Session 5: Aggregating and reshaping Data

Wali Reheman

2024-09-30

Introduction

The `summarize()` function from the `dplyr` package is a powerful tool for creating summary statistics of your data. It allows you to collapse a dataset to a single row or a summary for each group of observations. In this tutorial, we'll explore the basic and advanced uses of `summarize()`, as well as ways to reshape data.

```
#install.packages("gapminder")
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(gapminder)
```

```
data("gapminder")
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

Basic Usage of `summarize()`

The basic syntax of `summarize()` is straightforward. You provide it with a dataset and specify the summary statistics you want to compute.

```r
gapminder %>%
  summarize(global_avg_lifeExp = mean(lifeExp,na.rm = TRUE),
            n= n())
```

```
## # A tibble: 1 x 2
##   global_avg_lifeExp     n
##                <dbl> <int>
## 1               59.5  1704
```

Explanation of `na.rm = TRUE`

When working with data in R, it's common to encounter missing values (`NA`s) in datasets. Most summarization functions in R, such as `mean()`, `sum()`, and `median()`, will return `NA` if any of the values being summarized are missing, which may distort the results.

To handle this, many R functions include an argument called `na.rm`. The argument stands for "remove NAs" and is a logical value (`TRUE` or `FALSE`). When set to `TRUE`, the function ignores any `NA` values and proceeds with the calculation using only the non-missing values.

In our case today, we know there is no NA in the data so I omitted `na.rm = TRUE`

Grouped Summaries with `group_by()`

Often, you want to compute summaries for subgroups within your data. This is where `group_by()` comes into play.

```r
gapminder %>%
  group_by(country) %>%
  summarize(avg_lifeExp = mean(lifeExp),
            n=n())
```

```
## # A tibble: 142 x 3
##    country     avg_lifeExp     n
##    <fct>             <dbl> <int>
##  1 Afghanistan        37.5    12
##  2 Albania            68.4    12
##  3 Algeria            59.0    12
##  4 Angola             37.9    12
##  5 Argentina          69.1    12
##  6 Australia          74.7    12
##  7 Austria            73.1    12
##  8 Bahrain            65.6    12
##  9 Bangladesh         49.8    12
## 10 Belgium            73.6    12
## # i 132 more rows
```

Calculate the total population growth for each country over the years (1952-2007).

```r
# Example: Summarizing Population Growth
population_growth <- gapminder %>%
  group_by(country) %>%
  summarize(
    from = first(year),
    pop1952 = first(pop),
    to = last(year),
    pop2007 = last(pop),
    pop_growth = last(pop) - first(pop))

head(population_growth)
```

```
## # A tibble: 6 x 6
##   country      from  pop1952    to  pop2007 pop_growth
##   <fct>       <int>    <int> <int>    <int>      <int>
## 1 Afghanistan  1952  8425333  2007 31889923   23464590
## 2 Albania      1952  1282697  2007  3600523    2317826
## 3 Algeria      1952  9279525  2007 33333216   24053691
## 4 Angola       1952  4232095  2007 12420476    8188381
## 5 Argentina    1952 17876956  2007 40301927   22424971
## 6 Australia    1952  8691212  2007 20434176   11742964
```

Creating Cross-Sectional Data from Longitudinal Data

By summarizing longitudinal data, you can create new cross-sectional datasets for further analysis.

Create a cross-sectional dataset that includes the average life expectancy, average GDP per capital and population growth for each continent.

```
cross_sectional_data <- gapminder %>%
  group_by(continent) %>%
  summarize(
    avg_lifeExp = mean(lifeExp),
    avg_gdpPercap = median(gdpPercap),
    continent_pop = sum(pop)
  )

head(cross_sectional_data)
```

```
## # A tibble: 5 x 4
##   continent avg_lifeExp avg_gdpPercap continent_pop
##   <fct>           <dbl>         <dbl>         <dbl>
## 1 Africa           48.9         1192.    6187585961
## 2 Americas         64.7         5466.    7351438499
## 3 Asia             60.1         2647.   30507333901
## 4 Europe           71.9        12082.    6181115304
## 5 Oceania          74.3        17983.     212992136
```

Why Summarizing Longitudinal Data to Cross-Sectional Data Could be Useful

Longitudinal data tracks the same subjects (e.g., countries, individuals) across multiple time points. While this is useful for analyzing trends over time, sometimes it's necessary to condense the data into a **cross-sectional format**, where each observation is represented by a single row. Cross-sectional data represents the "snapshot" of each entity at a given moment or an aggregation over time, and it's often used for comparative or overview analyses.

Benefits of Summarizing Longitudinal Data:

1. **Simplification**: Summarizing longitudinal data into cross-sectional form simplifies the dataset, making it easier to analyze, visualize, or compare.
2. **Comparative Analysis**: By reducing data over time into key metrics (like averages, sums, or differences), we can compare entities (e.g., countries, individuals) in a more direct manner.
3. **Data Reduction**: Summarizing data reduces the number of rows and complexity, which can be helpful when analyzing or visualizing large datasets.

Advanced Usage

Summarizing with Multiple Grouping Variables

You can summarize data using multiple grouping variables to get more granular insights.

```r
#Example: Average Life Expectancy ect by Continent and Year
by_continent_year <- gapminder %>%
  group_by(continent, year) %>%
  summarize(
    avg_lifeExp = mean(lifeExp),
    avg_gdpPercap = mean(gdpPercap),
    continent_pop = sum(pop))
```

```
## ‘summarise()‘ has grouped output by ’continent’. You can override using the
## ‘.groups‘ argument.
```

```r
head(by_continent_year)
```

```
## # A tibble: 6 x 5
## # Groups:   continent [1]
##   continent  year avg_lifeExp avg_gdpPercap continent_pop
##   <fct>     <int>       <dbl>         <dbl>         <dbl>
## 1 Africa     1952        39.1         1253.     237640501
## 2 Africa     1957        41.3         1385.     264837738
## 3 Africa     1962        43.3         1598.     296516865
## 4 Africa     1967        45.3         2050.     335289489
## 5 Africa     1972        47.5         2340.     379879541
## 6 Africa     1977        49.6         2586.     433061021
```

Counts and proportions of logical values: `sum(x > 10)`, `mean(y == 0)`. When used with numeric functions, `TRUE` is converted to 1 and `FALSE` to 0. This makes `sum()` and `mean()` very useful: `sum(x)` gives the number of `TRUE`s in `x`, and `mean(x)` gives the proportion.

```r
gapminder %>%
  group_by(continent,year) %>%
  summarize(
    prop_1000 = mean(gdpPercap<1000)*100
  )
```

```
## ‘summarise()‘ has grouped output by ’continent’. You can override using the
## ‘.groups‘ argument.
```

```
## # A tibble: 60 x 3
## # Groups:   continent [5]
##    continent  year prop_1000
##    <fct>     <int>     <dbl>
##  1 Africa     1952      50
##  2 Africa     1957      48.1
##  3 Africa     1962      42.3
##  4 Africa     1967      34.6
##  5 Africa     1972      36.5
##  6 Africa     1977      38.5
##  7 Africa     1982      42.3
##  8 Africa     1987      42.3
##  9 Africa     1992      40.4
## 10 Africa     1997      42.3
## # i 50 more rows
```

Merging Summaries with Original Data

You can merge the summarized data back with the original dataset for comparative analysis.

```r
# Example: Merging Average Life Expectancy with Original Data
gapminder_with_summary <- gapminder %>%
  left_join(by_continent_year, by = c("continent","year"))

head(gapminder_with_summary)
```

```
## # A tibble: 6 x 9
##   country     continent  year lifeExp    pop gdpPercap avg_lifeExp avg_gdpPercap
##   <fct>       <fct>     <int>   <dbl>  <int>     <dbl>       <dbl>         <dbl>
## 1 Afghanistan Asia       1952    28.8 8.43e6      779.        46.3         5195.
## 2 Afghanistan Asia       1957    30.3 9.24e6      821.        49.3         5788.
## 3 Afghanistan Asia       1962    32.0 1.03e7      853.        51.6         5729.
## 4 Afghanistan Asia       1967    34.0 1.15e7      836.        54.7         5971.
## 5 Afghanistan Asia       1972    36.1 1.31e7      740.        57.3         8187.
## 6 Afghanistan Asia       1977    38.4 1.49e7      786.        59.6         7791.
## # i 1 more variable: continent_pop <dbl>
```

* Working with `window` Functions

```r
gapminder_with_summary<-gapminder_with_summary%>%
  mutate(lag_avg_GPDpc = lag(avg_gdpPercap))

head(gapminder_with_summary)
```

```
## # A tibble: 6 x 10
##   country     continent  year lifeExp    pop gdpPercap avg_lifeExp avg_gdpPercap
##   <fct>       <fct>     <int>   <dbl>  <int>     <dbl>       <dbl>         <dbl>
## 1 Afghanistan Asia       1952    28.8 8.43e6      779.        46.3         5195.
## 2 Afghanistan Asia       1957    30.3 9.24e6      821.        49.3         5788.
## 3 Afghanistan Asia       1962    32.0 1.03e7      853.        51.6         5729.
## 4 Afghanistan Asia       1967    34.0 1.15e7      836.        54.7         5971.
## 5 Afghanistan Asia       1972    36.1 1.31e7      740.        57.3         8187.
## 6 Afghanistan Asia       1977    38.4 1.49e7      786.        59.6         7791.
## # i 2 more variables: continent_pop <dbl>, lag_avg_GPDpc <dbl>
```

** Transfer data to wide

```r
by_continent_year_wide <- by_continent_year %>%
  pivot_wider(names_from = year, values_from = c(avg_lifeExp,avg_gdpPercap,continent_pop))

head(by_continent_year_wide)
```

```
## # A tibble: 5 x 37
## # Groups:   continent [5]
##   continent avg_lifeExp_1952 avg_lifeExp_1957 avg_lifeExp_1962 avg_lifeExp_1967
##   <fct>                <dbl>            <dbl>            <dbl>            <dbl>
## 1 Africa                39.1             41.3             43.3             45.3
## 2 Americas              53.3             56.0             58.4             60.4
## 3 Asia                  46.3             49.3             51.6             54.7
## 4 Europe                64.4             66.7             68.5             69.7
## 5 Oceania               69.3             70.3             71.1             71.3
## # i 32 more variables: avg_lifeExp_1972 <dbl>, avg_lifeExp_1977 <dbl>,
## #   avg_lifeExp_1982 <dbl>, avg_lifeExp_1987 <dbl>, avg_lifeExp_1992 <dbl>,
## #   avg_lifeExp_1997 <dbl>, avg_lifeExp_2002 <dbl>, avg_lifeExp_2007 <dbl>,
## #   avg_gdpPercap_1952 <dbl>, avg_gdpPercap_1957 <dbl>,
## #   avg_gdpPercap_1962 <dbl>, avg_gdpPercap_1967 <dbl>,
## #   avg_gdpPercap_1972 <dbl>, avg_gdpPercap_1977 <dbl>,
## #   avg_gdpPercap_1982 <dbl>, avg_gdpPercap_1987 <dbl>, ...
```

Using across() for Summarizing Multiple Columns

Demonstrate how to apply summary functions across multiple columns using the `across()` helper.

```r
# Example: Calculate the mean of multiple numeric columns
gapminder %>%
  group_by(continent) %>%
  summarize(across(c(lifeExp, gdpPercap), mean))
```

```
## # A tibble: 5 x 3
##   continent lifeExp gdpPercap
##   <fct>       <dbl>     <dbl>
## 1 Africa       48.9     2194.
## 2 Americas     64.7     7136.
## 3 Asia         60.1     7902.
## 4 Europe       71.9    14469.
## 5 Oceania      74.3    18622.
```

Applying Multiple Functions with across()

Apply different functions to different columns within a single summarize() call.

```
# Example: Apply different functions to different columns
gapminder %>%
  group_by(continent) %>%
  summarize(
    across(c(lifeExp,gdpPercap), mean, .names = "avg_{col}"),
    across(c(lifeExp,gdpPercap), median, .names = "median_{col}")
  )
```

```
## # A tibble: 5 x 5
##   continent avg_lifeExp avg_gdpPercap median_lifeExp median_gdpPercap
##   <fct>           <dbl>         <dbl>          <dbl>            <dbl>
## 1 Africa           48.9         2194.           47.8            1192.
## 2 Americas         64.7         7136.           67.0            5466.
## 3 Asia             60.1         7902.           61.8            2647.
## 4 Europe           71.9        14469.           72.2           12082.
## 5 Oceania          74.3        18622.           73.7           17983.
```

Bonus: Mapping Your Data

Make sure you have the necessary packages installed:

```
#install.packages("ggplot2")
#install.packages("rnaturalearth")
#install.packages("rnaturalearthdata")
```

```
library(tidyverse)
library(gapminder)
library(rnaturalearth)
library(rnaturalearthdata)
```

```
##
## Attaching package: 'rnaturalearthdata'

## The following object is masked from 'package:rnaturalearth':
##
##     countries110
```

```r
library(ggplot2)
```

we will summarize the gapminder data by **country** to calculate the average life expectancy for each country.

```r
# Summarizing data by continent
cross_sectional_data <- gapminder %>%
  group_by(country) %>%
  summarize(
    avg_lifeExp = mean(lifeExp, na.rm = TRUE)
  )
```

Use the rnaturalearth package to get the world map data for countries.

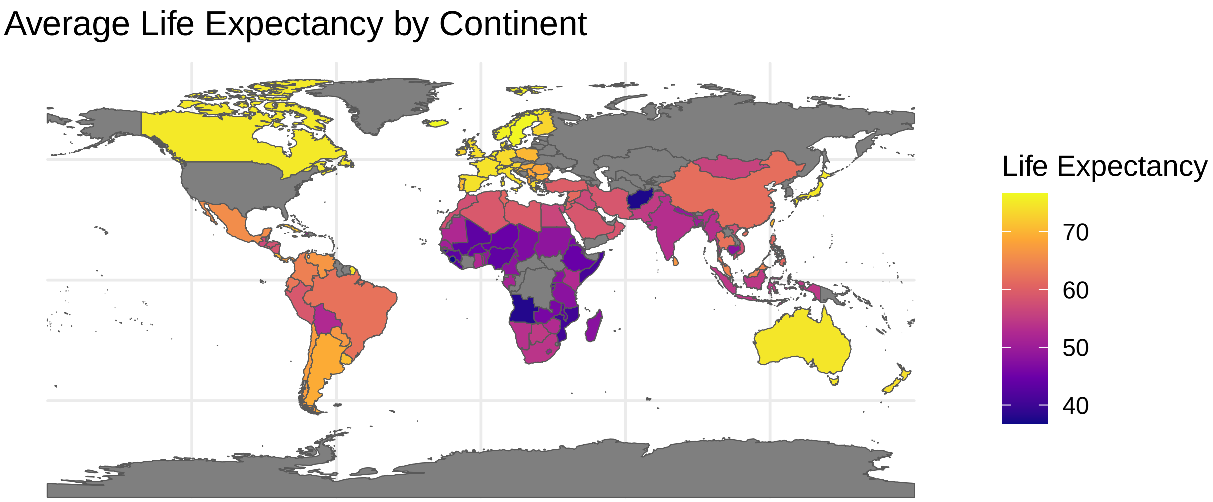
```r
# Getting world map data
world_map <- ne_countries(scale = "medium", returnclass = "sf")
```

Next, we will merge the country_data (average life expectancy) with the world_map dataset. The world_map dataset has country names, so we will use left_join() to merge them based on the country name.

```r
# Merging the country-level life expectancy with the world map
world_map_data <- world_map %>%
  left_join(cross_sectional_data, by = c("name" = "country"))
```

Now we can create the map using ggplot2. We will use geom_sf() to plot the map, and scale_fill_viridis_c() to color the countries based on life expectancy.

```r
# Plotting the map
ggplot(data = world_map_data)+
  geom_sf(aes(fill = avg_lifeExp)) +
  scale_fill_viridis_c(option = "plasma", na.value = "gray50") +
  labs(title = "Average Life Expectancy by Continent",
       fill = "Life Expectancy") +
  theme_minimal()
```



Average Life Expectancy by Continent