Lab 2 Airlines, Mixed Drinks to Tidy Data

Darwhin Gomez

2024-09-18

Overview

In this assignment you will work to tidy, clean, and analyze two different datasets, the first is a small dataset contained in a csv file called flightdelays.csv, and the second called MixedDrinkRecipes-Prep.csv.

The most important book chapters which cover the techniques you will practice here are R4DS Chapters 5 and 7. Also helpful are the `tidyr` vignette on pivoting and the ggplot help page on the geom_dotplot.

Submit your completed assignment on the course brightspace page by uploading your `.qmd` file and a compiled `pdf` or link to a compiled `html`, which you could host on your `github` or `rpubs` page as you wish.

Part 1: Airplane flight delays

Consider the following dataset:

Airline		Los_Angeles	Phoenix	San_Diego	San_Francisco	Seattle
ALASKA	On_Time	497	221	212	503	1841
	Delayed	62	12	20	102	305
AM WEST	On_Time	694	4840	383	320	301
	Delayed	117	415	65	129	61

The above table describes arrival delays for two different airlines across several destinations. The numbers correspond the number of flights that were in either the delayed category or the on time category.

Problems

Problem 1: Read the information from `flightdelays.csv` into R, and use `tidyr` and `dplyr` to convert this data into a tidy/tall format with names and complete data for all columns. Your final data frame should have `City`, `On_Time_Flights` and `Delayed_Flights` as columns (the exact names are up to you). In addition to `pivot_longer`, `pivot_wider` and `rename`, you might find the `tidyr` function `fill` helpful for completing this task efficiently. Although this is a small dataset that you could easily reshape by hand, you should solve this problem using tidyverse functions that do the work for you.

```
flight_long <- flightdelays2|>
  pivot_longer(Los_Angeles:Seattle,
               names_to = "City",
               values_to = "Values")
```

```
colnames(flight_long)[which(names(flight_long) == "...1")] <- "Carrier"
```

```
colnames(flight_long)[which(names(flight_long) == "...2")] <- "Status"
```

```
  flight_long_filled<- flight_long |>
    fill(Carrier, .direction = "down")
  print(flight_long_filled)
```

```
# A tibble: 20 x 4
   Carrier Status  City          Values
   <chr>   <chr>   <chr>          <dbl>
 1 ALASKA  On_Time Los_Angeles      497
 2 ALASKA  On_Time Phoenix          221
 3 ALASKA  On_Time San_Diego        212
 4 ALASKA  On_Time San_Francisco    503
 5 ALASKA  On_Time Seattle         1841
 6 ALASKA  Delayed Los_Angeles       62
 7 ALASKA  Delayed Phoenix           12
 8 ALASKA  Delayed San_Diego         20
 9 ALASKA  Delayed San_Francisco    102
10 ALASKA  Delayed Seattle          305
11 AM WEST On_Time Los_Angeles      694
12 AM WEST On_Time Phoenix         4840
13 AM WEST On_Time San_Diego        383
14 AM WEST On_Time San_Francisco    320
```

```
15 AM WEST On_Time Seattle          301
16 AM WEST Delayed Los_Angeles      117
17 AM WEST Delayed Phoenix          415
18 AM WEST Delayed San_Diego         65
19 AM WEST Delayed San_Francisco    129
20 AM WEST Delayed Seattle           61
```

```
flight_tidy <- flight_long_filled|>
  pivot_wider(names_from = Status,
              values_from = Values)
print(flight_tidy)
```

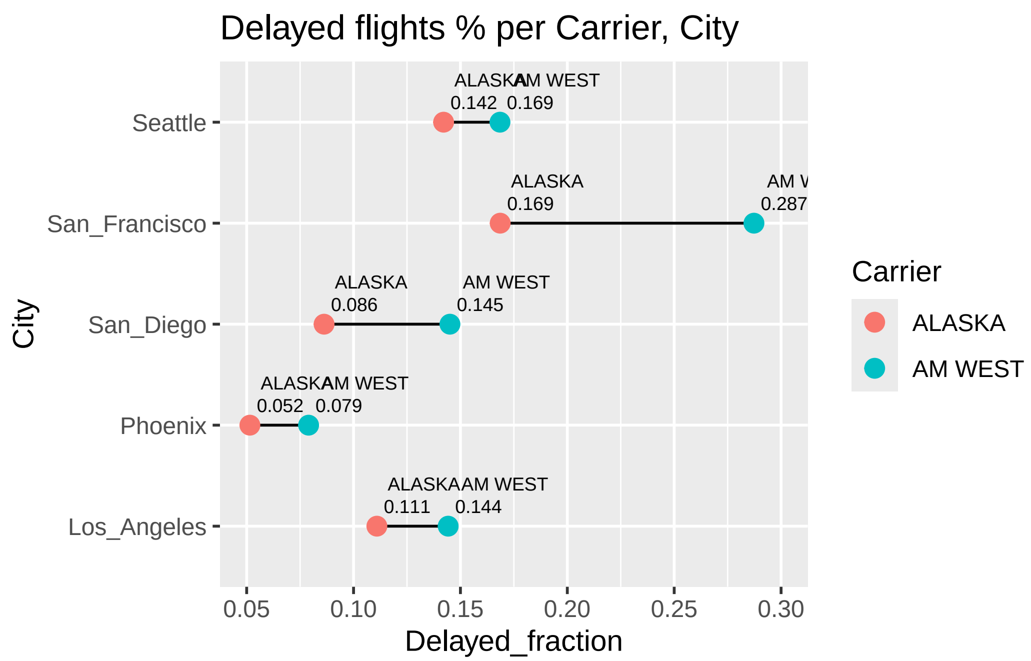
```
# A tibble: 10 x 4
   Carrier City          On_Time Delayed
   <chr>   <chr>           <dbl>   <dbl>
 1 ALASKA  Los_Angeles       497      62
 2 ALASKA  Phoenix           221      12
 3 ALASKA  San_Diego         212      20
 4 ALASKA  San_Francisco     503     102
 5 ALASKA  Seattle          1841     305
 6 AM WEST Los_Angeles       694     117
 7 AM WEST Phoenix          4840     415
 8 AM WEST San_Diego         383      65
 9 AM WEST San_Francisco     320     129
10 AM WEST Seattle           301      61
```

Problem 2: Take the data-frame that you tidied and cleaned in Problem 1 and create additional columns which contain the fraction of on-time and delayed flights at each airport. Then create a Cleveland Multiway Dot Plot (see this tutorial page for a description for how) to visualize the difference in flight delays between the two airlines at each city in the dataset. Compare the airlines and airports using the dot-plot- what are your conclusions?

```
flight_tidy|>
  group_by(City,Carrier)|>
  mutate(
    On_Time_Fraction = On_Time / (On_Time + Delayed),
         Delayed_fraction = Delayed / (On_Time + Delayed))|>

  ggplot(aes(Delayed_fraction,City))+
```

```r
        geom_line(aes(group = City))+
        geom_point(aes(color = Carrier), size = 3)+
        geom_text(aes(label = round(Delayed_fraction, 3)),
          vjust = -1,hjust = -0.15 , color = "black", size = 2.5) +
        geom_text(aes(label = Carrier),
          position = position_nudge(y = .42),
          hjust = -0.15, color = "black", size = 2.5)+
        labs(title="Delayed flights % per Carrier, City ")+
        scale_x_continuous(limits = c(0.05, 0.3),
                           breaks = seq(0, 0.3, by = 0.05))
```



Alaska Airlines flights are delayed less often AM West flights, San Francisco experiences delays most often then the other cities in the Data Frame.

Part 2: Mixed Drink Recipes

In the second part of this assignment we will be working with a dataset containing ingredients for different types of mixed drinks. This dataset is untidy and messy- it is in a wide data format and contains some inconsistencies that should be fixed.

Problems

Problem 3: Load the mixed drink recipe dataset into R from the file `MixedDrinkRecipes-prep.csv`, which you can download from my github page by clicking here. The variables `ingredient1` through `ingredient6` list the ingredients of the cocktail listed in the `name` column. Notice that there are many `NA` values in the ingredient columns, indicating that most cocktails have under 6 ingredients.

Tidy this dataset using `pivot_longer` to create a new data frame where each there is a row corresponding to each ingredient of all the cocktails, and an additional variable specifying the "rank" of that cocktail in the original recipe, i.e. it should look like this:

name	category	Ingredient_Rank	Ingredient
Gauguin	Cocktail Classics	1	Light Rum
Gauguin	Cocktail Classics	2	Passion Fruit Syrup
Gauguin	Cocktail Classics	3	Lemon Juice
Gauguin	Cocktail Classics	4	Lime Juice
Fort Lauderdale	Cocktail Classics	1	Light Rum

where the data-type of `Ingredient_Rank` is an integer. Hint: Use the `parse_number()` function in mutate after your initial pivot.

```
MixedDrinkRecipes_Prep <- read_csv("MixedDrinkRecipes-Prep.csv")
```

```
Rows: 990 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (8): name, category, ingredient1, ingredient2, ingredient3, ingredient4,...

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.
```

```r
mixed_drinks_long <- MixedDrinkRecipes_Prep |>
  pivot_longer(
    cols = ingredient1:ingredient6,
    names_to = "Ingredient Rank",
    values_to = "Ingredient"
  ) |>
  mutate(
    `Ingredient Rank` = parse_number(`Ingredient Rank`),


  )|>
  filter(!is.na(Ingredient))
print(mixed_drinks_long)|>
  slice_head(n = 10)
```

```
# A tibble: 3,934 x 4
   name            category              `Ingredient Rank` Ingredient
   <chr>           <chr>                             <dbl> <chr>
 1 Gauguin         Cocktail Classics                     1 Light Rum
 2 Gauguin         Cocktail Classics                     2 Passion Fruit Syrup
 3 Gauguin         Cocktail Classics                     3 Lemon Juice
 4 Gauguin         Cocktail Classics                     4 Lime Juice
 5 Fort Lauderdale Cocktail Classics                     1 Light Rum
 6 Fort Lauderdale Cocktail Classics                     2 Sweet Vermouth
 7 Fort Lauderdale Cocktail Classics                     3 Juice of Orange
 8 Fort Lauderdale Cocktail Classics                     4 Juice of a Lime
 9 Apple Pie       Cordials and Liqueurs                 1 Apple schnapps
10 Apple Pie       Cordials and Liqueurs                 2 Cinnamon schnapps
# i 3,924 more rows
```

```
# A tibble: 10 x 4
   name            category              `Ingredient Rank` Ingredient
   <chr>           <chr>                             <dbl> <chr>
 1 Gauguin         Cocktail Classics                     1 Light Rum
 2 Gauguin         Cocktail Classics                     2 Passion Fruit Syrup
 3 Gauguin         Cocktail Classics                     3 Lemon Juice
 4 Gauguin         Cocktail Classics                     4 Lime Juice
 5 Fort Lauderdale Cocktail Classics                     1 Light Rum
 6 Fort Lauderdale Cocktail Classics                     2 Sweet Vermouth
 7 Fort Lauderdale Cocktail Classics                     3 Juice of Orange
 8 Fort Lauderdale Cocktail Classics                     4 Juice of a Lime
```

```
 9 Apple Pie       Cordials and Liqueurs                      1 Apple schnapps
10 Apple Pie       Cordials and Liqueurs                      2 Cinnamon schnapps
```

Problem 4: Some of the ingredients in the ingredient list have different names, but are nearly the same thing. An example of such a pair is `Lemon Juice` and `Juice of a lemon`, which are considered different ingredients in this dataset, but which perhaps should be treated as the same depending on the analysis you are doing. Make a list of the ingredients appearing in the ingredient list ranked by how commonly they occur along with the number of occurrences, and print the first 10 elements of the list here. Then check more ingredients (I suggest looking at more ingredients and even sorting them alphabetically using `arrange(asc(ingredient)))` and see if you can spot pairs of ingredients that are similar but have different names. Use `if_else`(click here for if_else) or `case_when` in combination with `mutate` to make it so that the pairs of ingredients you found have the same name. You don't have to find all pairs, but find at least 5 pairs of ingredients to rename. Because the purpose of this renaming is to facilitate a hypothetical future analysis, you can choose your own criteria for similarity as long as it is somewhat justifiable.

```r
ingredients_n <- mixed_drinks_long|>
  group_by(Ingredient)|>
  summarise(Count = n())|>
              arrange(desc(Count))
print(ingredients_n)|>
  slice_head(n=10)
```

```
# A tibble: 673 x 2
   Ingredient        Count
   <chr>             <int>
 1 Gin                 176
 2 Fresh lemon juice   138
 3 Simple Syrup        115
 4 Light Rum           114
 5 Vodka               114
 6 Dry Vermouth        107
 7 Fresh Lime Juice    107
 8 Triple Sec          107
 9 Powdered Sugar       90
10 Grenadine            85
# i 663 more rows
```

```
# A tibble: 10 x 2
   Ingredient        Count
```

```
   <chr>              <int>
 1 Gin                  176
 2 Fresh lemon juice    138
 3 Simple Syrup         115
 4 Light Rum            114
 5 Vodka                114
 6 Dry Vermouth         107
 7 Fresh Lime Juice     107
 8 Triple Sec           107
 9 Powdered Sugar        90
10 Grenadine             85
```

```
mixed_drinks_long <- mixed_drinks_long |>
  mutate(Ingredient = case_when(
    Ingredient %in% c(
      "Absinthe", "Absinthe or pastis", "Absinthe Substitute")
    ~ "Absinthe or pastis",
    Ingredient %in% c(
      "Sugar Syrup", "Simple Syrup")
    ~ "Simple Syrup",
    Ingredient %in% c(
      "Fresh Lime Juice", "Lime Juice",
      "Fresh Lime Juice, lime wheel",
      "Fresh lime juice (reserve 1/2 lime shell for garnish)")
    ~ "Lime Juice",
    Ingredient %in% c(
      "Peychaud's Bitters", "Peychaud Bitters")
    ~ "Peychaud Bitters",
    Ingredient %in% c(
      "port", "Port")
    ~ "Port",
    TRUE ~ Ingredient
  ))
```

Notice that there are some ingredients that appear to be two or more ingredients strung together with commas. These would be candidates for more cleaning though this exercise doesn't ask you to fix them.

Problem 5: Some operations are easier to do on `wide` data rather than `tall` data. Find the 10 most common pairs of ingredients occurring in the top 2 ingredients in a recipe. It is much easier to do this with a `wide` dataset, so use `pivot_wider` to change the data so that each row contains all of the ingredients of a single cocktail, just like in the format of the original data-set. Then use `count` on the `1` and `2` columns to determine the most common pairs (see chapter 3 for a refresher on `count`).

```
mixed_drinks_wide <- mixed_drinks_long |>
  pivot_wider(
    names_from = `Ingredient Rank`,
    names_prefix = "Rank_",
    values_from = Ingredient
  )
print(mixed_drinks_wide)
```

```
# A tibble: 990 x 8
   name                 category       Rank_1 Rank_2 Rank_3 Rank_4 Rank_5 Rank_6
   <chr>                <chr>          <chr>  <chr>  <chr>  <chr>  <chr>  <chr>
 1 Gauguin              Cocktail Clas~ Light~ Passi~ Lemon~ Lime ~ <NA>   <NA>
 2 Fort Lauderdale      Cocktail Clas~ Light~ Sweet~ Juice~ Juice~ <NA>   <NA>
 3 Apple Pie            Cordials and ~ Apple~ Cinna~ Apple~ <NA>   <NA>   <NA>
 4 Cuban Cocktail No. 1 Cocktail Clas~ Juice~ Powde~ Light~ <NA>   <NA>   <NA>
 5 Cool Carlos          Cocktail Clas~ Dark ~ Cranb~ Pinea~ Orang~ Sour ~ <NA>
 6 John Collins         Whiskies       Bourb~ Fresh~ Simpl~ Soda ~ Orang~ <NA>
 7 Cherry Rum           Cocktail Clas~ Light~ cherr~ Light~ <NA>   <NA>   <NA>
 8 Casa Blanca          Cocktail Clas~ Light~ Lime ~ Tripl~ Maras~ <NA>   <NA>
 9 Caribbean Champagne  Cocktail Clas~ Light~ Creme~ Chill~ <NA>   <NA>   <NA>
10 Amber Amour          Cordials and ~ Amare~ Fresh~ Simpl~ Soda ~ <NA>   <NA>
# i 980 more rows
```

Note: You may be interested to read about the **widyr** package here: widyr page. It is designed to solve problems like this one and uses internal pivot steps to accomplish it so that the final result is tidy. I'm actually unaware of any easy ways of solving problem 5 without pivoting to a wide dataset.

```
mixed_dynamic_dou<- mixed_drinks_wide|>
  pairwise_count(Rank_1,Rank_2, upper=FALSE )|>
  arrange(desc(n))
print(mixed_dynamic_dou)
```

```
# A tibble: 1,647 x 3
   item1           item2               n
   <chr>           <chr>           <dbl>
 1 Vodka           Blanco tequila     18
 2 Light Rum       Vodka              15
 3 Light Rum       Blanco tequila     14
 4 Bourbon whiskey Vodka              13
 5 Light Rum       Gin                11
 6 Bourbon whiskey Blanco tequila     11
 7 Brandy          Vodka              10
 8 Light Rum       Bourbon whiskey     8
 9 Gin             Vodka               8
10 Light Rum       Brandy              7
# i 1,637 more rows
```

##Wide count method

```
mixed_drinks_wide |>
  group_by(Rank_1, Rank_2) |>
  summarise(n = n(), .groups = "drop") |>
  arrange(desc(n))|>
  print()
```

```
# A tibble: 699 x 3
   Rank_1           Rank_2                n
   <chr>            <chr>             <int>
 1 Gin              Dry Vermouth         23
 2 Juice of a Lemon Powdered Sugar       23
 3 Light Rum        Lime Juice           14
 4 Whole Egg        Powdered Sugar       13
 5 Gin              Triple Sec            9
 6 Bourbon whiskey  Fresh lemon juice     8
 7 Brandy           Sweet Vermouth        7
 8 Gin              Sweet Vermouth        7
 9 Light Rum        Pineapple Juice       7
10 Light Rum        Sweet Vermouth        7
# i 689 more rows
```

##pairwise_count method:
I only care about Rank_1 and Rank_2 so i will filter for any data with those ranks

```
filtered_ranks<- mixed_drinks_long |>
  filter(`Ingredient Rank` == 1 | `Ingredient Rank` == 2)
```

Now I can use that to pass it into pairwise_count() where were asking for the name of the most common

Ingredient pairs.

```
pairwise_counts <- pairwise_count(
  filtered_ranks,
  item = Ingredient,
  feature = name,
  upper= FALSE
)|>
  arrange(desc(n)) |>
  print()
```

```
# A tibble: 662 x 3
   item1           item2                n
   <chr>           <chr>            <dbl>
 1 Gin             Dry Vermouth        28
 2 Powdered Sugar  Juice of a Lemon    25
 3 Light Rum       Lime Juice          14
 4 Powdered Sugar  Whole Egg           14
 5 Sweet Vermouth  Gin                  9
 6 Sweet Vermouth  Dry Vermouth         9
 7 Gin             Triple Sec           9
 8 Bourbon whiskey Fresh lemon juice    8
 9 Light Rum       Sweet Vermouth       7
10 Light Rum       Brandy               7
# i 652 more rows
```

Oh oh not what I was expecting….

Remarks:

I believe I am observing discrepancies in the counts between the two methods due to differences in how pairs of ingredients are considered.

In the pairwise counting method, the function counts pairs regardless of the order of ingredients. For example, if we have a pair of ingredients "Gin" and "Juice," (Yea I am a 90's kid #SNOOP)the function will count both "Gin, Juice" and "Juice, Gin" as the same pair. This results in a combined count of 13 if "Gin, Juice" appears 10 times and "Juice, Gin" appears 3 times.

In contrast, when using the wide-format counting method, the pairs are counted in a specific order, as grouped by `Rank_1` and `Rank_2`. Therefore, only pairs where "Gin" appears in `Rank_1` and "Juice" appears in `Rank_2` are counted. This approach results in a total count of 10 for the pair "Gin, Juice," not accounting for the reverse order.

Hence, the discrepancy arises because the pairwise method counts regardless of order, while the wide-format method strictly follows the order of ranks, leading to different total counts for the same pairs.
To test this I can make sure that ingredients in rank_1 and rank _2 are counted regardless which of these two ranks it shows up in.

```
normalized_data <- mixed_drinks_long |>
  filter(`Ingredient Rank` == 1 | `Ingredient Rank` == 2) |>
  pivot_wider(names_from = `Ingredient Rank`, values_from = Ingredient) |>
  mutate(
```

```r
    pair = pmap_chr(list(`1`, `2`), ~ {
      ingredients <- c(..1, ..2)
      paste(sort(ingredients), collapse = ", ")
    })
  ) |>
  count(pair) |>
  arrange(desc(n))

print(normalized_data)
```

```
# A tibble: 668 x 2
   pair                                n
   <chr>                           <int>
 1 Dry Vermouth, Gin                  28
 2 Juice of a Lemon, Powdered Sugar   25
 3 Light Rum, Lime Juice              14
 4 Powdered Sugar, Whole Egg          14
 5 Dry Vermouth, Sweet Vermouth        9
 6 Gin, Sweet Vermouth                 9
 7 Gin, Triple Sec                     9
 8 Bourbon whiskey, Fresh lemon juice  8
 9 Brandy, Light Rum                   7
10 Brandy, Sweet Vermouth              7
# i 658 more rows
```

Did this make sense, or did I just spend 2 hours working on something that doesn't make sense? Please let me know.It was fun though. :)