pivot\_lobsters

Lakshman

2022-12-02

# Pivot Tables with dplyr

create these tables using the group\_by and summarize functions from the dplyr package (part of the Tidyverse)

## attach libraries  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 0.3.5   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1   
## ✔ readr 2.1.3 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(readxl)  
library(here)

## here() starts at D:/New folder/R/WD/r-for-excel-data/r-workshop/r-workshop

library(skimr) # install.packages('skimr')  
library(kableExtra) # install.packages('kableExtra')

## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output  
## %in% : 'length(x) = 3 > 1' in coercion to 'logical(1)'

##   
## Attaching package: 'kableExtra'  
##   
## The following object is masked from 'package:dplyr':  
##   
## group\_rows

# Read in data

lobsters <- read\_xlsx(here("data/lobsters.xlsx"), skip=4)

use head(lobsters) # for top 6 rows

## explore data

skimr::skim(lobsters) # skim lets us look more at each variable

skimr::skim(lobsters) # skim lets us look more at each variable

Data summary

Name

lobsters

Number of rows

2893

Number of columns

7

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Column type frequency:

character

3

numeric

4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group variables

None

**Variable type: character**

skim\_variable

n\_missing

complete\_rate

min

max

empty

n\_unique

whitespace

date

0

1

6

7

0

28

0

site

0

1

4

4

0

5

0

replicate

0

1

1

1

0

4

0

**Variable type: numeric**

skim\_variable

n\_missing

complete\_rate

mean

sd

p0

p25

p50

p75

p100

hist

year

0

1

2014.70

1.19

2012

2014

2015

2016

2016

▂▂▃▇▆

month

0

1

8.04

0.19

8

8

8

8

9

▇▁▁▁▁

transect

0

1

3.72

2.30

1

2

3

5

9

▇▅▃▂▂

size\_mm

5

1

71.38

14.75

18

62

72

81

165

▁▇▆▁▁

group\_by() %>% summarize() In R, we can create the functionality of pivot tables with the same logic: we will tell R to group by something and then summarize by something

Take the data and then group by something and then summarize by something Syntax data %>% group\_by() %>% summarize()

The pipe operator %>% is a really critical feature of the dplyr package, originally created for the magrittr package. It lets us chain together steps of our data wrangling, enabling us to tell a clear story about our entire data analysis.

View(lobsters) shows up in your Console. View() (capital V) is the R function to view any variable in the viewer.

# group\_by one variable

group\_by() %>% summarize() with our lobsters data, just like we did in Excel. We will first group\_by year and then summarize by count, using the function n() (in the dplyr package). n() counts the number of times an observation shows up, and since this is uncounted data, this will count each row.

lobsters %>%  
group\_by(year) %>%  
 summarise(count\_by\_year=n())

## # A tibble: 5 × 2  
## year count\_by\_year  
## <dbl> <int>  
## 1 2012 231  
## 2 2013 243  
## 3 2014 510  
## 4 2015 1100  
## 5 2016 809

# group\_by multiple variables

lobsters %>%  
 group\_by(site, year) %>%  
 summarize(count\_by\_siteyear = n())

## `summarise()` has grouped output by 'site'. You can override using the  
## `.groups` argument.

## # A tibble: 25 × 3  
## # Groups: site [5]  
## site year count\_by\_siteyear  
## <chr> <dbl> <int>  
## 1 aque 2012 38  
## 2 aque 2013 32  
## 3 aque 2014 100  
## 4 aque 2015 83  
## 5 aque 2016 48  
## 6 carp 2012 78  
## 7 carp 2013 93  
## 8 carp 2014 79  
## 9 carp 2015 90  
## 10 carp 2016 231  
## # … with 15 more rows

# summarize multiple variables

Let’s also calculate the mean and standard deviation. First let’s use the mean() function to calculate the mean. We do this within the same summarize() function

lobsters %>%  
 group\_by(site, year) %>%  
 summarize(count\_by\_siteyear = n(),  
 mean\_size\_mm = mean(size\_mm)) # by count and mean

## `summarise()` has grouped output by 'site'. You can override using the  
## `.groups` argument.

## # A tibble: 25 × 4  
## # Groups: site [5]  
## site year count\_by\_siteyear mean\_size\_mm  
## <chr> <dbl> <int> <dbl>  
## 1 aque 2012 38 71   
## 2 aque 2013 32 72.1  
## 3 aque 2014 100 76.9  
## 4 aque 2015 83 68.5  
## 5 aque 2016 48 68.7  
## 6 carp 2012 78 74.4  
## 7 carp 2013 93 76.6  
## 8 carp 2014 79 NA   
## 9 carp 2015 90 70.7  
## 10 carp 2016 231 68.9  
## # … with 15 more rows

NA because one or more values in that year are NA. pass an argument na.rm=TRUE that says to remove NAs first before calculating the average. Then Calculate the standard deviation with the sd() function

lobsters %>%  
 group\_by(site, year) %>%  
 summarize(count\_by\_siteyear = n(),   
 mean\_size\_mm = mean(size\_mm, na.rm=TRUE), #na.rm to remove NA  
 sd\_size\_mm = sd(size\_mm, na.rm=TRUE)) #summarise by count, mean, sd

## `summarise()` has grouped output by 'site'. You can override using the  
## `.groups` argument.

## # A tibble: 25 × 5  
## # Groups: site [5]  
## site year count\_by\_siteyear mean\_size\_mm sd\_size\_mm  
## <chr> <dbl> <int> <dbl> <dbl>  
## 1 aque 2012 38 71 10.2   
## 2 aque 2013 32 72.1 12.3   
## 3 aque 2014 100 76.9 9.32  
## 4 aque 2015 83 68.5 12.6   
## 5 aque 2016 48 68.7 12.5   
## 6 carp 2012 78 74.4 14.6   
## 7 carp 2013 93 76.6 8.71  
## 8 carp 2014 79 79.1 8.57  
## 9 carp 2015 90 70.7 14.6   
## 10 carp 2016 231 68.9 12.5   
## # … with 15 more rows

Now we are at the point where we actually want to save this summary information as a variable so we can use it in further analyses and formatting.

So let’s add a variable assignment to that first line:

siteyear\_summary <- lobsters %>%  
 group\_by(site, year) %>%  
 summarize(count\_by\_siteyear = n(),   
 mean\_size\_mm = mean(size\_mm, na.rm=TRUE), #na.rm to remove NA  
 sd\_size\_mm = sd(size\_mm, na.rm=TRUE)) #summarise by count, mean, sd

## `summarise()` has grouped output by 'site'. You can override using the  
## `.groups` argument.

siteyear\_summary #inspect our new variable

## # A tibble: 25 × 5  
## # Groups: site [5]  
## site year count\_by\_siteyear mean\_size\_mm sd\_size\_mm  
## <chr> <dbl> <int> <dbl> <dbl>  
## 1 aque 2012 38 71 10.2   
## 2 aque 2013 32 72.1 12.3   
## 3 aque 2014 100 76.9 9.32  
## 4 aque 2015 83 68.5 12.6   
## 5 aque 2016 48 68.7 12.5   
## 6 carp 2012 78 74.4 14.6   
## 7 carp 2013 93 76.6 8.71  
## 8 carp 2014 79 79.1 8.57  
## 9 carp 2015 90 70.7 14.6   
## 10 carp 2016 231 68.9 12.5   
## # … with 15 more rows

# Table formatting with kable()

## make a table with our new variable  
siteyear\_summary %>%  
 kable()

site

year

count\_by\_siteyear

mean\_size\_mm

sd\_size\_mm

aque

2012

38

71.00000

10.150223

aque

2013

32

72.12500

12.262584

aque

2014

100

76.92000

9.321074

aque

2015

83

68.45783

12.555536

aque

2016

48

68.68750

12.510687

carp

2012

78

74.35897

14.616282

carp

2013

93

76.56989

8.709562

carp

2014

79

79.08974

8.569329

carp

2015

90

70.65556

14.646517

carp

2016

231

68.90476

12.470122

ivee

2012

26

66.07692

12.092719

ivee

2013

40

73.77500

7.640941

ivee

2014

132

76.02273

17.860984

ivee

2015

361

69.80332

17.470534

ivee

2016

193

71.61658

13.450454

mohk

2012

83

77.25301

10.587433

mohk

2013

15

71.86667

10.190098

mohk

2014

36

75.75000

10.038142

mohk

2015

296

59.19932

16.770357

mohk

2016

210

63.01286

11.875763

napl

2012

6

73.00000

11.747340

napl

2013

63

75.31746

12.989854

napl

2014

163

79.51572

9.556531

napl

2015

270

78.24074

12.438899

napl

2016

127

74.39370

10.732060

include median

siteyear\_summary <- lobsters %>%  
 group\_by(site, year) %>%  
 summarize(count\_by\_siteyear = n(),   
 mean\_size\_mm = mean(size\_mm, na.rm = TRUE),   
 sd\_size\_mm = sd(size\_mm, na.rm = TRUE),   
 median\_size\_mm = median(size\_mm, na.rm = TRUE))

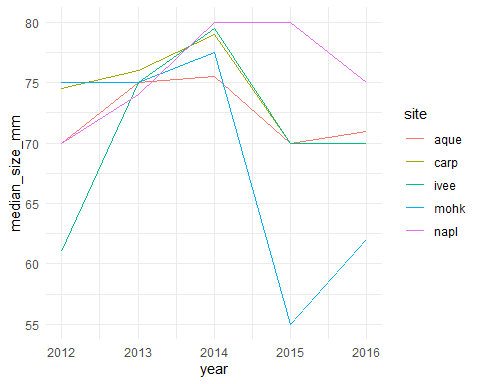
## `summarise()` has grouped output by 'site'. You can override using the  
## `.groups` argument.

siteyear\_summary

## # A tibble: 25 × 6  
## # Groups: site [5]  
## site year count\_by\_siteyear mean\_size\_mm sd\_size\_mm median\_size\_mm  
## <chr> <dbl> <int> <dbl> <dbl> <dbl>  
## 1 aque 2012 38 71 10.2 70   
## 2 aque 2013 32 72.1 12.3 75   
## 3 aque 2014 100 76.9 9.32 75.5  
## 4 aque 2015 83 68.5 12.6 70   
## 5 aque 2016 48 68.7 12.5 71   
## 6 carp 2012 78 74.4 14.6 74.5  
## 7 carp 2013 93 76.6 8.71 76   
## 8 carp 2014 79 79.1 8.57 79   
## 9 carp 2015 90 70.7 14.6 70   
## 10 carp 2016 231 68.9 12.5 70   
## # … with 15 more rows

ggplot function

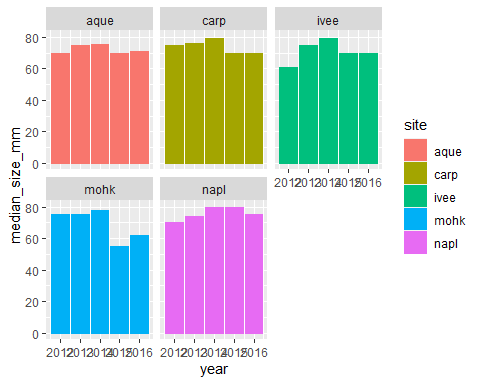
## a ggplot option:  
ggplot(data = siteyear\_summary, aes(x = year, y = median\_size\_mm, color = site)) +  
 geom\_line() +  
 theme\_minimal()



ggsave(here("figures", "lobsters-line.png")) # save image

## Saving 5 x 4 in image

## another option:  
ggplot(siteyear\_summary, aes(x = year, y = median\_size\_mm, fill = site, color =site)) +  
 geom\_col() +  
 facet\_wrap(~site)



ggsave(here("figures", "lobsters-col.png"))

## Saving 5 x 4 in image

#dplyr::count()

Now that we’ve spent time with group\_by %>% summarize, there is a shortcut if you only want to summarize by count. This is with a function called count(), and it will group\_by your selected variable, count, and then also ungroup.

lobsters %>%  
 count(site, year)

## # A tibble: 25 × 3  
## site year n  
## <chr> <dbl> <int>  
## 1 aque 2012 38  
## 2 aque 2013 32  
## 3 aque 2014 100  
## 4 aque 2015 83  
## 5 aque 2016 48  
## 6 carp 2012 78  
## 7 carp 2013 93  
## 8 carp 2014 79  
## 9 carp 2015 90  
## 10 carp 2016 231  
## # … with 15 more rows

## This is the same as:  
lobsters %>%  
 group\_by(site, year) %>%   
 summarize(n = n()) %>%  
 ungroup()

## `summarise()` has grouped output by 'site'. You can override using the  
## `.groups` argument.

## # A tibble: 25 × 3  
## site year n  
## <chr> <dbl> <int>  
## 1 aque 2012 38  
## 2 aque 2013 32  
## 3 aque 2014 100  
## 4 aque 2015 83  
## 5 aque 2016 48  
## 6 carp 2012 78  
## 7 carp 2013 93  
## 8 carp 2014 79  
## 9 carp 2015 90  
## 10 carp 2016 231  
## # … with 15 more rows

# Make new variable with mutate()

The sizes are in millimeters but let’s say it was important for them to be in meters. We can add a column with this calculation

lobsters %>%  
 mutate(size\_m = size\_mm / 1000)

## # A tibble: 2,893 × 8  
## year month date site transect replicate size\_mm size\_m  
## <dbl> <dbl> <chr> <chr> <dbl> <chr> <dbl> <dbl>  
## 1 2012 8 8/20/12 ivee 3 A 70 0.07   
## 2 2012 8 8/20/12 ivee 3 B 60 0.06   
## 3 2012 8 8/20/12 ivee 3 B 65 0.065  
## 4 2012 8 8/20/12 ivee 3 B 70 0.07   
## 5 2012 8 8/20/12 ivee 3 B 85 0.085  
## 6 2012 8 8/20/12 ivee 3 C 60 0.06   
## 7 2012 8 8/20/12 ivee 3 C 65 0.065  
## 8 2012 8 8/20/12 ivee 3 C 67 0.067  
## 9 2012 8 8/20/12 ivee 3 D 70 0.07   
## 10 2012 8 8/20/12 ivee 4 B 85 0.085  
## # … with 2,883 more rows

If we want to add a column that has the same value repeated, we can pass it just one value, either a number or a character string (in quotes). And let’s save this as a variable called lobsters\_detailed

lobsters\_detailed <- lobsters %>%  
 mutate(size\_m = size\_mm / 1000,   
 millenia = 2000,  
 observer = "Allison Horst")

# select()

To choose, retain, and move your data by columns To present this data finally with only columns for date, site, and size in meters

lobsters\_detailed %>%  
 select(date, site, size\_m)

## # A tibble: 2,893 × 3  
## date site size\_m  
## <chr> <chr> <dbl>  
## 1 8/20/12 ivee 0.07   
## 2 8/20/12 ivee 0.06   
## 3 8/20/12 ivee 0.065  
## 4 8/20/12 ivee 0.07   
## 5 8/20/12 ivee 0.085  
## 6 8/20/12 ivee 0.06   
## 7 8/20/12 ivee 0.065  
## 8 8/20/12 ivee 0.067  
## 9 8/20/12 ivee 0.07   
## 10 8/20/12 ivee 0.085  
## # … with 2,883 more rows