Important codes

Humphrey

2022-07-07

Learn group_by from the dplyr package

- `group_by()` allows us to group on a categorical column.
- I `count()` the presence of automobile classes by manufacturer.
- Then `ungroup()` to remove any leftover groups.
- And now have counts by manufacturer + class.

```
library(dplyr)
library(ggplot2)

mpg %>%
  group_by(manufacturer) %>%
  count(class, name="n") %>%
  ungroup()
```

```
## # A tibble: 32 x 3
##    manufacturer class          n
##    <chr>        <chr>      <int>
##  1 audi         compact       15
##  2 audi         midsize        3
##  3 chevrolet    2seater        5
##  4 chevrolet    midsize        5
##  5 chevrolet    suv            9
##  6 dodge        minivan       11
##  7 dodge        pickup        19
##  8 dodge        suv            7
##  9 ford         pickup         7
## 10 ford         subcompact     9
## # ... with 22 more rows
```

But you can simplify with just `count()`

```
mpg %>%
  count(manufacturer, class, name = "n")
```

```
## # A tibble: 32 x 3
##    manufacturer class          n
##    <chr>        <chr>      <int>
##  1 audi         compact       15
##  2 audi         midsize        3
##  3 chevrolet    2seater        5
##  4 chevrolet    midsize        5
##   5 chevrolet    suv            9
##   6 dodge        minivan       11
##   7 dodge        pickup        19
##   8 dodge        suv            7
##   9 ford         pickup         7
##  10 ford         subcompact     9
## # ... with 22 more rows
```

Learn `pivot_wider()` from `tidyr`

- `pivot_wider()` is a great function (replaces `spread()` which I also enjoy using)
- And we can now make pivot tables that shift our values from the column class into new columns.
- Super valuable for creating those pivot table reports.

```
library(tidyr)
mpg_pivot_table1 <- mpg %>%
  group_by(manufacturer) %>%
  count(class, name = "n") %>%
  ungroup() %>%
  pivot_wider(
    names_from = class,
    values_from = n,
    values_fill = 0
  )
mpg_pivot_table1
```

```
## # A tibble: 15 x 8
##    manufacturer compact midsize `2seater`   suv minivan pickup subcompact
##    <chr>          <int>   <int>     <int> <int>   <int>  <int>      <int>
##  1 audi              15       3         0     0       0      0          0
##  2 chevrolet          0       5         5     9       0      0          0
##  3 dodge              0       0         0     7      11     19          0
##  4 ford               0       0         0     9       0      7          9
##  5 honda              0       0         0     0       0      0          9
##  6 hyundai            0       7         0     0       0      0          7
##  7 jeep               0       0         0     8       0      0          0
##  8 land rover         0       0         0     4       0      0          0
##  9 lincoln            0       0         0     3       0      0          0
## 10 mercury            0       0         0     4       0      0          0
## 11 nissan             2       7         0     4       0      0          0
## 12 pontiac            0       5         0     0       0      0          0
## 13 subaru             4       0         0     6       0      0          4
## 14 toyota            12       7         0     8       0      7          0
## 15 volkswagen        14       7         0     0       0      0          6
```

Learn `pivot_longer()` from `tidyr`

- Pivot longer reverses the operation by taking a wide data frame and pivoting it into a long "tidy" format.
- The "long format" is super important for making visualizations with ggplot2 & doing iteration with `purrr::map()`.

```
mpg_long_summary_table <- mpg_pivot_table1 %>%
  pivot_longer(
    cols = compact:subcompact,
    names_to = "class",
    values_to = "value"
  )
mpg_long_summary_table
```

```
## # A tibble: 105 x 3
##    manufacturer class      value
##    <chr>        <chr>      <int>
##  1 audi         compact       15
##  2 audi         midsize        3
##  3 audi         2seater        0
##  4 audi         suv            0
##  5 audi         minivan        0
##  6 audi         pickup         0
##  7 audi         subcompact     0
##  8 chevrolet    compact        0
##  9 chevrolet    midsize        5
## 10 chevrolet    2seater        5
## # ... with 95 more rows
```

Learn how to combine `group_by()` + `summarise()` + `across()`

- `group_by()` for setting up groups
- `summarise()` for applying summary functions to each group
- `across()` for applying multiple functions to one or more columns

```
mpg %>%
  group_by(class) %>%
  summarise(
    across(cty, .fns = list(mean=mean, stdev=sd)), .groups = "drop"
  )
```

```
## # A tibble: 7 x 3
##   class      cty_mean cty_stdev
##   <chr>         <dbl>     <dbl>
## 1 2seater        15.4     0.548
## 2 compact        20.1     3.39
## 3 midsize        18.8     1.95
## 4 minivan        15.8     1.83
## 5 pickup         13       2.05
## 6 subcompact     20.4     4.60
## 7 suv            13.5     2.42
```

Learn `relocate()` from dplyr

- `relocate()` allows us to have complete control on how we move columns.
- I can still use `select()` for this, but when I absolutely need fine control, I switch over to `relocate()`

```
mpg %>%
  relocate(where(is.character), .after = last_col())
```

```
## # A tibble: 234 x 11
##    displ  year   cyl   cty   hwy manufacturer model      trans drv   fl    class
##    <dbl> <int> <int> <int> <int> <chr>        <chr>      <chr> <chr> <chr> <chr>
##  1   1.8  1999     4    18    29 audi         a4         auto~ f     p     comp~
##  2   1.8  1999     4    21    29 audi         a4         manu~ f     p     comp~
##  3   2    2008     4    20    31 audi         a4         manu~ f     p     comp~
##  4   2    2008     4    21    30 audi         a4         auto~ f     p     comp~
##  5   2.8  1999     6    16    26 audi         a4         auto~ f     p     comp~
##  6   2.8  1999     6    18    26 audi         a4         manu~ f     p     comp~
##  7   3.1  2008     6    18    27 audi         a4         auto~ f     p     comp~
##  8   1.8  1999     4    18    26 audi         a4 quattro manu~ 4     p     comp~
##  9   1.8  1999     4    16    25 audi         a4 quattro auto~ 4     p     comp~
## 10   2    2008     4    20    28 audi         a4 quattro manu~ 4     p     comp~
## # ... with 224 more rows
```

Learn `group_split()` from dplyr

- I prefer group_split() for splitting data frames that are grouped into a list containing sub-data frames as elements.

  You'll see why in the next tip.

Note *(Pro-Tip)*: Make sure to convert your grouping column into a factor first, which preserves the order

```
df_split <- mpg %>%
  mutate(manufacturer=as.factor(manufacturer)) %>%
  group_by(manufacturer) %>%
  group_split()
df_split[[1]]
```

```
## # A tibble: 18 x 11
##    manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
##    <fct>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
##  2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
##  3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
##  4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
##  5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
##  6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
##  7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
##  8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
##  9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
## 10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
## 11 audi         a4 quattro   2    2008     4 auto~ 4        19    27 p     comp~
## 12 audi         a4 quattro   2.8  1999     6 auto~ 4        15    25 p     comp~
## 13 audi         a4 quattro   2.8  1999     6 manu~ 4        17    25 p     comp~
## 14 audi         a4 quattro   3.1  2008     6 auto~ 4        17    25 p     comp~
## 15 audi         a4 quattro   3.1  2008     6 manu~ 4        15    25 p     comp~
## 16 audi         a6 quattro   2.8  1999     6 auto~ 4        15    24 p     mids~
## 17 audi         a6 quattro   3.1  2008     6 auto~ 4        17    25 p     mids~
## 18 audi         a6 quattro   4.2  2008     8 auto~ 4        16    23 p     mids~
```

We can now combine `group_split()` with `map()`

- This is a cool example where I'm splitting the data frame by manufacturer then applying a linear regression model to each data frame.
- I get a linear reg model for each car manufacturer.

- print out the linear reg model for the first 2 car manufacturers

```
library(purrr)
#library(kableExtra)
linear_reg_models <- mpg %>%
  mutate(manufacturer=as.factor(manufacturer)) %>%
  group_by(manufacturer) %>%
  group_split() %>%

  map(.f=function(df){
  lm(hwy~cty, data = df)
}
  )
linear_reg_models[c(1:2)]
```

```
## [[1]]
##
## Call:
## lm(formula = hwy ~ cty, data = df)
##
## Coefficients:
## (Intercept)          cty
##      9.9405       0.9371
##
##
## [[2]]
##
## Call:
## lm(formula = hwy ~ cty, data = df)
##
## Coefficients:
## (Intercept)          cty
##      -2.456        1.623
```