Parsers

David Gerard

2023-10-09

Learning Objectives

- Change character vectors into other types using parsers.
- Parsers and reader.
- Chapter 11 of RDS

Motivation

- Suppose you have the following data frame

```
suppressPackageStartupMessages(library(tidyverse))
dfdat <- tribble(
  ~date,        ~time,      ~number, ~factor, ~logical,
  ##----------  ----------  -------  -------  --------
  "12-01-1988", "10:10:02", "2",     "A",     "TRUE",
  "11-12-1987", "11:10:57", "4",     "A",     "TRUE",
  "02-03-1989", "10:10:25", "6",     "B",     "FALSE",
  "06-03-1982", "22:10:55", "2",     "B",     "TRUE",
  "09-21-1981", "10:10:02", "1",     "A",     "FALSE"
  )
dfdat
```

```
## # A tibble: 5 x 5
##   date       time     number factor logical
##   <chr>      <chr>    <chr>  <chr>  <chr>
## 1 12-01-1988 10:10:02 2      A      TRUE
## 2 11-12-1987 11:10:57 4      A      TRUE
## 3 02-03-1989 10:10:25 6      B      FALSE
## 4 06-03-1982 22:10:55 2      B      TRUE
## 5 09-21-1981 10:10:02 1      A      FALSE
```

- How do we convert the characters to the types we want? Parse!

Parsing dates and times

- See `{lubridate}` notes.

Parsing Numbers

- `parse_double()` and `parse_integer()` expect strict numbers and will fail if there is anything non-number-like.

```r
parse_double("2.11")
```

```
## [1] 2.11
```

```r
parse_double("$2.11")
```

```
## Warning: 1 parsing failure.
## row col expected actual
##   1  -- a double  $2.11

## [1] NA
## attr(,"problems")
## # A tibble: 1 x 4
##     row   col expected actual
##   <int> <int> <chr>    <chr>
## 1     1    NA a double $2.11
```

```r
parse_integer("2")
```

```
## [1] 2
```

```r
parse_integer("2%")
```

```
## Warning: 1 parsing failure.
## row col               expected actual
##   1  -- no trailing characters     2%

## [1] NA
## attr(,"problems")
## # A tibble: 1 x 4
##     row   col expected               actual
##   <int> <int> <chr>                  <chr>
## 1     1    NA no trailing characters 2%
```

- `parse_number()` removes non-numeric characters.

```r
parse_number("$2.11")
```

```
## [1] 2.11
```

```r
parse_number("2%")
```

```
## [1] 2
```

- You can change the grouping variable from "," to "." with

```r
parse_number("2.555,11",
             locale = locale(grouping_mark = ".",
                             decimal_mark = ","))
```

```
## [1] 2555
```

- Example:

```r
dfdat %>%
  mutate(number = parse_number(number))
```

```
## # A tibble: 5 x 5
##   date       time     number factor logical
##   <chr>      <chr>     <dbl> <chr>  <chr>
## 1 12-01-1988 10:10:02      2 A      TRUE
## 2 11-12-1987 11:10:57      4 A      TRUE
## 3 02-03-1989 10:10:25      6 B      FALSE
## 4 06-03-1982 22:10:55      2 B      TRUE
## 5 09-21-1981 10:10:02      1 A      FALSE
```

Parsing other types

- `parse_logical()` and `parse_factor()` and `parse_string()` are pretty self-explanatory.

```r
dfdat %>%
  mutate(factor = parse_factor(factor))
```

```
## # A tibble: 5 x 5
##   date       time     number factor logical
##   <chr>      <chr>    <chr>  <fct>  <chr>
## 1 12-01-1988 10:10:02 2      A      TRUE
## 2 11-12-1987 11:10:57 4      A      TRUE
## 3 02-03-1989 10:10:25 6      B      FALSE
## 4 06-03-1982 22:10:55 2      B      TRUE
## 5 09-21-1981 10:10:02 1      A      FALSE
```

```r
dfdat %>%
  mutate(logical = parse_logical(logical))
```

```
## # A tibble: 5 x 5
##   date       time     number factor logical
##   <chr>      <chr>    <chr>  <chr>  <lgl>
## 1 12-01-1988 10:10:02 2      A      TRUE
## 2 11-12-1987 11:10:57 4      A      TRUE
## 3 02-03-1989 10:10:25 6      B      FALSE
## 4 06-03-1982 22:10:55 2      B      TRUE
## 5 09-21-1981 10:10:02 1      A      FALSE
```

Parsing and readr

- When you specify `col_types` in `read_csv()`, those are wrappers for parsers.

```
read_csv("../../data/estate.csv",
         col_types = cols(
           Price   = col_double(),
           Area    = col_double(),
           Bed     = col_double(),
           Bath    = col_double(),
           AC      = col_logical(),
           Garage  = col_double(),
           Pool    = col_logical(),
           Year    = col_double(),
           Quality = col_factor(),
           Style   = col_factor(),
           Lot     = col_double(),
           Highway = col_logical()
           )) ->
  estate
estate
```

```
## # A tibble: 522 x 12
##     Price  Area   Bed  Bath AC    Garage Pool   Year Quality Style   Lot Highway
##     <dbl> <dbl> <dbl> <dbl> <lgl>  <dbl> <lgl> <dbl> <fct>   <fct> <dbl> <lgl>
##  1 360000  3032     4     4 TRUE       2 FALSE  1972 Medium  1     22221 FALSE
##  2 340000  2058     4     2 TRUE       2 FALSE  1976 Medium  1     22912 FALSE
##  3 250000  1780     4     3 TRUE       2 FALSE  1980 Medium  1     21345 FALSE
##  4 205500  1638     4     2 TRUE       2 FALSE  1963 Medium  1     17342 FALSE
##  5 275500  2196     4     3 TRUE       2 FALSE  1968 Medium  7     21786 FALSE
##  6 248000  1966     4     3 TRUE       5 TRUE   1972 Medium  1     18902 FALSE
##  7 229900  2216     3     2 TRUE       2 FALSE  1972 Medium  7     18639 FALSE
##  8 150000  1597     2     1 TRUE       1 FALSE  1955 Medium  1     22112 FALSE
##  9 195000  1622     3     2 TRUE       2 FALSE  1975 Low     1     14321 FALSE
## 10 160000  1976     3     3 FALSE      1 FALSE  1918 Low     1     32358 FALSE
## # i 512 more rows
```