Data import

Luis Francisco Gomez Lopez

2023-09-01

Contents

Reading data from a file

- **CSV**: comma separated values

```
students <- read_csv(file = '../000_data_sets/008_students.csv')
```

```
Rows: 6 Columns: 5
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (4): Full Name, favourite.food, mealPlan, AGE
dbl (1): Student ID

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
students
```

```
# A tibble: 6 x 5
  `Student ID` `Full Name`      favourite.food     mealPlan            AGE
         <dbl> <chr>            <chr>              <chr>               <chr>
1            1 Sunil Huffmann   Strawberry yoghurt Lunch only          4
2            2 Barclay Lynn     French fries       Lunch only          5
3            3 Jayendra Lyne    N/A                Breakfast and lunch 7
4            4 Leon Rossini     Anchovies          Lunch only          <NA>
5            5 Chidiegwu Dunkel Pizza              Breakfast and lunch five
6            6 Güvenç Attila    Ice cream          Lunch only          6
```

Reading data from a file

- Specifying NA values

```r
students <- read_csv(file = '../000_data_sets/008_students.csv',
                     na = c('', 'N/A'))
```

```
Rows: 6 Columns: 5
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (4): Full Name, favourite.food, mealPlan, AGE
dbl (1): Student ID

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
students
```

```
# A tibble: 6 x 5
  `Student ID` `Full Name`      favourite.food     mealPlan            AGE
         <dbl> <chr>            <chr>              <chr>               <chr>
1            1 Sunil Huffmann   Strawberry yoghurt Lunch only          4
2            2 Barclay Lynn     French fries       Lunch only          5
3            3 Jayendra Lyne    <NA>               Breakfast and lunch 7
4            4 Leon Rossini     Anchovies          Lunch only          <NA>
5            5 Chidiegwu Dunkel Pizza              Breakfast and lunch five
6            6 Güvenç Attila    Ice cream          Lunch only          6
```

Reading data from a file

- Clean names with `janitor::clean_names()`

```
students <- students |>
  janitor::clean_names()
students
```

```
# A tibble: 6 x 5
  student_id full_name        favourite_food     meal_plan           age
       <dbl> <chr>            <chr>              <chr>               <chr>
1          1 Sunil Huffmann   Strawberry yoghurt Lunch only          4
2          2 Barclay Lynn     French fries       Lunch only          5
3          3 Jayendra Lyne    <NA>               Breakfast and lunch 7
4          4 Leon Rossini     Anchovies          Lunch only          <NA>
5          5 Chidiegwu Dunkel Pizza              Breakfast and lunch five
6          6 Güvenç Attila    Ice cream          Lunch only          6
```

Reading data from a file

- Specify the correct column types

```r
students <- students |>
  mutate(meal_plan = factor(x = meal_plan, ordered = FALSE),
         age = parse_number(x = if_else(condition = age == 'five',
                                            true = '5',
                                            false = age)))
students
```

```
# A tibble: 6 x 5
  student_id full_name        favourite_food     meal_plan             age
       <dbl> <chr>            <chr>              <fct>               <dbl>
1          1 Sunil Huffmann   Strawberry yoghurt Lunch only              4
2          2 Barclay Lynn     French fries       Lunch only              5
3          3 Jayendra Lyne    <NA>               Breakfast and lunch     7
4          4 Leon Rossini     Anchovies          Lunch only             NA
5          5 Chidiegwu Dunkel Pizza              Breakfast and lunch     5
6          6 Güvenç Attila    Ice cream          Lunch only              6
```

Reading data from a file

- Other arguments
  - `skip`: number of lines to skip before reading data
  - `colnames`: specify column names

```
"The first line of metadata
 The second line of metadata
 x,y,z
 1,2,3" |>
  read_csv(skip = 2)
```

```
# A tibble: 1 x 3
      x     y     z
  <dbl> <dbl> <dbl>
1     1     2     3
```

```
"1,2,3
 4,5,6" |>
  read_csv(col_names = c('x', 'y', 'z'))
```

```
# A tibble: 2 x 3
      x     y     z
  <dbl> <dbl> <dbl>
1     1     2     3
2     4     5     6
```

Reading data from a file

- Other file types
  - `read_csv2()`: the delimiter is `;`
  - `read_tsv()`: the delimiter is `\t`
  - `read_fwf()`: fixed-width files
  - `read_delim()`: any delimiter

Controlling column types

- Guessing types: `readr` uses a heuristic to figure out the column types
  - Inspect 1000 values evenly spaced from the first to the last row
  - Does it contain only `F`, `T`, `FALSE`, or `TRUE` (ignoring case)? If so, it's a logical
  - Does it contain only numbers (for example `1`, `-4.5`, `5e6`, `Inf`)? If so, it's a number
  - Does it match the ISO8601 standard? If so, it's a date or date-time
  - Otherwise, it must be a string

```r
read_csv("
  logical,numeric,date,string
  TRUE,1,2021-01-15,abc
  false,4.5,2021-02-15,def
  T,Inf,2021-02-16,ghi
")
```

```
# A tibble: 3 x 4
  logical numeric date       string
  <lgl>     <dbl> <date>     <chr>
1 TRUE        1   2021-01-15 abc
2 FALSE       4.5 2021-02-15 def
3 TRUE      Inf   2021-02-16 ghi
```

Controlling column types

- Missing values, column types, and problems

```
simple_csv <- "
  x
  10
  .
  20
  30"

simple_csv |>
  read_csv()
```

```
# A tibble: 4 x 1
  x
  <chr>
1 10
2 .
3 20
4 30
```

Controlling column types

- Missing values, column types, and problems

```
my_tibble <- simple_csv |>
  read_csv(col_types = cols(x = col_double()))
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for details,
e.g.:
  dat <- vroom(...)
  problems(dat)
```

```
problems(x = my_tibble)
```

```
# A tibble: 1 x 5
    row   col expected actual file
  <int> <int> <chr>    <chr>  <chr>
1     3     1 a double .      C:/Users/Usuario/AppData/Local/Temp/RtmpcTvuZS/fi~
```

```
read_csv(file = simple_csv,
         col_types = cols(x = col_double()), na = c('.'))
```

```
# A tibble: 4 x 1
      x
  <dbl>
1    10
2    NA
3    20
4    30
```

Controlling column types

- Column types
  - `col_logical()`: containing only `T`, `F`, `TRUE` or `FALSE`
  - `col_integer()`: integers
  - `col_double()`: doubles
  - `col_character()`: strings
  - `col_factor()`: factors
  - `col_date()`: dates with a format specification
  - `col_datetime()`: ISO8601 date times
  - `col_number()`: numbers containing a grouping mark
  - `col_skip()`: skip and don't import this column

Reading data from multiple files

- Read data separated in different files and stack them on top of each other in a single data frame

```
sale_files <- c('../000_data_sets/008_01-sales.csv',
                '../000_data_sets/008_02-sales.csv',
                '../000_data_sets/008_03-sales.csv')
read_csv(file = sale_files, id = 'file') |> head(n = 5)
```

```
# A tibble: 5 x 6
  file                               month    year brand  item     n
  <chr>                              <chr>   <dbl> <dbl> <dbl> <dbl>
1 ../000_data_sets/008_01-sales.csv January  2019     1  1234     3
2 ../000_data_sets/008_01-sales.csv January  2019     1  8721     9
3 ../000_data_sets/008_01-sales.csv January  2019     1  1822     2
4 ../000_data_sets/008_01-sales.csv January  2019     2  3333     1
5 ../000_data_sets/008_01-sales.csv January  2019     2  2156     9
```

- List the files in a directory

```
list.files(path = '../000_data_sets/', pattern = r'(sales\.csv$)')
```

```
[1] "008_01-sales.csv" "008_02-sales.csv" "008_03-sales.csv"
```

Writing to a file

- Write a data frame to a `csv` file

```
students |> write_csv(file = '../000_data_sets/008_students2.csv',
                      na = '')
```

- Write a data frame to a single **R** object

```
students |> write_rds(file = '../000_data_sets/008_students.rds')
```

- Write a data frame to a `parquet`[1] file

```
students |> write_parquet(sink = '../000_data_sets/008_students.parquet')
```

[1]Apache Parquet is a free and open-source column-oriented data storage format in the Apache Hadoop ecosystem

Data entry

- Using a `tibble`

```
tibble(x = c(1, 2, 5),
       y = c("h", "m", "g"),
       z = c(0.08, 0.83, 0.60))
```

```
# A tibble: 3 x 3
      x y         z
  <dbl> <chr> <dbl>
1     1 h      0.08
2     2 m      0.83
3     5 g      0.6
```

- Using a `tribble`: **tr**ansposed t**ibble**

```
tribble(~x, ~y, ~z,
        1, "h", 0.08,
        2, "m", 0.83,
        5, "g", 0.60)
```

```
# A tibble: 3 x 3
      x y         z
  <dbl> <chr> <dbl>
1     1 h      0.08
2     2 m      0.83
3     5 g      0.6
```

References I