R2: Advanced Data Wrangling

Data manipulation with `{dplyr}` in `{tidyverse}`

Andreas Reschreiter

2023-11-20

Table of contents

1 Advanced data manipulation **2**
- 1.1 Regular expressions . . . 2
- 1.2 Rules . . . 2
- 1.3 Advanced `select` columns . . . 2
- 1.4 Advanced `filter` rows . . . 3
- 1.5 Advanced `rename` columns . . . 5

2 Spezial data formats **5**
- 2.1 Data in `rownames` . . . 5
- 2.2 Long and wide data . . . 7
- 2.3 `iris` data wide and long . . . 9
- 2.4 A `sample_df` wide and long . . . 11

3 Combining data **12**
- 3.1 Binding . . . 12
- 3.2 Joins . . . 13

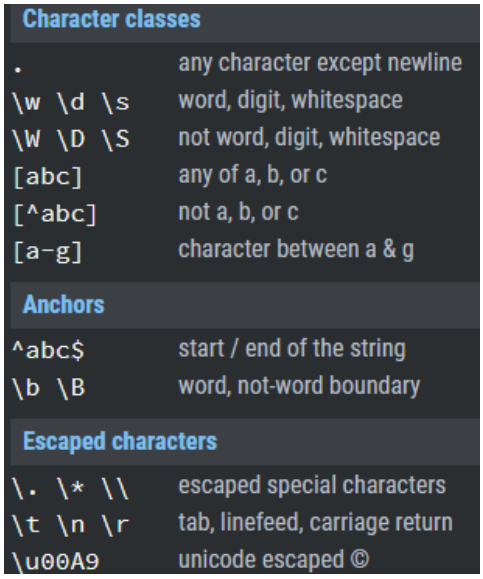
Packages used in this notebook:

```
suppressPackageStartupMessages({
  library(tidyverse)
})
```

1 Advanced data manipulation

1.1 Regular expressions

Regular expressions allow advanced data manipulation:



1.2 Rules

Following rules apply for symbols on how many times:

Symbol	Meaning	In other words
`?`	zero or one	at most once
`*`	zero or more	any
`+`	one or more	at least once

1.3 Advanced `select` columns

A regular expression can be used to select only column names that begin with the letter `S` and contain somewhere in the column name at least once a `.` in combination with `matches()`:

```
data = as_tibble(iris)
select(data, matches("^S.*\\."))
```

```
# A tibble: 150 x 2
   Sepal.Length Sepal.Width
          <dbl>       <dbl>
 1          5.1         3.5
 2          4.9         3
 3          4.7         3.2
 4          4.6         3.1
 5          5           3.6
 6          5.4         3.9
 7          4.6         3.4
 8          5           3.4
 9          4.4         2.9
10          4.9         3.1
# i 140 more rows
```

The regular expression `^S.*\\.` shows only data with column (names) that satisfy the following conditions:

- The first part in the regular expression `^S` specifies that the column name begins with an "S"
- The regular expression term .* means that any characters can follow (zero or more times) after the "S" at the beginning of the column name
- The regular expression part `\\.` is needed, because the `.` has a special meaning in regular expressions. The `.` needs the escape character `\` in front of it. The escape character `\` itself need also an escape character `\` and this results in the term `\\` before the `.` and in the `\\.` part of the regular expression.

1.4 Advanced `filter` rows

Find all rows where *any* variable has a value > 5:

```
filter_all(data, any_vars(. > 5))
```

```
# A tibble: 118 x 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1          5.1         3.5          1.4         0.2 setosa
 2          5.4         3.9          1.7         0.4 setosa
 3          5.4         3.7          1.5         0.2 setosa
 4          5.8         4            1.2         0.2 setosa
```

```
 5          5.7         4.4          1.5         0.4 setosa
 6          5.4         3.9          1.3         0.4 setosa
 7          5.1         3.5          1.4         0.3 setosa
 8          5.7         3.8          1.7         0.3 setosa
 9          5.1         3.8          1.5         0.3 setosa
10          5.4         3.4          1.7         0.2 setosa
# i 108 more rows
```

Find all rows where *all* variables ending with `Length` have a value > 5:

```
  filter_at(data, vars(ends_with("Length")), all_vars(. > 5))
```

```
# A tibble: 42 x 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1          6           2.7          5.1         1.6 versicolor
 2          6.3         3.3          6           2.5 virginica
 3          5.8         2.7          5.1         1.9 virginica
 4          7.1         3            5.9         2.1 virginica
 5          6.3         2.9          5.6         1.8 virginica
 6          6.5         3            5.8         2.2 virginica
 7          7.6         3            6.6         2.1 virginica
 8          7.3         2.9          6.3         1.8 virginica
 9          6.7         2.5          5.8         1.8 virginica
10          7.2         3.6          6.1         2.5 virginica
# i 32 more rows
```

Find all rows where all numeric variables exceed 2:

```
  filter_if(data, is.numeric, all_vars(. > 2))
```

```
# A tibble: 23 x 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1          6.3         3.3          6           2.5 virginica
 2          7.1         3            5.9         2.1 virginica
 3          6.5         3            5.8         2.2 virginica
 4          7.6         3            6.6         2.1 virginica
 5          7.2         3.6          6.1         2.5 virginica
 6          6.8         3            5.5         2.1 virginica
```

```
 7          5.8         2.8          5.1         2.4 virginica
 8          6.4         3.2          5.3         2.3 virginica
 9          7.7         3.8          6.7         2.2 virginica
10          7.7         2.6          6.9         2.3 virginica
# i 13 more rows
```

1.5 Advanced `rename` columns

Rename several variables with a transform function:

```
rename_all(data, str_replace, "\\.", "_")
```

```
# A tibble: 150 x 5
   Sepal_Length Sepal_Width Petal_Length Petal_Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa
 3          4.7         3.2          1.3         0.2 setosa
 4          4.6         3.1          1.5         0.2 setosa
 5          5           3.6          1.4         0.2 setosa
 6          5.4         3.9          1.7         0.4 setosa
 7          4.6         3.4          1.4         0.3 setosa
 8          5           3.4          1.5         0.2 setosa
 9          4.4         2.9          1.4         0.2 setosa
10          4.9         3.1          1.5         0.1 setosa
# i 140 more rows
```

2 Spezial data formats

2.1 Data in `rownames`

The `rownames()` of a `data.frame` may contain important information.

```
USArrests %>% head(4)
```

```
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
```

Consider the `USArests` data set and the information in `rownames()`:

```
rownames(USArrests)
```

```
 [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
 [5] "California"     "Colorado"       "Connecticut"    "Delaware"
 [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
[13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
[17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
[21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
[25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
[29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
[33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
[37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
[41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
[45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
[49] "Wisconsin"      "Wyoming"
```

The names of the 50 states are in `rownames()` and the names of the states are lost with a too simple transformation into a `tibble`:

```
as_tibble(USArrests) %>% head(4)
```

```
# A tibble: 4 x 4
  Murder Assault UrbanPop  Rape
   <dbl>   <int>    <int> <dbl>
1   13.2     236       58  21.2
2   10       263       48  44.5
3    8.1     294       80  31
4    8.8     190       50  19.5
```

The `rownames()` can be placed into a separate column before the transformation into a tibble:

```
USArrests %>% rownames_to_column("state") %>% as_tibble %>% head(4)
```

```
# A tibble: 4 x 5
  state    Murder Assault UrbanPop  Rape
  <chr>     <dbl>   <int>    <int> <dbl>
1 Alabama    13.2     236       58  21.2
```

```
2 Alaska     10       263       48  44.5
3 Arizona     8.1     294       80  31
4 Arkansas    8.8     190       50  19.5
```

Alternatively the following also keeps the `rownames`:

```
USArrests %>% as_tibble(rownames = "state") %>% head(4)
```

```
# A tibble: 4 x 5
  state    Murder Assault UrbanPop  Rape
  <chr>     <dbl>   <int>    <int> <dbl>
1 Alabama    13.2     236       58  21.2
2 Alaska     10       263       48  44.5
3 Arizona     8.1     294       80  31
4 Arkansas    8.8     190       50  19.5
```

2.2 Long and wide data

Consider the wide data format of the `USArrests` data:

```
USArrests %>% filter(Murder > 16)
```

```
            Murder Assault UrbanPop Rape
Georgia       17.4     211       60 25.8
Mississippi   16.1     259       44 17.1
```

Use `pivot_longer()` to get wide data into long format:

```
USArrests |>
  filter(Murder > 16) |>
  rownames_to_column(var = "State") |>
  pivot_longer(
    cols = -c("State","UrbanPop") # NOTE all columns expect State & urbanPop
    # ,names_to = "Crime"         # NOTE name if not specified
    # ,values_to = "Arrests"      # NOTE value if not specified
    ) -> arrests_long

arrests_long
```

```
# A tibble: 6 x 4
  State       UrbanPop name    value
  <chr>          <int> <chr>   <dbl>
1 Georgia           60 Murder   17.4
2 Georgia           60 Assault 211
3 Georgia           60 Rape     25.8
4 Mississippi       44 Murder   16.1
5 Mississippi       44 Assault 259
6 Mississippi       44 Rape     17.1
```

Older code may use `gather()` to transform wide data into long format:

```
arrests_long <- USArrests %>% filter(Murder > 16) %>%
  rownames_to_column("State") %>% ## to keep info -- gather() will remove rownames
  gather(key   = "Crime",
         value = "Arrests",
         Murder, Assault, Rape)
arrests_long
```

```
        State UrbanPop   Crime Arrests
1     Georgia       60  Murder    17.4
2 Mississippi       44  Murder    16.1
3     Georgia       60 Assault   211.0
4 Mississippi       44 Assault   259.0
5     Georgia       60    Rape    25.8
6 Mississippi       44    Rape    17.1
```

Use `pivot_wider()` for transforming long into wide format:

```
arrests_long |>
  pivot_wider(
    names_from = "Crime",
    values_from = "Arrests")
```

```
# A tibble: 2 x 5
  State       UrbanPop Murder Assault  Rape
  <chr>          <int>  <dbl>   <dbl> <dbl>
1 Georgia           60   17.4     211  25.8
2 Mississippi       44   16.1     259  17.1
```

Older code may use `spread()` for transforming long into wide format:

```
arrests_long %>% spread(Crime, Arrests)
```

```
        State UrbanPop Assault Murder Rape
1     Georgia       60     211   17.4 25.8
2 Mississippi       44     259   16.1 17.1
```

2.3 `iris` data wide and long

- Transform `iris` into long format and back.
- Need an unique identifier (like `id`) for each case (row).

Without `id` variable:

```
iris |> as_tibble() |>
  pivot_longer(cols = -Species) # NOTE row information is lost
```

```
# A tibble: 600 x 3
   Species name         value
   <fct>   <chr>        <dbl>
 1 setosa  Sepal.Length   5.1
 2 setosa  Sepal.Width    3.5
 3 setosa  Petal.Length   1.4
 4 setosa  Petal.Width    0.2
 5 setosa  Sepal.Length   4.9
 6 setosa  Sepal.Width    3
 7 setosa  Petal.Length   1.4
 8 setosa  Petal.Width    0.2
 9 setosa  Sepal.Length   4.7
10 setosa  Sepal.Width    3.2
# i 590 more rows
```

```
iris |> as_tibble() |>
  pivot_longer(cols = -Species) |>
  pivot_wider(
    names_from = "name",
    values_from = "value")
```

```
# A tibble: 3 x 5
  Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
```

```
  <fct>      <list>       <list>      <list>       <list>
1 setosa     <dbl [50]>   <dbl [50]>  <dbl [50]>   <dbl [50]>
2 versicolor <dbl [50]>   <dbl [50]>  <dbl [50]>   <dbl [50]>
3 virginica  <dbl [50]>   <dbl [50]>  <dbl [50]>   <dbl [50]>
```

With an `id` variable as identifier

```
iris_long <- iris |>
  as_tibble() |>
  group_by(Species) |>                    # NOTE group by different Species (id=1..50)
  mutate(id = row_number(), .before=1) |> # NOTE rownumber() as id per Species
  ungroup() |>                            # NOTE remove grouping of data
  pivot_longer(cols = -c("id","Species")) # NOTE keep info about data link

iris_long |>
  arrange(id, Species)  # alternatively slice_sample(n = 7)
```

```
# A tibble: 600 x 4
      id Species    name         value
   <int> <fct>      <chr>        <dbl>
 1     1 setosa     Sepal.Length   5.1
 2     1 setosa     Sepal.Width    3.5
 3     1 setosa     Petal.Length   1.4
 4     1 setosa     Petal.Width    0.2
 5     1 versicolor Sepal.Length   7
 6     1 versicolor Sepal.Width    3.2
 7     1 versicolor Petal.Length   4.7
 8     1 versicolor Petal.Width    1.4
 9     1 virginica  Sepal.Length   6.3
10     1 virginica  Sepal.Width    3.3
# i 590 more rows
```

```
iris_long |>
  pivot_wider(
    names_from = "name",
    values_from = "value")
```

```
# A tibble: 150 x 6
      id Species Sepal.Length Sepal.Width Petal.Length Petal.Width
   <int> <fct>          <dbl>       <dbl>        <dbl>       <dbl>
```

```
 1     1 setosa          5.1         3.5          1.4         0.2
 2     2 setosa          4.9         3            1.4         0.2
 3     3 setosa          4.7         3.2          1.3         0.2
 4     4 setosa          4.6         3.1          1.5         0.2
 5     5 setosa          5           3.6          1.4         0.2
 6     6 setosa          5.4         3.9          1.7         0.4
 7     7 setosa          4.6         3.4          1.4         0.3
 8     8 setosa          5           3.4          1.5         0.2
 9     9 setosa          4.4         2.9          1.4         0.2
10    10 setosa          4.9         3.1          1.5         0.1
# i 140 more rows
```

2.4 A `sample_df` wide and long

Source: R-Studio Community

Transform the following `sample_df` into long-format and add a variable `Response_Age`:

```
sample_df <- data.frame(
  stringsAsFactors = FALSE,
                ID = c("001", "002", "003", "004", "005"),
       Current_Age = c(75, 38, 29, 45, 47),
     Response_2015 = c("Yes", "No", "Yes", "Yes", "No"),
     Response_2010 = c("No", "No", "Yes", NA, "Yes"),
     Response_2007 = c("Yes", "No", "Yes", "No", NA),
     Response_2005 = c("Yes", "Yes", NA, "No", "No")
)
sample_df
```

```
  ID Current_Age Response_2015 Response_2010 Response_2007 Response_2005
1 001          75           Yes            No           Yes           Yes
2 002          38            No            No            No           Yes
3 003          29           Yes           Yes           Yes          <NA>
4 004          45           Yes          <NA>            No            No
5 005          47            No           Yes          <NA>            No
```

```
sample_df %>%
  pivot_longer(
    cols = starts_with("Response"),
    names_to = "Year",
    names_pattern = "Response_(.+)",
```

```
    names_transform = list(Year = as.integer),
    values_to = "Response") %>%
  group_by(ID) %>%
  mutate(Response_Age = Current_Age - (max(Year)-Year)) %>%
  arrange(desc(Year), ID)
```

```
# A tibble: 20 x 5
# Groups:   ID [5]
   ID    Current_Age  Year Response Response_Age
   <chr>       <dbl> <int> <chr>           <dbl>
 1 001            75  2015 Yes                75
 2 002            38  2015 No                 38
 3 003            29  2015 Yes                29
 4 004            45  2015 Yes                45
 5 005            47  2015 No                 47
 6 001            75  2010 No                 70
 7 002            38  2010 No                 33
 8 003            29  2010 Yes                24
 9 004            45  2010 <NA>               40
10 005            47  2010 Yes                42
11 001            75  2007 Yes                67
12 002            38  2007 No                 30
13 003            29  2007 Yes                21
14 004            45  2007 No                 37
15 005            47  2007 <NA>               39
16 001            75  2005 Yes                65
17 002            38  2005 Yes                28
18 003            29  2005 <NA>               19
19 004            45  2005 No                 35
20 005            47  2005 No                 37
```

3 Combining data

3.1 Binding

```
A = select(slice(iris, 1:2), 1:2)
A
```

```
  Sepal.Length Sepal.Width
```

```
1          5.1         3.5
2          4.9         3.0
```

```
B = select(slice(iris, 3:4), 2:3)
B
```

```
  Sepal.Width Petal.Length
1         3.2          1.3
2         3.1          1.5
```

```
bind_rows(A, B)
```

```
  Sepal.Length Sepal.Width Petal.Length
1          5.1         3.5           NA
2          4.9         3.0           NA
3           NA         3.2          1.3
4           NA         3.1          1.5
```

```
bind_cols(A, B)
```

```
  Sepal.Length Sepal.Width...2 Sepal.Width...3 Petal.Length
1          5.1             3.5             3.2          1.3
2          4.9             3.0             3.1          1.5
```

3.2 Joins

Classic data base operations (e.g., inner/left/right/full outer-joins, union/intersect etc.) are also availabe:

```
tmp <- tribble(~ Species, ~German_Name, ~Color,
               "versicolor", "Sumpfschwertlilie", "diverse",
               "setosa", "Borsten-Schwertlilie", "magenta")
tmp
```

```
# A tibble: 2 x 3
  Species    German_Name          Color
  <chr>      <chr>                <chr>
1 versicolor Sumpfschwertlilie    diverse
2 setosa     Borsten-Schwertlilie magenta
```

```r
inner_join(data, tmp) ## only 100 rows, since virginica is missing in tmp
```

```
# A tibble: 100 x 7
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species German_Name   Color
          <dbl>       <dbl>        <dbl>       <dbl> <chr>   <chr>         <chr>
 1          5.1         3.5          1.4         0.2 setosa  Borsten-Schw~ mage~
 2          4.9         3            1.4         0.2 setosa  Borsten-Schw~ mage~
 3          4.7         3.2          1.3         0.2 setosa  Borsten-Schw~ mage~
 4          4.6         3.1          1.5         0.2 setosa  Borsten-Schw~ mage~
 5          5           3.6          1.4         0.2 setosa  Borsten-Schw~ mage~
 6          5.4         3.9          1.7         0.4 setosa  Borsten-Schw~ mage~
 7          4.6         3.4          1.4         0.3 setosa  Borsten-Schw~ mage~
 8          5           3.4          1.5         0.2 setosa  Borsten-Schw~ mage~
 9          4.4         2.9          1.4         0.2 setosa  Borsten-Schw~ mage~
10          4.9         3.1          1.5         0.1 setosa  Borsten-Schw~ mage~
# i 90 more rows
```

```r
left_join(data, tmp) ## all rows, but cols are NA for virginica
```

```
# A tibble: 150 x 7
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species German_Name   Color
          <dbl>       <dbl>        <dbl>       <dbl> <chr>   <chr>         <chr>
 1          5.1         3.5          1.4         0.2 setosa  Borsten-Schw~ mage~
 2          4.9         3            1.4         0.2 setosa  Borsten-Schw~ mage~
 3          4.7         3.2          1.3         0.2 setosa  Borsten-Schw~ mage~
 4          4.6         3.1          1.5         0.2 setosa  Borsten-Schw~ mage~
 5          5           3.6          1.4         0.2 setosa  Borsten-Schw~ mage~
 6          5.4         3.9          1.7         0.4 setosa  Borsten-Schw~ mage~
 7          4.6         3.4          1.4         0.3 setosa  Borsten-Schw~ mage~
 8          5           3.4          1.5         0.2 setosa  Borsten-Schw~ mage~
 9          4.4         2.9          1.4         0.2 setosa  Borsten-Schw~ mage~
10          4.9         3.1          1.5         0.1 setosa  Borsten-Schw~ mage~
# i 140 more rows
```