Plumbing 2 - Solutions

Setup

```
library(tidyverse)
set.seed(92815)

gradebook <- tibble(
  student_id = c(192297, 291857, 500286, 449192, 372152, 627561),
  name = c('Alice', 'Bob', 'Charlotte', 'Dante',
           'Ethelburga', 'Felix'),
  quiz1 = round(rnorm(6, 65, 15)), quiz2 = round(rnorm(6, 88, 5)),
  quiz3 = round(rnorm(6, 75, 10)), midterm1 = round(rnorm(6, 75, 10)),
  midterm2 = round(rnorm(6, 80, 8)), final = round(rnorm(6, 78, 11)))

gradebook
```

```
# A tibble: 6 × 8
  student_id name       quiz1 quiz2 quiz3 midterm1 midterm2 final
       <dbl> <chr>      <dbl> <dbl> <dbl>    <dbl>    <dbl> <dbl>
1     192297 Alice         64    96    68       81       90    99
2     291857 Bob           58    91    91       75       75    79
3     500286 Charlotte     70    94    71       81       70    74
4     449192 Dante         57    85    84       83       94    83
5     372152 Ethelburga    74    91    70       63       73    96
6     627561 Felix         77    86    68       78       83    75
```

```
emails <- tibble(
  student_id = c(101198, 192297, 372152, 918276, 291857),
  email = c('unclejoe@whitehouse.gov', 'alice.liddell@chch.ox.ac.uk',
            'ethelburga@lyminge.org', 'mzuckerberg@gmail.com',
            'microsoftbob@hotmail.com'))
emails
```

```
# A tibble: 5 × 2
  student_id email
       <dbl> <chr>
1     101198 unclejoe@whitehouse.gov
2     192297 alice.liddell@chch.ox.ac.uk
3     372152 ethelburga@lyminge.org
4     918276 mzuckerberg@gmail.com
5     291857 microsoftbob@hotmail.com
```

```
quiz_scores <- gradebook |>
  pivot_longer(starts_with('quiz'),
               names_to = 'quiz',
               names_prefix = 'quiz',
               names_transform = list(quiz = as.numeric),
               values_to = 'score') |>
  select(student_id, name, quiz, score)
```

```
2     291857 Bob           58    91    91       75       75    79 microsoftbob@…
3     372152 Ethelburga    74    91    70       63       73    96 ethelburga@ly…
4     101198 <NA>          NA    NA    NA       NA       NA    NA unclejoe@whit…
5     918276 <NA>          NA    NA    NA       NA       NA    NA mzuckerberg@g…
```

```
# Part 2
# The result contains everyone whose id appears in *either* dataset. This
# requires lots of padding out with missing values.
full_join(gradebook, emails)
```

```
Joining with `by = join_by(student_id)`

# A tibble: 8 × 9
  student_id name       quiz1 quiz2 quiz3 midterm1 midterm2 final email
       <dbl> <chr>      <dbl> <dbl> <dbl>    <dbl>    <dbl> <dbl> <chr>
1     192297 Alice         64    96    68       81       90    99 alice.liddell…
2     291857 Bob           58    91    91       75       75    79 microsoftbob@…
3     500286 Charlotte     70    94    71       81       70    74 <NA>
4     449192 Dante         57    85    84       83       94    83 <NA>
5     372152 Ethelburga    74    91    70       63       73    96 ethelburga@ly…
6     627561 Felix         77    86    68       78       83    75 <NA>
7     101198 <NA>          NA    NA    NA       NA       NA    NA unclejoe@whit…
8     918276 <NA>          NA    NA    NA       NA       NA    NA mzuckerberg@g…
```

```
# Part 3
# The result contains only those whose id appears in *both* datasets. Everyone
# else is dropped.
inner_join(gradebook, emails)
```

```
Joining with `by = join_by(student_id)`

# A tibble: 3 × 9
  student_id name       quiz1 quiz2 quiz3 midterm1 midterm2 final email
       <dbl> <chr>      <dbl> <dbl> <dbl>    <dbl>    <dbl> <dbl> <chr>
1     192297 Alice         64    96    68       81       90    99 alice.liddell…
2     291857 Bob           58    91    91       75       75    79 microsoftbob@…
3     372152 Ethelburga    74    91    70       63       73    96 ethelburga@ly…
```

```
# Part 4
gradebook |>
  left_join(emails)
```

```
Joining with `by = join_by(student_id)`

# A tibble: 6 × 9
  student_id name       quiz1 quiz2 quiz3 midterm1 midterm2 final email
       <dbl> <chr>      <dbl> <dbl> <dbl>    <dbl>    <dbl> <dbl> <chr>
1     192297 Alice         64    96    68       81       90    99 alice.liddell…
2     291857 Bob           58    91    91       75       75    79 microsoftbob@…
3     500286 Charlotte     70    94    71       81       70    74 <NA>
4     449192 Dante         57    85    84       83       94    83 <NA>
```

```
quiz_scores
```

```
# A tibble: 18 × 4
   student_id name        quiz score
        <dbl> <chr>      <dbl> <dbl>
 1     192297 Alice          1    64
 2     192297 Alice          2    96
 3     192297 Alice          3    68
 4     291857 Bob            1    58
 5     291857 Bob            2    91
 6     291857 Bob            3    91
 7     500286 Charlotte      1    70
 8     500286 Charlotte      2    94
 9     500286 Charlotte      3    71
10     449192 Dante          1    57
11     449192 Dante          2    85
12     449192 Dante          3    84
13     372152 Ethelburga     1    74
14     372152 Ethelburga     2    91
15     372152 Ethelburga     3    70
16     627561 Felix          1    77
17     627561 Felix          2    86
18     627561 Felix          3    68
```

Exercise A - (10 min)

Answer the following, consulting the `dplyr` help files as needed.

1. Run `right_join(gradebook, emails)`. What happens? Explain.
2. Run `full_join(gradebook, emails)`. What happens? Explain.
3. Run `inner_join(gradebook, emails)`. What happens? Explain.
4. Above I ran `left_join(gradebook, emails)`. How could I have used the pipe?
5. Add a column called `name` to the `emails` tibble, containing the following names in order: `c('Joe', 'Alice', 'Ethelburga', 'Mark', 'Bob')`. Then use a left join to merge `gradebook` with `emails`. What happens? Now try setting the parameter `by = 'student_id'`. What changes?

Solution

```
# Part 1
# The result contains students whose ids are in emails. Those with ids
# in gradebook who are *not* in gradebook are dropped.
right_join(gradebook, emails)
```

```
Joining with `by = join_by(student_id)`

# A tibble: 5 × 9
  student_id name       quiz1 quiz2 quiz3 midterm1 midterm2 final email
       <dbl> <chr>      <dbl> <dbl> <dbl>    <dbl>    <dbl> <dbl> <chr>
1     192297 Alice         64    96    68       81       90    99 alice.liddell…
5     372152 Ethelburga    74    91    70       63       73    96 ethelburga@ly…
6     627561 Felix         77    86    68       78       83    75 <NA>
```

```
# Part 5
emails$name <- c('Joe', 'Alice', 'Ethelburga', 'Mark', 'Bob')
left_join(gradebook, emails)
```

```
Joining with `by = join_by(student_id, name)`

# A tibble: 6 × 9
  student_id name       quiz1 quiz2 quiz3 midterm1 midterm2 final email
       <dbl> <chr>      <dbl> <dbl> <dbl>    <dbl>    <dbl> <dbl> <chr>
1     192297 Alice         64    96    68       81       90    99 alice.liddell…
2     291857 Bob           58    91    91       75       75    79 microsoftbob@…
3     500286 Charlotte     70    94    71       81       70    74 <NA>
4     449192 Dante         57    85    84       83       94    83 <NA>
5     372152 Ethelburga    74    91    70       63       73    96 ethelburga@ly…
6     627561 Felix         77    86    68       78       83    75 <NA>
```

Exercise B - (15 min)

1. Try dropping `names_prefix` in the preceding example. What happens and why?
2. Read the help file for `billboard` from `tidyr`. Then use `pivot_longer()` to convert it to a "panel data layout," as we did with `gradebook` above.
3. Use `pivot_wider()` to reverse your `pivot_longer()` transformation from part 2.
4. Use pivoting to plot kernel density estimates of `kid.score` and `mom.iq` from `kids` in a single graph.
5. Add a column to `gradebook` called `quiz_avg` that equals a student's average across the three quizzes *dropping* the lowest score. Hint: pivot twice.

Solutions

R names the columns based on the values of `quiz`, namely `1, 2, 3`.

```
# Part 1
quiz_scores |>
  pivot_wider(names_from = quiz, values_from = score)
```

```
# A tibble: 6 × 5
  student_id name         `1`   `2`   `3`
       <dbl> <chr>      <dbl> <dbl> <dbl>
1     192297 Alice         64    96    68
2     291857 Bob           58    91    91
3     500286 Charlotte     70    94    71
4     449192 Dante         57    85    84
5     372152 Ethelburga    74    91    70
6     627561 Felix         77    86    68
```

```
# Part 2
long_billboard <- billboard |>
```

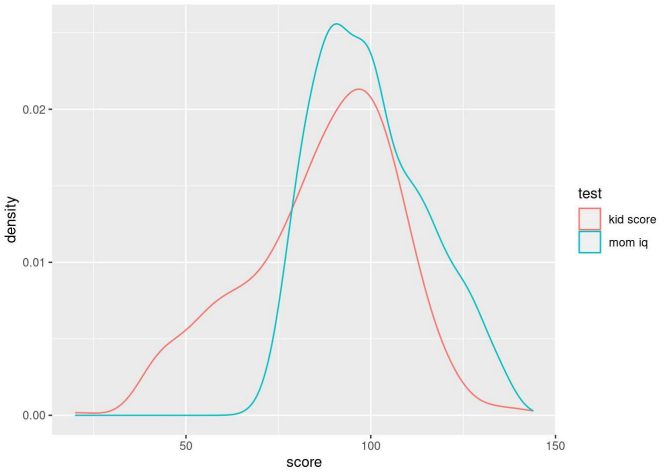
```
  pivot_longer(cols = starts_with('wk'),
               names_to = 'week',
               values_to = 'rank',
               names_prefix = 'wk')
long_billboard
```

```
# A tibble: 24,092 × 5
   artist track                   date.entered week   rank
   <chr>  <chr>                   <date>       <chr> <dbl>
 1 2 Pac  Baby Don't Cry (Keep... 2000-02-26   1        87
 2 2 Pac  Baby Don't Cry (Keep... 2000-02-26   2        82
 3 2 Pac  Baby Don't Cry (Keep... 2000-02-26   3        72
 4 2 Pac  Baby Don't Cry (Keep... 2000-02-26   4        77
 5 2 Pac  Baby Don't Cry (Keep... 2000-02-26   5        87
 6 2 Pac  Baby Don't Cry (Keep... 2000-02-26   6        94
 7 2 Pac  Baby Don't Cry (Keep... 2000-02-26   7        99
 8 2 Pac  Baby Don't Cry (Keep... 2000-02-26   8        NA
 9 2 Pac  Baby Don't Cry (Keep... 2000-02-26   9        NA
10 2 Pac  Baby Don't Cry (Keep... 2000-02-26   10       NA
# ℹ 24,082 more rows
```

```
# Part 3
long_billboard |>
  pivot_wider(names_from = week,
              values_from = rank,
              names_prefix = 'wk')
```

```
# A tibble: 317 × 79
   artist     track date.entered   wk1   wk2   wk3   wk4   wk5   wk6   wk7   wk8
   <chr>      <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 2 Pac      Baby… 2000-02-26      87    82    72    77    87    94    99    NA
 2 2Ge+her    The … 2000-09-02      91    87    92    NA    NA    NA    NA    NA
 3 3 Doors D… Kryp… 2000-04-08      81    70    68    67    66    57    54    53
 4 3 Doors D… Loser 2000-10-21      76    76    72    69    67    65    55    59
 5 504 Boyz   Wobb… 2000-04-15      57    34    25    17    17    31    36    49
 6 98^0       Give… 2000-08-19      51    39    34    26    26    19     2     2
 7 A*Teens    Danc… 2000-07-08      97    97    96    95   100    NA    NA    NA
 8 Aaliyah    I Do… 2000-01-29      84    62    51    41    38    35    35    38
 9 Aaliyah    Try … 2000-03-18      59    53    38    28    21    18    16    14
10 Adams, Yo… Open… 2000-08-26      76    76    74    69    68    67    61    58
# ℹ 307 more rows
# ℹ 68 more variables: wk9 <dbl>, wk10 <dbl>, wk11 <dbl>, wk12 <dbl>,
#   wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>, wk18 <dbl>,
#   wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>, wk24 <dbl>,
#   wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>, wk30 <dbl>,
#   wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>, wk36 <dbl>,
#   wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>, wk42 <dbl>, …
```

```
# Part 4
read_csv('https://ditraglia.com/data/child_test_data.csv') |>
  select(kid.score, mom.iq) |>
  rename(`kid score` = kid.score, `mom iq` = mom.iq) |>
  pivot_longer(c(`kid score`, `mom iq`),
               values_to = 'score',
               names_to = 'test') |>
ggplot(aes(x = score, col = test)) +
geom_density()
```




```
# Part 5
drop1_avg <- function(x){
# Calculate the mean of x dropping the lowest value
  x <- sort(x)
  mean(x[-1])
}
gradebook |>
  pivot_longer(starts_with('quiz'), names_to = 'quiz', values_to = 'score') |>
  group_by(name) |>
  mutate(quiz_avg = drop1_avg(score)) |>
  pivot_wider(names_from = 'quiz', values_from = 'score')
```

```
# A tibble: 6 × 9
# Groups:   name [6]
  student_id name       midterm1 midterm2 final quiz_avg quiz1 quiz2 quiz3
       <dbl> <chr>         <dbl>    <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl>
1     192297 Alice            81       90    99     82      64    96    68
2     291857 Bob              75       75    79     91      58    91    91
3     500286 Charlotte        81       70    74     82.5    70    94    71
4     449192 Dante            83       94    83     84.5    57    85    84
5     372152 Ethelburga       63       73    96     82.5    74    91    70
6     627561 Felix            78       83    75     81.5    77    86    68
```

Exercise C - (∞ minutes)

1. Use `rmvnorm()` to write a function that generates `n` draws from a bivariate standard normal distribution with correlation coefficient `r`. Check you work by generating a large number of simulations and calculating the sample variance-covariance matrix.
2. The function `cov()` calculates the sample covariance between X and Y as $S_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$. In contrast, the MLE $\widehat{\sigma}_{xy}$ for jointly normal (X_i, Y_i) divides by n rather than $(n-1)$. Write a function that takes a matrix with two columns and `n` rows as its input and calculates $\widehat{\sigma}_{xy}$.
3. Use the functions you wrote in the preceding two parts to carry out a simulation study investigating the bias of $\widehat{\sigma}_{xy}$. Use 5000 replications and a parameter grid of `n` $\in \{5, 10, 15, 20, 25\}$, `r` $\in \{-0.5, 0.25, 0, 0.25, 0.5\}$. Try to run it in parallel. Summarize your findings.

Solutions

```
library(furrr)
library(mvtnorm)
library(tidyr) # for expand_grid

draw_sim_data <- function(n, r) {
  var_mat <- matrix(c(1, r,
                      r, 1), 2, 2, byrow = TRUE)
  rmvnorm(n, sigma = var_mat)
}

get_estimate <- function(dat) {
  stopifnot(ncol(dat) == 2)
  x <- dat[,1]
  y <- dat[,2]
  mean((x - mean(x)) * (y - mean(y)))
}

run_sim <- function(n, r, nreps = 5000) {
  map(1:nreps, \(i) draw_sim_data(n, r)) |>
    map_dbl(get_estimate)
}

sim_params <- expand_grid(n = c(5, 10, 15, 20, 25),
                          r = c(-0.5, -0.25, 0, 0.25, 0.5))

plan(multisession, workers = 4)
my_options <- furrr_options(seed = 4321)
sim_results <- future_pmap(sim_params, run_sim, .options = my_options)

sim_bias <- sim_params |>
  mutate(sim_mean = map_dbl(sim_results, mean),
         bias = sim_mean - r)

sim_bias |>
  select(n, r, bias) |>
  ggplot(aes(x = n, y = bias)) +
  geom_point() +
  geom_line() +
  facet_wrap(~ r)
```