34
## 5 Canada  Age65.74    24
## 6 Israel  Age25.34     9
```

b. Construct side-by-side box plots for the data from different age groups, and comment on what the graphic tells us about the data.

```r
ggplot(suicrates_long, aes(x = AgeGroup, y = Rate)) +
  geom_boxplot() +
  labs(title = "Male Suicide Rates by Age Group",
       x = "Age Group",
       y = "Suicide Rate") +
  theme_minimal() +
  coord_flip()
```



1.Suicide rates tend to increase with age.

2.There is considerable variability in suicide rates across countries, particularly in the middle-aged groups, as indicated by the wide interquartile ranges.

3.Some age groups also show outliers, indicating countries with exceptionally high suicide rates.

7.

Load the `LaborSupply` dataset from the `{Ecdat}` package and answer the following questions:

```
#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.
```

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

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

```
a_summary <- labor %>%
  group_by(year) %>%
  summarise(
    avg_hour = mean(hour, na.rm = TRUE),
    sd_hour = sd(hour, na.rm = TRUE)
  )
print(a_summary)
```

```
## # A tibble: 10 x 3
##     year avg_hour sd_hour
##    <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. What age group worked the most hours in the year 1982?

```
b_result <- labor %>%
  filter(year == 1982) %>%
  mutate(age_group = case_when(
    age >= 20 & age < 30 ~ "20-29",
    age >= 30 & age < 40 ~ "30-39",
    age >= 40 & age < 50 ~ "40-49",
    age >= 50 & age < 60 ~ "50-59",
    TRUE ~ "60+"
  )) %>%
  group_by(age_group) %>%
  summarise(total_hour = sum(hour, na.rm = TRUE)) %>%
  slice_max(total_hour, n = 1)
print(b_result)
```

```
## # A tibble: 1 x 2
##   age_group total_hour
##   <chr>          <dbl>
## 1 30-39        545697.
```

c. Create a variable, `n_years` that equals the number of years an individual stays in the panel. Is the panel balanced?

```
c_data <- labor %>%
  group_by(id) %>%
  mutate(n_years = n()) %>%
  ungroup()
```

```r
is_balanced <- n_distinct(c_data$n_years) == 1
cat(is_balanced)
```

```
## TRUE
```

d. Which are the individuals that do not have any kids during the whole period? Create a variable, `no_kids`, that flags these individuals (1 = no kids, 0 = kids)

```r
d_data <- c_data %>%
  group_by(id) %>%
  mutate(no_kids = as.integer(all(kids == 0))) %>%
  ungroup()
no_kids_ids <- d_data %>%
  filter(no_kids == 1) %>%
  distinct(id) %>%
  pull(id)
cat(paste(no_kids_ids, collapse = ", "))
```

```
## 3, 10, 20, 30, 36, 52, 53, 63, 71, 94, 109, 135, 149, 150, 154, 155, 161, 175, 200, 202, 214, 224, 2
```

e. Using the `no_kids` variable from before compute the average wage, standard deviation and number of observations in each group for the year 1980 (no kids group vs kids group).

```r
e_result <- d_data %>%
  filter(year == 1980) %>%
  group_by(no_kids) %>%
  summarise(
    avg_wage = mean(wage),
    sd_wage = sd(wage),
    n_obs = n()
  )
print(e_result)
```

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