Data Tidying and Merging

EVR 628- Intro to Environmental Data Science

Juan Carlos Villaseñor-Derbez (JC)

Last week you were asked:

> How much money have tuna purse seiners made since 2000 when fishing for bigeye tuna (*Thunnus obesus*) in the Eastern Pacific Ocean?

We made some simplifying assumptions and got some values (a total of 3,070 M USD since 2000, or about 127 M USD per year). You are now tasked with coming up with more refined estimates. For example, we will account for the fact that the price of fish varies every year.

How we will approach this:

- Find data that shows prices per year and species
- Read them, clean them, tidy them up (The "data tidying" part)
- Combine our catch data from last week with this new price data (The "merging" part)
- Re-calculate our total revenues since 2000

This will require three pipelines:

- Tidy price data (Exercise 1)
- Wrangle catch data (Exercise 2)
- Combine tidy prices and catch data (Exercise 3)

Pipelines 1 and 3 contain tools covered this week. You should already be familiar with pipeline 2.

Exercise 1: Tidying price data

Part A: Downloading the data

Post-it up

1. In a web browser, go to ffa.int. This is the website for the Pacific Islands Forum Fisheries Agency
2. Hover over "Publication and Statistics" on the top menu
3. Select "Statistics"

4. You will be taken to a site with five items. Download the zip folder called `Economic and Development Indicators and Statistics: Tuna Fishery of the Western and Central Pacific Ocean 2024`
5. As before, place the downloaded zip file in your `EVR628/data/raw` folder and proceed to extract it
6. Open the excel file called `Compendium of Economic and Development Statistics 2024` and study the `Contents` tab
7. Can you identify the price data that we need?

- Which sheet
- What range?

Post-it down

Part B: Reading excel data

Post-it up

1. Open your RStudio project for EVR628
2. In your console, install the `readxl` package: `install.packages("readxl")`
3. Start a **new** script called `tuna_analysis_prices.R`[1]
4. Add the usual code commenting outline
5. We will need three packages: `readxl`, `janitor`, and `tidyverse`, load them at the top of your script using `library()`
6. Use `?read_excel()` to look at the documentation for the function
7. Use `read_excel()` to create a new object called `tuna_prices` and read the price data we need[2]. Immediately pipe it into `clean_names`.

Post-it down

```
library(readxl)
library(janitor)
library(tidyverse)

tuna_prices <- read_excel(path = "data/raw/Economic-and-Development-Indicators-and-Statistic
                          sheet = "B. Prices",
                          range = "A35:E63",
                          na = "na") |>
  clean_names()
```

Part C: Inspecting price data

Be prepared to discuss the following points:

Post-it up

[1] I would typically suggest to overwrite whatever we had last week in `tuna_analysis.R` because GitHub would keep a version, but I understand you might want to keep the script as is

[2] Hint: You will need to specify a file path, a sheet, and a range of cells.

1. Inspect the column names of `tuna_prices` using `colnames()` in your console.
2. How many columns and rows do we have?
3. Any missing values?
4. Do we need to make the data wider or longer?
5. Using comments, write out what the target data should be (expand my code chunk see what I wrote)

Post-it down

See an example of my description below

```
colnames(tuna_prices)
```

```
[1] "year"          "japan_fresha"  "japan_frozenb" "us_freshc"
[5] "us_frozend"
```

```
dim(tuna_prices)
```

```
[1] 28  5
```

```
tuna_prices |>
  filter_all(any_vars(is.na(.)))
```

```
# A tibble: 4 x 5
   year japan_fresha japan_frozenb us_freshc us_frozend
  <dbl>        <dbl>         <dbl>     <dbl>      <dbl>
1  1997        8204.         8169.        NA         NA
2  1998        7703.         6320.        NA         NA
3  1999        8809.         9093.        NA         NA
4  2000        9198.         8557.        NA         NA
```

```
# The final data set should have two columns: year and price. Since we have four
# prices (two markets, two presentations), I will use the average price per year.
# The tidy data set should therefore have four columns: year, market,
# presentation, and price.
```

Part D: Tidy your price data

Post-it up

1. Look at the documentation for your `pivot_*` function. What does it say about cases where `names_to` is of length > 1?
2. What about the `names_sep` argument?
3. Use the appropriate `pivot_*` function to reshape your data and save them to a new object called `tidy_tuna_prices`[3]

[3]Hint: Your `names_to` argument should be a character vector of with two items. `names_sep` should be inspired by our clever use of `snake_case`.

4. Your resulting tibble should have 104 rows and 4 columns and look like this:[4]

```
tidy_tuna_prices <- tuna_prices |>
  pivot_longer(cols = 2:5,
               names_to = c("market", "presentation"),
               names_sep = "_",
               values_to = "price",
               values_drop_na = T)

tidy_tuna_prices
```

```
# A tibble: 104 x 4
    year market presentation price
   <dbl> <chr>  <chr>        <dbl>
 1  1997 japan  fresha       8204.
 2  1997 japan  frozenb      8169.
 3  1998 japan  fresha       7703.
 4  1998 japan  frozenb      6320.
 5  1999 japan  fresha       8809.
 6  1999 japan  frozenb      9093.
 7  2000 japan  fresha       9198.
 8  2000 japan  frozenb      8557.
 9  2001 japan  fresha       8260.
10  2001 japan  frozenb      5983.
# i 94 more rows
```

Post-it down

> **!** Values in `presentation`
>
> Note that the values in the `presentation` column are not ideal. They end in `a`, `b`, `c`, and `d` due to footnotes included in Excel. For now this doesn't matter because we will quickly remove them. We'll cover some text wrangling in Week 9.

Part E: Calculate mean annual price

Post-it up

1. Modify the pipeline that creates `tidy_tuna_prices` to get the mean price per year[5]

```
tidy_tuna_prices <- tuna_prices |>
  pivot_longer(cols = 2:5,
```

[4] Hint: If you have 112 rows, remember you can use `values_drop_na = T`

[5] Hint: You will use `group_by()` and `summarize()`, as well as `|>`

```
                names_to = c("market", "presentation"),
                names_sep = "_",
                values_to = "price",
                values_drop_na = T) |>
  group_by(year) |>
  summarize(price = mean(price))

tidy_tuna_prices
```

```
# A tibble: 28 x 2
    year price
   <dbl> <dbl>
 1  1997 8186.
 2  1998 7011.
 3  1999 8951.
 4  2000 8877.
 5  2001 5633.
 6  2002 5342.
 7  2003 5285.
 8  2004 5739.
 9  2005 5554.
10  2006 5177.
# i 18 more rows
```

Post-it down

Exercise 2: Tidying tuna catch data (again)

Part A: Read the tuna catch data

> Note: You can copy-paste and modify your code from last week, but make sure your code is organized.

Post-it up

1. Read in the tuna catch data from last week
2. Filter it to retain bigeye tuna (`BET`) caught by the purse seine fleet (`PS`) since 2000
3. Calculate **total** catch by year. Your final data should have 24 rows and 2 columns, as below

Post-it down

```
# Load the data
tuna_data <- read_csv("data/raw/CatchByFlagGear/CatchByFlagGear1918-2023.csv") |>
  # Clean column names
  clean_names() |>
```

```r
  # Rename some columns
  rename(year = ano_year,
         flag = bandera_flag,
         gear = arte_gear,
         species = especies_species,
         catch = t)

ps_tuna_data <- tuna_data |>
  filter(species == "BET", # Retain BET values only
         gear == "PS",     # Retain PS values only
         year >= 2000) |>  # Retain data from 2000
  group_by(year) |>        # Specify that I am grouping by year
  # Tell summarize that I want to collapse the catch column by summing all its values
  summarize(catch = sum(catch))

ps_tuna_data
```

```
# A tibble: 24 x 2
    year catch
   <dbl> <dbl>
 1  2000 95283
 2  2001 60518
 3  2002 57422
 4  2003 53051
 5  2004 65471
 6  2005 67895
 7  2006 83837
 8  2007 63451
 9  2008 75028
10  2009 76800
# i 14 more rows
```

Exercise 3: Combine your catch and price data

Part A: Plan the join

1. Think about what type of join you want
2. What will be on the left and what will be on the right?
3. What is the key?
4. Write down, using human language, what you want to do.

Post-it up

Part B: Perform the join

1. Perform the join and save the output to an object called `tuna_revenues`

2. Create a new column that contains the annual revenue in M USD. Pay attention to the units.

Post-it down

```
tuna_revenues <- ps_tuna_data |>
  left_join(tidy_tuna_prices, by = join_by(year)) |>
  mutate(revenue = price * catch / 1e6)

tuna_revenues
```

```
# A tibble: 24 x 4
    year catch price revenue
   <dbl> <dbl> <dbl>   <dbl>
 1  2000 95283 8877.    846.
 2  2001 60518 5633.    341.
 3  2002 57422 5342.    307.
 4  2003 53051 5285.    280.
 5  2004 65471 5739.    376.
 6  2005 67895 5554.    377.
 7  2006 83837 5177.    434.
 8  2007 63451 5054.    321.
 9  2008 75028 5636.    423.
10  2009 76800 6175.    474.
# i 14 more rows
```

Part C: Answer the questions again

1. How much TOTAL revenue since 2000?
2. How much mean ANNUAL revenue since 2000?
3. Make a figure
4. How do these plot and numbers compare to what we found last week?

```
sum(tuna_revenues$revenue)
```

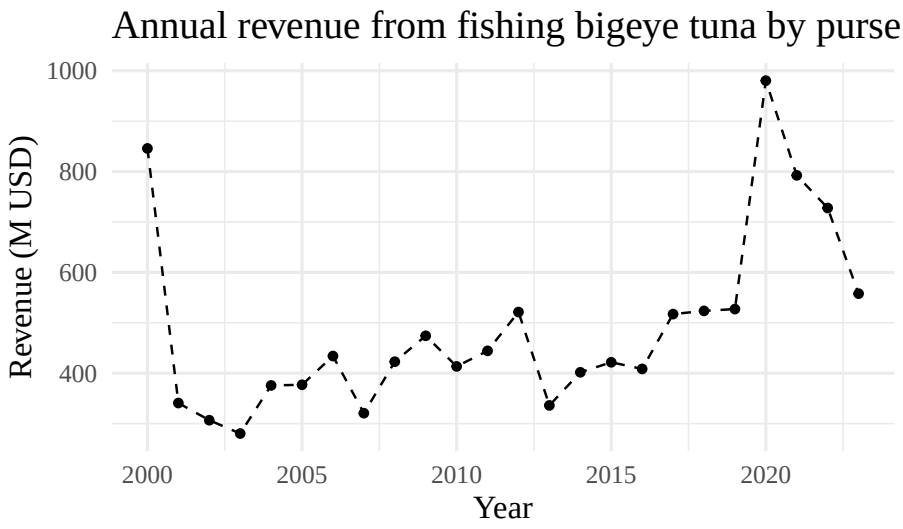
```
[1] 11752.32
```

```
mean(tuna_revenues$revenue)
```

```
[1] 489.68
```

```
# Build  plot
ggplot(data = tuna_revenues,                    # Specify my data
       mapping = aes(x = year, y = revenue)) + # And my aesthetics
  geom_line(linetype = "dashed") +              # Add a dashed line
  geom_point() +                                # With points on top
  labs(x = "Year",                              # Add some labels
       y = "Revenue (M USD)",
       title = "Annual revenue from fishing bigeye tuna by purse seine vessels",
```

```
      caption = "Data come from the IATTC") +
  # Modify the theme
  theme_minimal(base_size = 14,              # Font size 14
                base_family = "Times")       # Font family Times
```



Extra exercises for you to practice

The following four exercises use data that are built directly in R. You will need to copy and paste the provided code in your console (or R script) to make sure the objects appear in your environment.

Exercise 1: Pivot Longer Practice

Scenario: You are a TA and have been given grade data where each row represents a student and columns represent their scores on assignments 1-4. You need to calculate the mean grade for each student.[6]

```
# Create the dataset
student_scores <- tribble(
  ~student_id, ~assignment_1, ~assignment_2, ~assignment_3, ~assignment_4,
  "S001", 85, 92, 78, 88,
  "S002", 91, 89, 95, 82,
  "S003", 76, 84, 91, 79,
```

[6]Hint: Use `pivot_longer()` to transform this data so that each row represents one student-assignment-score combination, then use `group_by()` and `summarize()` to calculate the mean grade for each student.

```
  "S004", 88, 93, 87, 94
)

student_scores
```

```
# A tibble: 4 x 5
  student_id assignment_1 assignment_2 assignment_3 assignment_4
  <chr>             <dbl>        <dbl>        <dbl>        <dbl>
1 S001                 85           92           78           88
2 S002                 91           89           95           82
3 S003                 76           84           91           79
4 S004                 88           93           87           94
```

How I did it:

```
student_scores_long <- student_scores |>
  pivot_longer(cols = starts_with("assignment"),  # All columns from assignment_1 to assignm
               names_to = "assignment",           # Create new column called "assignment" wit
               values_to = "score")               # Create new column called "score" with the

student_means <- student_scores_long |>
  group_by(student_id) |>                         # Group rows by student_id
  summarize(mean_grade = mean(score))            # Calculate mean score for each student

student_means                                     # View the result
```

```
# A tibble: 4 x 2
  student_id mean_grade
  <chr>           <dbl>
1 S001             85.8
2 S002             89.2
3 S003             82.5
4 S004             90.5
```

Exercise 2: Pivot Wider Practice

Scenario: You have hurricane exposure data from different Florida counties. You are asked to build a figure showing the relationship between pressure and wind speed. Modify the data as needed and build a figure.[7]

```
# Create the dataset
hurricane_data <- tribble(
  ~county, ~metric, ~measurement,
  "Miami-Dade", "pressure", 950,
  "Miami-Dade", "precipitation", 12.5,
```

[7] Hint: Use `pivot_wider()` to transform this data so that each row represents a county and each metric becomes its own column.

```
  "Miami-Dade", "wind_speed", 85,
  "Broward", "pressure", 955,
  "Broward", "precipitation", 8.3,
  "Broward", "wind_speed", 72,
  "Palm Beach", "pressure", 960,
  "Palm Beach", "precipitation", 6.1,
  "Palm Beach", "wind_speed", 68,
  "Monroe", "pressure", 945,
  "Monroe", "precipitation", 15.2,
  "Monroe", "wind_speed", 95
)

hurricane_data
```

```
# A tibble: 12 x 3
   county     metric        measurement
   <chr>      <chr>               <dbl>
 1 Miami-Dade pressure            950
 2 Miami-Dade precipitation        12.5
 3 Miami-Dade wind_speed           85
 4 Broward    pressure            955
 5 Broward    precipitation         8.3
 6 Broward    wind_speed           72
 7 Palm Beach pressure            960
 8 Palm Beach precipitation         6.1
 9 Palm Beach wind_speed           68
10 Monroe     pressure            945
11 Monroe     precipitation        15.2
12 Monroe     wind_speed           95
```

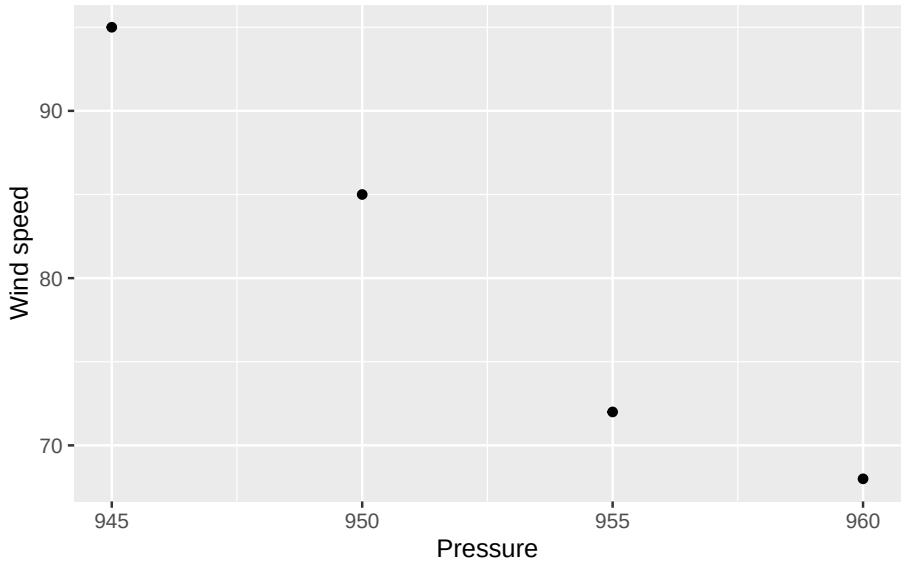
How I did it:

```
hurricane_wide <- hurricane_data |>
  pivot_wider(names_from = metric,     # Use values in "metric" column as new column names
              values_from = measurement) # Use values in "measurement" column to fill new co

hurricane_wide                       # Now each county is a row with separate columns for p
```

```
# A tibble: 4 x 4
  county     pressure precipitation wind_speed
  <chr>         <dbl>         <dbl>      <dbl>
1 Miami-Dade      950          12.5         85
2 Broward         955           8.3         72
3 Palm Beach      960           6.1         68
4 Monroe          945          15.2         95
```

```r
ggplot(hurricane_wide,
       aes(x = pressure, y = wind_speed)) +
  geom_point() +
  labs(x = "Pressure",
       y = "Wind speed")
```



Exercise 3: Joining Data Practice

Scenario: You have two datasets - one with student information and another with their enrollment details. You need to be able to identify the names of students taking stats courses.

```r
# Create the datasets
students <- tribble(
  ~student_id, ~name, ~age,
  "S001", "Alice Johnson", 20,
  "S002", "Bob Smith", 22,
  "S003", "Carol Davis", 19,
  "S004", "David Wilson", 21,
  "S005", "Eva Brown", 23
)

enrollments <- tribble(
  ~stdt_identifier, ~course, ~credits,
  "S001", "Statistics", 3,
  "S001", "Biology", 4,
  "S002", "Statistics", 3,
```

```
  "S003", "Chemistry", 4,
  "S004", "Statistics", 3,
  "S004", "Physics", 4,
  "S006", "Math", 3
)

students
```

```
# A tibble: 5 x 3
  student_id name            age
  <chr>      <chr>         <dbl>
1 S001       Alice Johnson    20
2 S002       Bob Smith        22
3 S003       Carol Davis      19
4 S004       David Wilson     21
5 S005       Eva Brown        23
```

```
enrollments
```

```
# A tibble: 7 x 3
  stdt_identifier course     credits
  <chr>           <chr>        <dbl>
1 S001            Statistics       3
2 S001            Biology          4
3 S002            Statistics       3
4 S003            Chemistry        4
5 S004            Statistics       3
6 S004            Physics          4
7 S006            Math             3
```

How I did it:

```
combined_data <- students |>
  left_join(enrollments, by = join_by(student_id == stdt_identifier)) # Keep all students, r

combined_data                         # View the combined data
```

```
# A tibble: 7 x 5
  student_id name            age course     credits
  <chr>      <chr>         <dbl> <chr>        <dbl>
1 S001       Alice Johnson    20 Statistics       3
2 S001       Alice Johnson    20 Biology          4
3 S002       Bob Smith        22 Statistics       3
4 S003       Carol Davis      19 Chemistry        4
5 S004       David Wilson     21 Statistics       3
6 S004       David Wilson     21 Physics          4
7 S005       Eva Brown        23 <NA>            NA
```

```r
statistics_students <- combined_data |>
  filter(course == "Statistics") |> # Keep only rows where course equals "Statistics"
  select(name, course)             # Keep only the name and course columns

statistics_students                # View the result
```

```
# A tibble: 3 x 2
  name          course
  <chr>         <chr>
1 Alice Johnson Statistics
2 Bob Smith     Statistics
3 David Wilson  Statistics
```

Exercise 4: Sharks!

Scenario: You have swimming data from beachgoers and bull shark detection data from acoustic telemetry during fourth of July. The swimming data tell you when someone entered and left the water. The shark detection data tells you which sharks were detected within the acoustic array in front of the beach, and the time of detection. Who was in the water while a shark was nearby?[8]

```r
# Create the datasets
swimming_data <- tribble(
  ~name, ~swim_start, ~swim_end,
  "Alice", "2024-07-04 10:30:00", "2024-07-04 11:15:00",
  "Bob", "2024-07-04 10:45:00", "2024-07-04 11:30:00",
  "Carol", "2024-07-04 11:00:00", "2024-07-04 11:45:00",
  "David", "2024-07-04 11:20:00", "2024-07-04 12:00:00",
  "Eva", "2024-07-04 12:10:00", "2024-07-04 12:45:00"
) |>
  mutate(swim_start = as_datetime(swim_start),
         swim_end = as_datetime(swim_end))

shark_detections <- tribble(
  ~shark_id, ~detection_time,
  "SH001", "2024-07-04 09:40:00",
  "SH002", "2024-07-04 11:25:00",
  "SH003", "2024-07-04 11:35:00"
) |>
  mutate(detection_time = as_datetime(detection_time))

swimming_data
```

```
# A tibble: 5 x 3
```

[8] Hint: Look at the documentation for `join_by()`. What does it say about "Overlap helpers"? You'll want to use the `between()` function.

```
  name  swim_start          swim_end
  <chr> <dttm>              <dttm>
1 Alice 2024-07-04 10:30:00 2024-07-04 11:15:00
2 Bob   2024-07-04 10:45:00 2024-07-04 11:30:00
3 Carol 2024-07-04 11:00:00 2024-07-04 11:45:00
4 David 2024-07-04 11:20:00 2024-07-04 12:00:00
5 Eva   2024-07-04 12:10:00 2024-07-04 12:45:00
```

```
shark_detections
```

```
# A tibble: 3 x 2
  shark_id detection_time
  <chr>    <dttm>
1 SH001    2024-07-04 09:40:00
2 SH002    2024-07-04 11:25:00
3 SH003    2024-07-04 11:35:00
```

How I did it:

```
swimmer_shark_overlap <- shark_detections |>
  inner_join(swimming_data,            # Note that I am using an inner join. Play with inner
             by = join_by(between(detection_time, swim_start, swim_end))) # Find swimmers in

swimmer_shark_overlap               # View all the overlap data
```

```
# A tibble: 5 x 5
  shark_id detection_time      name  swim_start          swim_end
  <chr>    <dttm>              <chr> <dttm>              <dttm>
1 SH002    2024-07-04 11:25:00 Bob   2024-07-04 10:45:00 2024-07-04 11:30:00
2 SH002    2024-07-04 11:25:00 Carol 2024-07-04 11:00:00 2024-07-04 11:45:00
3 SH002    2024-07-04 11:25:00 David 2024-07-04 11:20:00 2024-07-04 12:00:00
4 SH003    2024-07-04 11:35:00 Carol 2024-07-04 11:00:00 2024-07-04 11:45:00
5 SH003    2024-07-04 11:35:00 David 2024-07-04 11:20:00 2024-07-04 12:00:00
```

```
at_risk_swimmers <- swimmer_shark_overlap |>
  group_by(name) |>  # Keep only the columns we want to see
  summarize(n_sharks_near = n_distinct(shark_id))

at_risk_swimmers                    # These swimmers were in the water when sharks were detec
```

```
# A tibble: 3 x 2
  name  n_sharks_near
  <chr>         <int>
1 Bob               1
2 Carol             2
3 David             2
```