Cleaning table data 2

Raphael Eisenhofer

2022_04_21

Outline for today:

- 1. Quick recap of last lesson
- 2. Identifying problems in our table data
- 3. Fixing problems in our table data

1. Quick recap

A quick recap

- Tidyverse

Wickham et al., (2019). Welcome to the Tidyverse.
Journal of Open Source Software, 4(43), 1686,
https://doi.org/10.21105/joss.01686

A quick recap

- Tidyverse
- Pipes %>%


```r
vector <- c(5, 5, 5, 5, 5)
vector_sum <- sum(vector)
sqrt(vector_sum)


c(5, 5, 5, 5, 5) %>%
  sum() %>%
  sqrt()
```

Wickham et al., (2019). Welcome to the Tidyverse. *Journal of Open Source Software*, 4(43), 1686, https://doi.org/10.21105/joss.01686

A quick recap

- Tidyverse
- Pipes %>%
- Importing tables
  - read_xlsx() and read_delim()


```r
vector <- c(5, 5, 5, 5, 5)
vector_sum <- sum(vector)
sqrt(vector_sum)


c(5, 5, 5, 5, 5) %>%
  sum() %>%
  sqrt()
```

Wickham et al., (2019). Welcome to the Tidyverse. *Journal of Open Source Software*, 4(43), 1686, https://doi.org/10.21105/joss.01686

A quick recap

- Tidyverse
- Pipes %>%
- Importing tables
  - read_xlsx() and read_delim()
- Initial cleaning with janitor


```r
vector <- c(5, 5, 5, 5, 5)
vector_sum <- sum(vector)
sqrt(vector_sum)


c(5, 5, 5, 5, 5) %>%
  sum() %>%
  sqrt()
```

Wickham et al., (2019). Welcome to the Tidyverse. *Journal of Open Source Software*, 4(43), 1686, https://doi.org/10.21105/joss.01686


Real world data
iris &
mtcars

2. Identifying problems in table data

Finding issues with our table

```
> df
# A tibble: 150 x 7
   sample_number sepal_length sepal_width petal_length_percent_number petal_width date_collected      species
           <dbl>        <dbl> <chr>       <chr>                        <chr>       <dttm>              <chr>
 1             1          5.1 3.5         1.4                          0.2         2022-04-06 00:00:00 setosa
 2             2          4.9 3           1.4                          0.2         2022-04-07 00:00:00 setosa
 3             3          4.7 3.2         1.3                          0.2         2022-04-09 00:00:00 setosa
 4             4          4.6 3.1         1.5                          0.2         2022-04-10 00:00:00 setosa
 5             5          5   3.6         1.4                          0.2         2022-04-11 00:00:00 setosa
 6             6          5.4 3.9         1.7                          0.4         2022-04-12 00:00:00 setosas
 7             7          4.6 3.4         1.4                          0.3         2022-04-13 00:00:00 Setosa
 8             8          5   3.4         1.5                          0.2         2022-04-14 00:00:00 setosa
 9             9          4.4 2.9         NA                           0.2         2022-04-15 00:00:00 Setosa
10            10          4.9 na          1.5                          N/A         2022-04-16 00:00:00 setosa
# ... with 140 more rows
```

Finding issues with our table

```
> df
# A tibble: 150 x 7
   sample_number sepal_length sepal_width petal_length_percent_number petal_width date_collected      species
           <dbl>        <dbl> <chr>       <chr>                        <chr>       <dttm>              <chr>
 1             1          5.1 3.5         1.4                          0.2         2022-04-06 00:00:00 setosa
 2             2          4.9 3           1.4                          0.2         2022-04-07 00:00:00 setosa
 3             3          4.7 3.2         1.3                          0.2         2022-04-09 00:00:00 setosa
 4             4          4.6 3.1         1.5                          0.2         2022-04-10 00:00:00 setosa
 5             5          5   3.6         1.4                          0.2         2022-04-11 00:00:00 setosa
 6             6          5.4 3.9         1.7                          0.4         2022-04-12 00:00:00 setosas
 7             7          4.6 3.4         1.4                          0.3         2022-04-13 00:00:00 Setosa
 8             8          5   3.4         1.5                          0.2         2022-04-14 00:00:00 setosa
 9             9          4.4 2.9         NA                           0.2         2022-04-15 00:00:00 Setosa
10            10          4.9 na          1.5                          N/A         2022-04-16 00:00:00 setosa
# ... with 140 more rows
```

- Inconsistent NA, na, N/A values

10	10	4.9	na	1.5	N/A	2022-04-16	setosa
11	11	5.4	*NA*	1.5	0.2	2022-04-17	Setosa

What R wants!

Finding issues with our table

```
> df
# A tibble: 150 x 7
   sample_number sepal_length sepal_width petal_length_percent_number petal_width date_collected      species
           <dbl>        <dbl> <chr>       <chr>                        <chr>       <dttm>              <chr>
 1             1          5.1 3.5         1.4                          0.2         2022-04-06 00:00:00 setosa
 2             2          4.9 3           1.4                          0.2         2022-04-07 00:00:00 setosa
 3             3          4.7 3.2         1.3                          0.2         2022-04-09 00:00:00 setosa
 4             4          4.6 3.1         1.5                          0.2         2022-04-10 00:00:00 setosa
 5             5          5   3.6         1.4                          0.2         2022-04-11 00:00:00 setosa
 6             6          5.4 3.9         1.7                          0.4         2022-04-12 00:00:00 setosas
 7             7          4.6 3.4         1.4                          0.3         2022-04-13 00:00:00 Setosa
 8             8          5   3.4         1.5                          0.2         2022-04-14 00:00:00 setosa
 9             9          4.4 2.9         NA                           0.2         2022-04-15 00:00:00 Setosa
10            10          4.9 na          1.5                          N/A         2022-04-16 00:00:00 setosa
# ... with 140 more rows
```

- Inconsistent NA, na, N/A values

Finding issues with our table

```
> df
# A tibble: 150 x 7
   sample_number sepal_length sepal_width petal_length_percent_number petal_width date_collected      species
           <dbl>        <dbl> <chr>       <chr>                        <chr>       <dttm>              <chr>
 1             1          5.1 3.5         1.4                          0.2         2022-04-06 00:00:00 setosa
 2             2          4.9 3           1.4                          0.2         2022-04-07 00:00:00 setosa
 3             3          4.7 3.2         1.3                          0.2         2022-04-09 00:00:00 setosa
 4             4          4.6 3.1         1.5                          0.2         2022-04-10 00:00:00 setosa
 5             5          5   3.6         1.4                          0.2         2022-04-11 00:00:00 setosa
 6             6          5.4 3.9         1.7                          0.4         2022-04-12 00:00:00 setosas
 7             7          4.6 3.4         1.4                          0.3         2022-04-13 00:00:00 Setosa
 8             8          5   3.4         1.5                          0.2         2022-04-14 00:00:00 setosa
 9             9          4.4 2.9         NA                           0.2         2022-04-15 00:00:00 Setosa
10            10          4.9 na          1.5                          N/A         2022-04-16 00:00:00 setosa
# ... with 140 more rows
```

- Inconsistent naming of variables (setosa, **S**etosa, setosa**s**)

Finding issues with our table

```
> df
# A tibble: 150 x 7
   sample_number sepal_length sepal_width petal_length_percent_number petal_width date_collected      species
           <dbl>        <dbl> <chr>       <chr>                        <chr>       <dttm>              <chr>
 1             1          5.1 3.5         1.4                          0.2         2022-04-06 00:00:00 setosa
 2             2          4.9 3           1.4                          0.2         2022-04-07 00:00:00 setosa
 3             3          4.7 3.2         1.3                          0.2         2022-04-09 00:00:00 setosa
 4             4          4.6 3.1         1.5                          0.2         2022-04-10 00:00:00 setosa
 5             5          5   3.6         1.4                          0.2         2022-04-11 00:00:00 setosa
 6             6          5.4 3.9         1.7                          0.4         2022-04-12 00:00:00 setosas
 7             7          4.6 3.4         1.4                          0.3         2022-04-13 00:00:00 Setosa
 8             8          5   3.4         1.5                          0.2         2022-04-14 00:00:00 setosa
 9             9          4.4 2.9         NA                           0.2         2022-04-15 00:00:00 Setosa
10            10          4.9 na          1.5                          N/A         2022-04-16 00:00:00 setosa
# ... with 140 more rows
```

- Incorrect data types

How to identify in a large table

- If you have a very large table (>100 rows), I would recommend not just eye-balling it
  - You could miss errors!

How to identify in a large table

- If you have a very large table (>100 rows), I would recommend not just eye-balling
  - You could miss errors!

- Base R way:
  - df$species %>% unique()
  - $ = print the column as a vector
  - %>% = pipe this vector into another function (*so not strictly base R* :P)
  - unique() = return a vector with duplicate elements removed

How to identify in a large table

- If you have a very large table (>100 rows), I would recommend not just eye-balling
  - You could miss errors!

- Base R way:
  - df$species %>% unique()

```
> df$species %>% unique()
[1] "setosa"     "setosas"    "Setosa"     "versicolor" "virginica"
```

  - $ = print the column as a vector
  - %>% = pipe this vector into another function (*so not strictly base R* :P)
  - unique() = return a vector with duplicate elements removed

How to identify in a large table

- If you have a very large table (>100 rows), I would recommend not just eye-balling
  - You could miss errors!

- dplyr way:
  - df %>% group_by(species) %>% summarise(n = n())
    - group_by() creates a grouped copy of a table by columns

How to identify in a large table

- If you have a very large table (>100 rows), I would recommend not just eye-balling
  - You could miss errors!

- dplyr way:
  - df %>% group_by(species) %>% summarise(n = n())
    - group_by() creates a grouped copy of a table by columns
    - summarise() applies summary functions to columns
      - **n()** = number of values/rows
      - **mean()** = calculate mean
      - **median()** = calculate median
      - **sd()** = calculate standard deviation

How to identify in a large table

- If you have a very large table (>100 rows), I would recommend not just eye-balling
  - You could miss errors!

- dplyr way:
  - df %>% group_by(species) %>% summarise(n = n())

How to identify in a large table

- If you have a very large table (>100 rows), I would recommend not just eye-balling
  - You could miss errors!

- dplyr way:
  - df %>% group_by(species) %>% summarise(n = n())

```
> df %>% group_by(species) %>% summarise(n = n())
# A tibble: 5 x 2
  species        n
  <chr>      <int>
1 setosa        43
2 Setosa         3
3 setosas        4
4 versicolor    50
5 virginica     50
```

A quick aside

- These two functions are a great way of exploring your dataset too!
  - Finding out samples sizes between groups **n()**
  - Finding means/SDs of variables between groups

3. Fixing problems in table data

The mutate() function

- We need a way of making changes to large table data without manually doing it in excel

The mutate() function

- We need a way of making changes to large table data without manually doing it in excel

- In comes the **mutate()** function!

  - Adds new variables (columns) while keeping existing ones
    - Can also overwrite columns with the new variable too

  - mutate is a 'vectorized function'
    - Fancy way of saying *"takes vectors as input, outputs vector of same length"*

The mutate() function

- We need a way of making changes to large table data without manually doing it in excel

- In comes the **mutate()** function!

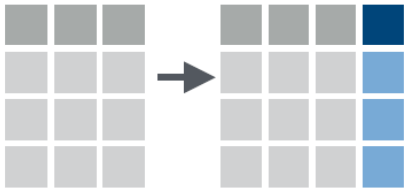
  - **Syntax:**

  - *dataframe* %>% mutate(new_column = what_you_want)

  - **Note:** if you're not piping you will have to specify the data you want to use!
    - E.g. mutate(*dataframe*, new_column = what_you_want)

The mutate() function

- We need a way of making changes to large table data without manually doing it in excel



- In comes the **mutate()** function!
  - **Example:** df %>% mutate(twox_sepal_length = 2 * sepal_length)

```
> df %>% mutate(twox_sepal_length = 2 * sepal_length)
# A tibble: 150 x 8
   sample_number sepal_length sepal_width petal_length_percent_numb~ petal_width date_collected      species twox_sepal_leng~
           <dbl>        <dbl> <chr>       <chr>                      <chr>       <dttm>              <chr>             <dbl>
 1             1          5.1 3.5         1.4                        0.2         2022-04-06 00:00:00 setosa             10.2
 2             2          4.9 3           1.4                        0.2         2022-04-07 00:00:00 setosa              9.8
 3             3          4.7 3.2         1.3                        0.2         2022-04-09 00:00:00 setosa              9.4
 4             4          4.6 3.1         1.5                        0.2         2022-04-10 00:00:00 setosa              9.2
 5             5          5   3.6         1.4                        0.2         2022-04-11 00:00:00 setosa             10
 6             6          5.4 3.9         1.7                        0.4         2022-04-12 00:00:00 setosas            10.8
 7             7          4.6 3.4         1.4                        0.3         2022-04-13 00:00:00 Setosa              9.2
 8             8          5   3.4         1.5                        0.2         2022-04-14 00:00:00 setosa             10
 9             9          4.4 2.9         NA                         0.2         2022-04-15 00:00:00 Setosa              8.8
10            10          4.9 na          1.5                        N/A         2022-04-16 00:00:00 setosa              9.8
# ... with 140 more rows
```

The mutate() function

- We need a way of making changes to large table data without manually doing it in excel

- In comes the **mutate()** function!

  - **Example:** df %>% mutate(twox_sepal_width = 2 * sepal_width)

```
> df %>% mutate(twox_sepal_width = 2 * sepal_width)
Error: Problem with `mutate()` column `twox_sepal_width`.
i `twox_sepal_width = 2 * sepal_width`.
x non-numeric argument to binary operator
```

```
> df
# A tibble: 150 x 7
   sample_number sepal_length sepal_width
           <dbl>        <dbl> <chr>
 1             1          5.1 3.5
 2             2          4.9 3
 3             3          4.7 3.2
 4             4          4.6 3.1
 5             5          5   3.6
 6             6          5.4 3.9
 7             7          4.6 3.4
 8             8          5   3.4
 9             9          4.4 2.9
10            10          4.9 na
# ... with 140 more rows
```

The mutate() function

- We need a way of making changes to large table data without manually doing it in excel

- In comes the **mutate()** function!

  - **Example:** df %>% mutate(twox_sepal_width = 2 * sepal_width)

```
> df %>% mutate(twox_sepal_width = 2 * sepal_width)
Error: Problem with `mutate()` column `twox_sepal_width`.
i `twox_sepal_width = 2 * sepal_width`.
x non-numeric argument to binary operator
```

```
> df
# A tibble: 150 x 7
   sample_number sepal_length sepal_width
           <dbl>        <dbl> <chr>
 1             1          5.1 3.5
 2             2          4.9 3
 3             3          4.7 3.2
 4             4          4.6 3.1
 5             5          5   3.6
 6             6          5.4 3.9
 7             7          4.6 3.4
 8             8          5   3.4
 9             9          4.4 2.9
10            10          4.9 na
# ... with 140 more rows
```

Correcting variables with stringr

- We have a way of overwriting variables in a column -- mutate()

- The **str_replace_all** function lets us change strings:

Correcting variables with stringr

- We have a way of overwriting variables in a column -- mutate()

- The **str_replace_all** function lets us change strings:

  - str_replace_all(string, pattern, replacement)
    - string = input vector (vectorized – like mutate)
    - pattern = the pattern to look for – e.g. cat
    - replacement = the replacement character vector – e.g. dog

Correcting variables with stringr

- We have a way of overwriting variables in a column -- mutate()

- The **str_replace_all** function lets us change strings:

  - str_replace_all(string, pattern, replacement)
    - string = input vector (vectorized – like mutate)
    - pattern = the pattern to look for – e.g. cat
    - replacement = the replacement character vector – e.g. dog

      - pets <- c("cat","cat","dog") #Create a character vector called pets
      - str_replace_all(pets,"cat","dog") #Replace all appearances of "cat" with "dog
      - output = "dog" "dog" "dog"

      - **Note that we need to use "", as these are character values!**

Correcting variables with stringr

- We have a way of overwriting variables in a column -- mutate()

- The **str_replace_all** function lets us change strings:

  - str_replace_all(string, pattern, replacement)
    - string = input vector (vectorized – like mutate)
    - pattern = the pattern to look for – e.g. cat
    - replacement = the replacement character vector – e.g. dog

      - pattern can itself be a vector if we have multiple patterns to match! E.g. "dog", "cat"
      - In this case, our character vector needs to have "|" separating each value ("|" = OR)
      - patterns <- c("cat|dog")
      - str_replace_all(pets, patterns, "possum")
      - output = "possum" "possum" "possum"

Applying a function across all columns

- We now need a way of applying this function across all columns to clean up the Nas
    - Because str_replace_all works on a vector (remember tibbles can be considered a list of vectors!)

Applying a function across all columns

- We now need a way of applying this function across all columns to clean up the Nas
  - Because str_replace_all works on a vector (remember tibbles can be considered a list of vectors!)

- Meet the **across()** function
  - across(.cols = X, .fns = X)
    - .cols = the columns that you want to apply a function
    - .fns = the function that you want to apply to the chosen columns
    - '~' is needed sometimes in front of .fns (short for function() in Tidyverse)

Applying a function across all columns

- We now need a way of applying this function across all columns to clean up the Nas
  - Because str_replace_all works on a vector (remember tibbles can be considered a list of vectors!)

- Meet the **across()** function
  - across(.cols = X, .fns = X)
    - .cols = the columns that you want to apply a function
    - .fns = the function that you want to apply to the chosen columns
    - '~' is needed sometimes in front of .fns (short for function() in Tidyverse)

- .cols examples:
  - c(sepal.length, sepal.width)
  - starts_with("sepal") #see also ends_with() and contains()
  - everything()

Applying a function across all columns

- We now need a way of applying this function across all columns to clean up the Nas
    - Because str_replace_all works on a vector (remember tibbles can be considered a list of vectors!)

- Meet the **across()** function
    - across(.cols = X, .fns = X)
        - .cols = the columns that you want to apply a function
        - .fns = the function that you want to apply to the chosen columns
        - '~' is needed sometimes in front of .fns (short for function() in Tidyverse)
- .cols examples:
    - c(sepal.length, sepal.width)
    - starts_with("sepal") #see also ends_with() and contains()
    - everything()
- **Note: across() is only used within dpylr functions**

Combining these functions

- Let's now combine these functions we learned to fix our NA problem!

- See the Fixing_NAs.R file

Exercise time!

- See the exercises in “Exercises.R”

Summary

- Learned to group and summarise our table data with
  - group_by() and summarise()
- Create new columns and overwrite existing
  - mutate()
- Replace character strings (equiv. to find/replace)
  - str_replace()
- Perform a dpylr function to multiple or all columns in a table
  - across()