AE 07: Data wrangling with rowwise/column-wise operations

Suggested answers

APPLICATION EXERCISE ANSWERS

MODIFIED
September 30, 2024

Packages

We will use the following packages in this application exercise.

- **tidyverse**: For data import, wrangling, and visualization.
- **janitor:** For cleaning column names.

```
library(tidyverse)
library(janitor)
library(scales)
```

Powerball

Last class we studied Powerball jackpots over time. Today we will continue this journey and focus on Colorado winners in Match N Powerball play, prizes available to players who match the red Powerball number and anywhere between 0-4 white ball numbers.

Import and clean the data

The dataset is available for download as a CSV file.

Demo: Import the data file. Store it as `powerball_raw`.

```
powerball_raw <- read_csv(file = "data/POWERBALL-from_0001-01-01_to_2024-09-24.csv")
```

```
Rows: 2577 Columns: 61
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (31): Draw date, Last Day To Claim, Winning Numbers, Jackpot, Jackpot Ca...
dbl (30): Powerball, Power Play, Jackpot Winners, Jackpot CO Winners, Match ...

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.
```

```
powerball_raw
```

```
# A tibble: 2,577 × 61
   `Draw date`      `Last Day To Claim` `Winning Numbers` Powerball `Power Play`
   <chr>            <chr>               <chr>                 <dbl>        <dbl>
 1 Monday, 9/23/24  03/22/2025          15 - 21 - 25 - 3…        19            3
 2 Saturday, 9/21/… 03/20/2025          17 - 19 - 21 - 3…        14            2
 3 Wednesday, 9/18… 03/17/2025          1 - 11 - 22 - 47…         7            4
 4 Monday, 9/16/24  03/15/2025          8 - 9 - 11 - 27 …        17            5
 5 Saturday, 9/14/… 03/13/2025          29 - 34 - 38 - 4…        16            2
 6 Wednesday, 9/11… 03/10/2025          10 - 12 - 55 - 6…         3            3
 7 Monday, 9/9/24   03/08/2025          1 - 16 - 21 - 47…         5            3
 8 Saturday, 9/7/24 03/06/2025          14 - 34 - 37 - 5…        20            2
 9 Wednesday, 9/4/… 03/03/2025          7 - 10 - 21 - 33…        20            3
10 Monday, 9/2/24   03/01/2025          8 - 42 - 46 - 48…        22            3
# ℹ 2,567 more rows
# ℹ 56 more variables: Jackpot <chr>, `Jackpot Cash Value` <chr>,
#   `Jackpot Winners` <dbl>, `Jackpot CO Winners` <dbl>, `Match 5 Prize` <chr>,
#   `Match 5 CO Winners` <dbl>, `Match 5 Prize (with Power Play)` <chr>,
#   `Match 5 CO Winners (with Power Play)` <dbl>,
#   `Match 4 + Powerball Prize` <chr>, `Match 4 + Powerball CO Winners` <dbl>,
#   `Match 4 + Powerball Prize (with Power Play)` <chr>, …
```

Your turn: Clean the raw data to fix the following issues:

- Standardize the column names using `snake_case` format.
- Create columns with appropriate data types for any date variables date of the drawing as well as the weekday. Append these columns to the beginning of the data frame.
- Fix all of the currency columns to be formatted as numeric types.

Store the cleaned data frame as `powerball`.

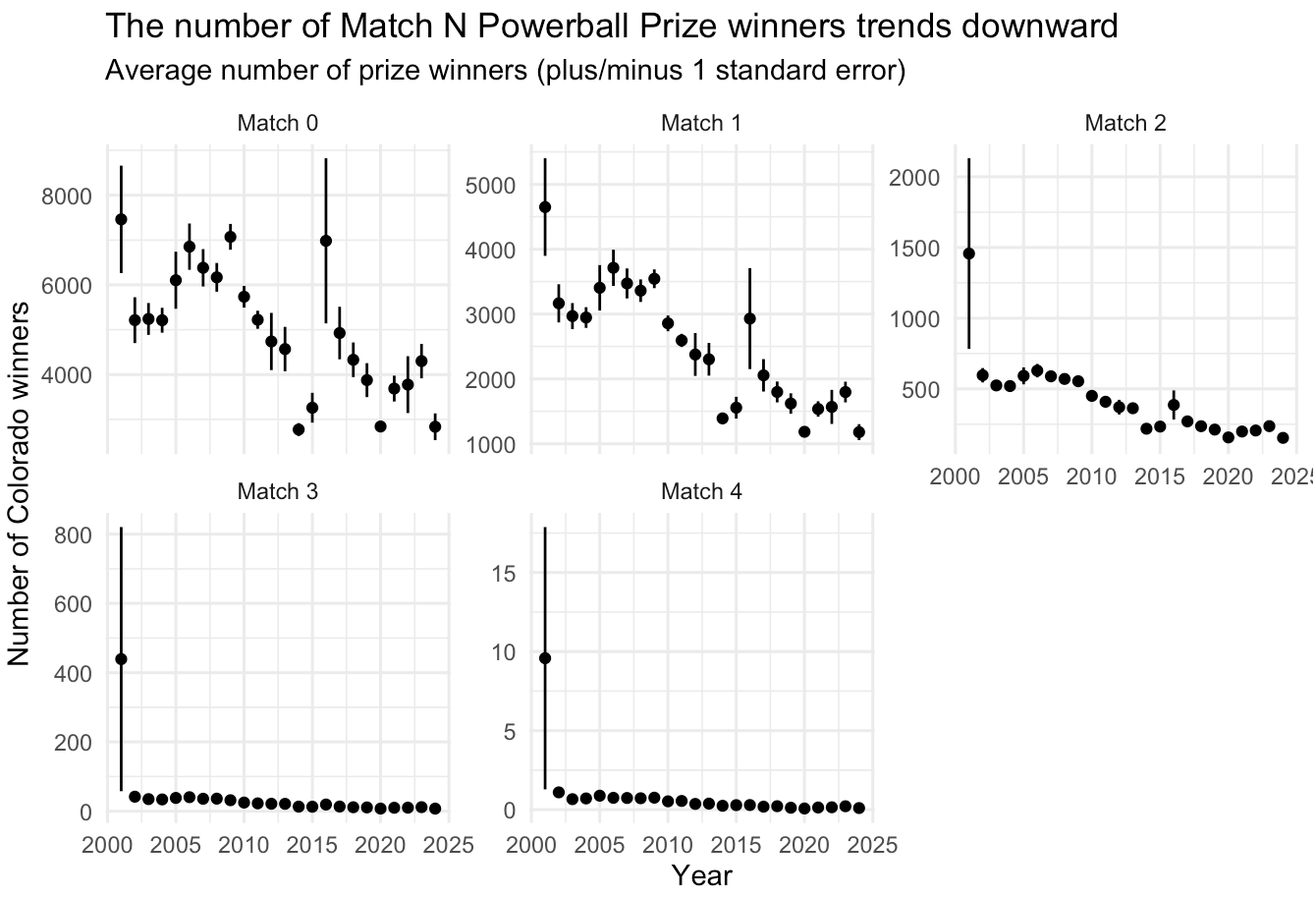
```
powerball <-  powerball_raw |>
  clean_names() |>
  # separate draw_date into two variables, clean both
  separate_wider_delim(
    cols = draw_date,
    delim = ",",
    names = c(NA, "draw_date")
  ) |>
  mutate(
    draw_date = mdy(draw_date),
    last_day_to_claim = mdy(last_day_to_claim),
    draw_weekday = wday(x = draw_date, label = TRUE),
    .before = last_day_to_claim
  ) |>
  # convert all currency columns to numeric type
  mutate(
    across(
      .cols = c(where(is.character), -contains("winning_numbers")),
```

```
      .fn = parse_number
    )
  )
powerball
```

```
# A tibble: 2,577 × 62
   draw_date  draw_weekday last_day_to_claim winning_numbers        powerball
   <date>     <ord>        <date>            <chr>                      <dbl>
 1 2024-09-23 Mon          2025-03-22        15 - 21 - 25 - 37 - 45        19
 2 2024-09-21 Sat          2025-03-20        17 - 19 - 21 - 37 - 45        14
 3 2024-09-18 Wed          2025-03-17        1 - 11 - 22 - 47 - 68          7
 4 2024-09-16 Mon          2025-03-15        8 - 9 - 11 - 27 - 31          17
 5 2024-09-14 Sat          2025-03-13        29 - 34 - 38 - 48 - 56        16
 6 2024-09-11 Wed          2025-03-10        10 - 12 - 55 - 65 - 67         3
 7 2024-09-09 Mon          2025-03-08        1 - 16 - 21 - 47 - 60          5
 8 2024-09-07 Sat          2025-03-06        14 - 34 - 37 - 55 - 63        20
 9 2024-09-04 Wed          2025-03-03        7 - 10 - 21 - 33 - 59         20
10 2024-09-02 Mon          2025-03-01        8 - 42 - 46 - 48 - 53         22
# ℹ 2,567 more rows
# ℹ 57 more variables: power_play <dbl>, jackpot <dbl>,
#   jackpot_cash_value <dbl>, jackpot_winners <dbl>, jackpot_co_winners <dbl>,
#   match_5_prize <dbl>, match_5_co_winners <dbl>,
#   match_5_prize_with_power_play <dbl>,
#   match_5_co_winners_with_power_play <dbl>, match_4_powerball_prize <dbl>,
#   match_4_powerball_co_winners <dbl>, …
```

Analyze the data

Our goal is to reproduce the following visualization:



In order to accomplish this, we have a few challenges ahead. We will need to:

- Determine the **year** for every drawing
- Calculate the **mean** and **standard error** of the number of winners for each Match N + Powerball prize for every year
- Structure the data frame so we have one row for each year and prize, and separate columns for the means and standard errors

Generate the `year` variable

Your turn: Generate a `year` variable from the `draw_date` column.

```
powerball |>
  # generate year variable
  mutate(
    year = year(x = draw_date),
    .before = everything()
  )
```

```
# A tibble: 2,577 × 63
    year draw_date  draw_weekday last_day_to_claim winning_numbers     powerball
   <dbl> <date>     <ord>        <date>            <chr>                   <dbl>
 1  2024 2024-09-23 Mon          2025-03-22        15 - 21 - 25 - 37 …        19
```

```
 2  2024 2024-09-21 Sat          2025-03-20        17 - 19 - 21 - 37 …        14
 3  2024 2024-09-18 Wed          2025-03-17        1 - 11 - 22 - 47 -…         7
 4  2024 2024-09-16 Mon          2025-03-15        8 - 9 - 11 - 27 - …        17
 5  2024 2024-09-14 Sat          2025-03-13        29 - 34 - 38 - 48 …        16
 6  2024 2024-09-11 Wed          2025-03-10        10 - 12 - 55 - 65 …         3
 7  2024 2024-09-09 Mon          2025-03-08        1 - 16 - 21 - 47 -…         5
 8  2024 2024-09-07 Sat          2025-03-06        14 - 34 - 37 - 55 …        20
 9  2024 2024-09-04 Wed          2025-03-03        7 - 10 - 21 - 33 -…        20
10  2024 2024-09-02 Mon          2025-03-01        8 - 42 - 46 - 48 -…        22
# ℹ 2,567 more rows
# ℹ 57 more variables: power_play <dbl>, jackpot <dbl>,
#   jackpot_cash_value <dbl>, jackpot_winners <dbl>, jackpot_co_winners <dbl>,
#   match_5_prize <dbl>, match_5_co_winners <dbl>,
#   match_5_prize_with_power_play <dbl>,
#   match_5_co_winners_with_power_play <dbl>, match_4_powerball_prize <dbl>,
#   match_4_powerball_co_winners <dbl>, …
```

Calculate means and standard errors

Your turn: Calculate the mean and standard error for each of the number of winners of the Match N + Powerball prizes for each year.

> **Tip**
>
> Recall the formula for the standard error of a sample mean is:
>
> $$\begin{aligned} \text{s.e.} &= \sqrt{\frac{\text{Variance}(X)}{\text{Sample size}}} \\ &= \frac{\text{Standard deviation}(X)}{\sqrt{\text{Sample size}}} \end{aligned}$$

```
powerball |>
  # generate year variable
  mutate(
    year = year(x = draw_date),
    .before = everything()
  ) |>
  # calculate mean and se for the match N powerball winner columns
  summarize(
    across(
      .cols = starts_with("match") & contains("powerball") & ends_with("winners"),
      .fns = list(mean = mean, se = \(x) sd(x) / sqrt(n()))
    ),
    # do this for each year in the dataset
    .by = year
  )
```

```
# A tibble: 24 × 11
    year match_4_powerball_co_wi…¹ match_4_powerball_co…² match_3_powerball_co…³
   <dbl>                     <dbl>                  <dbl>                  <dbl>
 1  2024                     0.104                 0.0286                   7.39
 2  2023                     0.218                 0.0475                  11.6
 3  2022                     0.153                 0.0363                  10.1
 4  2021                     0.138                 0.0312                   9.88
 5  2020                     0.0762                0.0260                   7.49
 6  2019                     0.125                 0.0326                  10.8
 7  2018                     0.221                 0.0510                  11.3
 8  2017                     0.192                 0.0412                  13.4
 9  2016                     0.295                 0.103                   19.2
10  2015                     0.288                 0.0839                  12.9
# ℹ 14 more rows
# ℹ abbreviated names: ¹match_4_powerball_co_winners_mean,
#   ²match_4_powerball_co_winners_se, ³match_3_powerball_co_winners_mean
# ℹ 7 more variables: match_3_powerball_co_winners_se <dbl>,
#   match_2_powerball_co_winners_mean <dbl>,
#   match_2_powerball_co_winners_se <dbl>,
#   match_1_powerball_co_winners_mean <dbl>, …
```

Clean up column names

Your turn: Remove `"powerball_co_winners_"` from each column name.

> **Tip**
>
> `rename()` does not allow use of the `across()` function. Instead, check out `rename_with()` from the **dplyr** package.

> **Tip**
>
> **stringr** contains many functions for working with character strings. Check out the cheat sheet for examples!

```
powerball |>
  # generate year variable
  mutate(
    year = year(x = draw_date),
    .before = everything()
  ) |>
  # calculate mean and se for the match N powerball winner columns
  summarize(
    across(
      .cols = starts_with("match") & contains("powerball") & ends_with("winners"),
      .fns = list(mean = mean, se = \(x) sd(x) / sqrt(n()))
    ),
    # do this for each year in the dataset
    .by = year
  ) |>
  # remove extraneous string from columns
```

```
  rename_with(
    .cols = starts_with("match"),
    .fn = \(x) str_remove(string = x, pattern = "powerball_co_winners_")
  )
```

```
# A tibble: 24 × 11
    year match_4_mean match_4_se match_3_mean match_3_se match_2_mean match_2_se
   <dbl>        <dbl>      <dbl>        <dbl>      <dbl>        <dbl>      <dbl>
 1  2024       0.104      0.0286         7.39      0.834         153.      16.0
 2  2023       0.218      0.0475        11.6       1.15          236.      21.4
 3  2022       0.153      0.0363        10.1       1.66          205.      34.4
 4  2021       0.138      0.0312         9.88      0.804         198.      15.5
 5  2020       0.0762     0.0260         7.49      0.481         156.       7.62
 6  2019       0.125      0.0326        10.8       1.18          212.      20.9
 7  2018       0.221      0.0510        11.3       1.04          236.      21.4
 8  2017       0.192      0.0412        13.4       1.89          270.      34.8
 9  2016       0.295      0.103         19.2       5.47          386.     103.
10  2015       0.288      0.0839        12.9       1.55          233.      26.1
# ℹ 14 more rows
# ℹ 4 more variables: match_1_mean <dbl>, match_1_se <dbl>, match_0_mean <dbl>,
#   match_0_se <dbl>
```

Restructure data frame to appropriate form for visualization

Demo: We need the structure to be one row for each year and prize (i.e. 0, 1, 2, 3, 4) and separate columns for the means and standard errors. We can use `pivot_longer()` to accomplish this task, but it's a bit more complicated than past pivoting operations since the column names contain both a variable (e.g. `match_4`) and a variable name (i.e. `mean` or `se`). We can leverage the `names_to` argument and its ability to pass in a character vector with multiple values to create multiple columns in the resulting data frame. According to the documentation,

> If length > 1, multiple columns will be created. In this case, one of `names_sep` or `names_pattern` must be supplied to specify how the column names should be split. There are also two additional character values you can take advantage of:
>
> - NA will discard the corresponding component of the column name.
> - `".value"` **indicates that the corresponding component of the column name defines the name of the output column containing the cell values, overriding** `values_to` **entirely.**

```
powerball |>
  # generate year variable
  mutate(
    year = year(x = draw_date),
    .before = everything()
  ) |>
  # calculate mean and se for the match N powerball winner columns
  summarize(
```

```
    across(
      .cols = starts_with("match") & contains("powerball") & ends_with("winners"),
      .fns = list(mean = mean, se = \(x) sd(x) / sqrt(n())),
      .names = "{.col}_{.fn}"
    ),
    # do this for each year in the dataset
    .by = year
  ) |>
  # remove extraneous string from columns
  rename_with(
    .cols = starts_with("match"),
    .fn = \(x) str_remove(string = x, pattern = "powerball_co_winners_")
  ) |>
  # restructure to one row per year per game
  # separate columns for mean and se
  pivot_longer(
    cols = -year,
    # columns contain a variable and a variable name
    names_to = c("game", ".value"),
    # ignore column prefix
    names_prefix = "match_",
    # separating character
    names_sep = "_"
  ) |>
  # reformat game column values for visualization
  mutate(game = str_glue("Match {game}"))
```

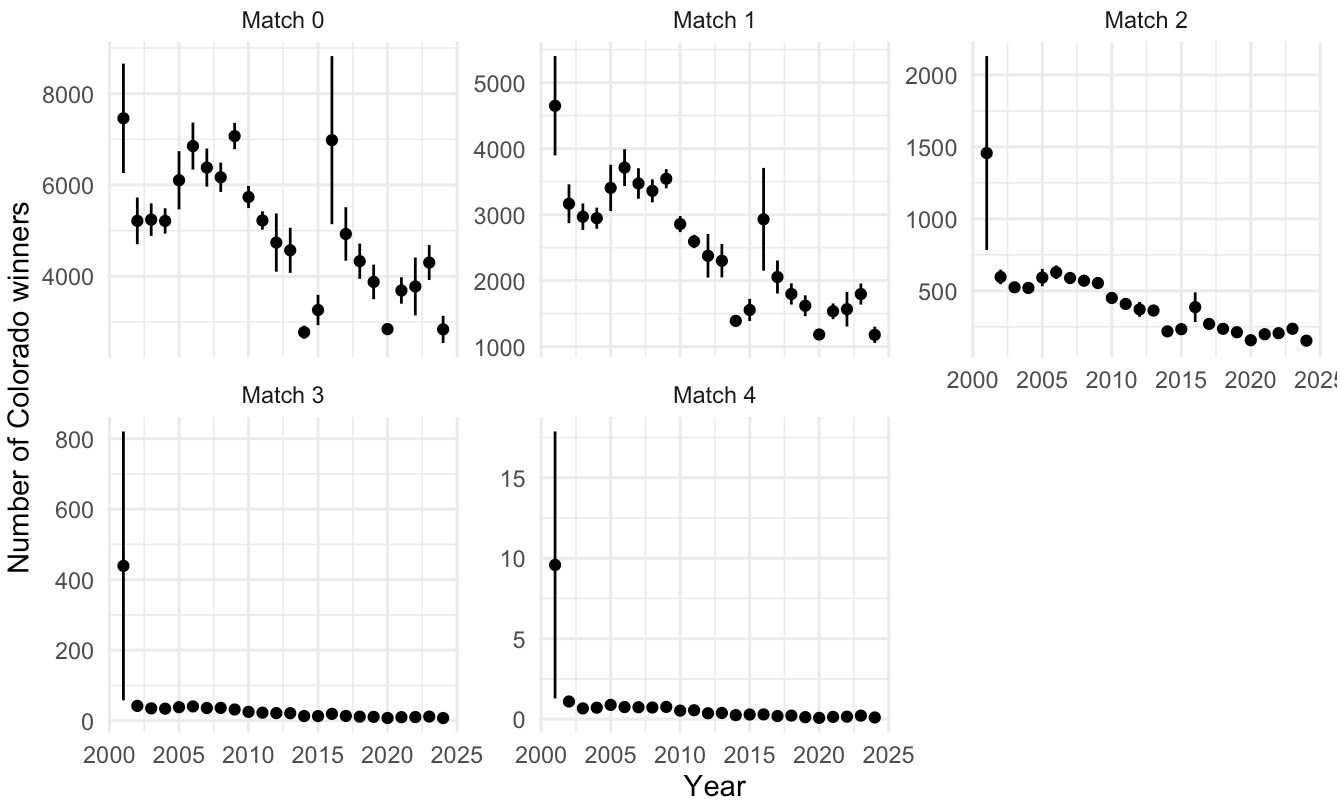
```
# A tibble: 120 × 4
    year game        mean       se
   <dbl> <glue>     <dbl>    <dbl>
 1  2024 Match 4    0.104   0.0286
 2  2024 Match 3    7.39    0.834
 3  2024 Match 2  153.     16.0
 4  2024 Match 1 1179.    123.
 5  2024 Match 0 2835.    299.
 6  2023 Match 4    0.218   0.0475
 7  2023 Match 3   11.6     1.15
 8  2023 Match 2  236.     21.4
 9  2023 Match 1 1797.    160.
10  2023 Match 0 4300.    384.
# ℹ 110 more rows
```

Plot the data

Demo: Now that we have the appropriate data structure, we can create the visualization.

```
powerball |>
  # generate year variable
  mutate(
```

```
  year = year(x = draw_date),
  .before = everything()
) |>
# calculate mean and se for the match N powerball winner columns
summarize(
  across(
    .cols = starts_with("match") & contains("powerball") & ends_with("winners"),
    .fns = list(mean = mean, se = \(x) sd(x) / sqrt(n())),
    .names = "{.col}_{.fn}"
  ),
  # do this for each year in the dataset
  .by = year
) |>
# remove extraneous string from columns
rename_with(
  .cols = starts_with("match"),
  .fn = \(x) str_remove(string = x, pattern = "powerball_co_winners_")
) |>
# restructure to one row per year per game
# separate columns for mean and se
pivot_longer(
  cols = -year,
  # columns contain a variable and a variable name
  names_to = c("game", ".value"),
  # ignore column prefix
  names_prefix = "match_",
  # separating character
  names_sep = "_"
) |>
# reformat game column values for visualization
mutate(game = str_glue("Match {game}")) |>
ggplot(mapping = aes(x = year, y = mean)) +
geom_point() +
geom_linerange(mapping = aes(
  ymin = mean - se,
  ymax = mean + se
)) +
facet_wrap(facets = vars(game), scales = "free_y") +
labs(
  title = "The number of Match N Powerball Prize winners trends downward",
  subtitle = "Average number of prize winners (plus/minus 1 standard error)",
  x = "Year",
  y = "Number of Colorado winners",
  caption = "Source: Colorado Lottery"
) +
theme_minimal()
```



Session information