# r4ds Ex 11

*MW*

*2019/06/19*

## 11.2.1

### 1

> What function would you use to read a file where fields were separated with "|"?

```
read_delim(hogehoge, delim = "|")
```

### 2

> Apart from `file`, `skip`, and `comment`, what other arguments do `read_csv()` and `read_tsv()` have in common?

```
?read_csv()
```

or

```
read_csv %>% formals() %>% names()
```

```
##  [1] "file"      "col_names" "col_types" "locale"    "na"
##  [6] "quoted_na" "quote"     "comment"   "trim_ws"   "skip"
## [11] "n_max"     "guess_max" "progress"
```

```
read_tsv %>% formals() %>% names()
```

```
##  [1] "file"      "col_names" "col_types" "locale"    "na"
##  [6] "quoted_na" "quote"     "comment"   "trim_ws"   "skip"
## [11] "n_max"     "guess_max" "progress"
```

```
df <- tibble(abc = 1, xyz = "a")
df$x
```

```
## Warning: Unknown or uninitialised column: 'x'.
```

```
## NULL
```

```
df[, "xyz"]
```

```
## # A tibble: 1 x 1
##   xyz
##   <chr>
## 1 a
```

```
df[, c("abc", "xyz")]
```

```
## # A tibble: 1 x 2
##     abc xyz
##   <dbl> <chr>
## 1     1 a
```

## 3

What are the most important arguments to `read_fwf()`?

This command can read fixed width files.

## 4

Sometimes strings in a CSV file contain commas. To prevent them from causing problems they need to be surrounded by a quoting character, like " or '. By convention, `read_csv()` assumes that the quoting character will be ", and if you want to change it you'll need to use `read_delim()` instead. What arguments do you need to specify to read the following text into a data frame?

```
"x,y\n1,'a,b'"
```

```
## [1] "x,y\n1,'a,b'"
```

```
x <- "x,y\n1,'a,b'"
read_delim(x, ",", quote="'")
```

```
## # A tibble: 1 x 2
##       x y
##   <int> <chr>
## 1     1 a,b
```

## 5

dentify what is wrong with each of the following inline CSV files. What happens when you run the code?

```
read_csv("a,b\n1,2,3\n4,5,6")
```

```
## Warning: 2 parsing failures.
## row # A tibble: 2 x 5 col     row col   expected  actual    file         expected   <int> <chr> <
```

```
## # A tibble: 2 x 2
##       a     b
##   <int> <int>
## 1     1     2
## 2     4     5
```

```
read_csv("a,b,c\n1,2\n1,2,3,4")
```

```
## Warning: 2 parsing failures.
## row # A tibble: 2 x 5 col     row col   expected  actual    file         expected   <int> <chr> <
```

```
## # A tibble: 2 x 3
##       a     b     c
##   <int> <int> <int>
## 1     1     2    NA
## 2     1     2     3
```

```
read_csv("a,b\n\"1")
```

```
## Warning: 2 parsing failures.
## row # A tibble: 2 x 5 col     row col   expected                    actual    file         expec
```

```
## # A tibble: 1 x 2
##       a b
##   <int> <chr>
## 1     1 <NA>
```

```
read_csv("a,b\n1,2\na,b")
```

```
## # A tibble: 2 x 2
##   a     b
##   <chr> <chr>
## 1 1     2
## 2 a     b
```

```
read_csv("a;b\n1;3")
```

```
## # A tibble: 1 x 1
##   `a;b`
##   <chr>
## 1 1;3
```

The wrong points of first three command are that number of columns and dataset do not match. The fourth problem is that data type between first and second rows are not match. The last one is that separate by using `;` is terrible in `read_csv()`

## 11.3.5

### 1

What are the most important arguments to `locale()`?

```
?locale()
```

`locale()` determine that - data format - time zone - numbers - encoding

## 2

What happens if you try and set `decimal_mark` and `grouping_mark` to the same character? What happens to the default value of `grouping_mark` when you set `decimal_mark` to `","`? What happens to the default value of `decimal_mark` when you set the `grouping_mark` to `"."`?

```
locale(decimal_mark = ".", grouping_mark = ".")
```

```
locale(decimal_mark = ",")
```

```
## <locale>
## Numbers:  123.456,78
## Formats:  %AD / %AT
## Timezone: UTC
## Encoding: UTF-8
## <date_names>
## Days:   Sunday (Sun), Monday (Mon), Tuesday (Tue), Wednesday (Wed),
##         Thursday (Thu), Friday (Fri), Saturday (Sat)
## Months: January (Jan), February (Feb), March (Mar), April (Apr), May
##         (May), June (Jun), July (Jul), August (Aug), September
##         (Sep), October (Oct), November (Nov), December (Dec)
## AM/PM:  AM/PM
```

```
locale(grouping_mark = ",")
```

```
## <locale>
## Numbers:  123,456.78
## Formats:  %AD / %AT
## Timezone: UTC
## Encoding: UTF-8
## <date_names>
## Days:   Sunday (Sun), Monday (Mon), Tuesday (Tue), Wednesday (Wed),
##         Thursday (Thu), Friday (Fri), Saturday (Sat)
## Months: January (Jan), February (Feb), March (Mar), April (Apr), May
##         (May), June (Jun), July (Jul), August (Aug), September
##         (Sep), October (Oct), November (Nov), December (Dec)
## AM/PM:  AM/PM
```

## 3

I didn't discuss the `date_format` and `time_format` options to `locale()`. What do they do? Construct an example that shows when they might be useful.

```
parse_date("19 junio 2019", "%d %B %Y", locale=locale("es"))
```

```
## [1] "2019-06-19"
```

### 4

If you live outside the US, create a new locale object that encapsulates the settings for the types of file you read most commonly.

```
japan_locale <- locale(date_format="%Y 年 %m 月 %d 日")
parse_date("2019 年 6 月 19 日", locale = japan_locale)
```

```
## [1] "2019-06-19"
```

### 5

What's the difference between `read_csv()` and `read_csv2()`?

Difference is delimiter.

### 6

What are the most common encodings used in Europe? What are the most common encodings used in Asia? Do some googling to find out.

skip

### 7

Generate the correct format string to parse each of the following dates and times:

```
d1 <- "January 1, 2010"
d2 <- "2015-Mar-07"
d3 <- "06-Jun-2017"
d4 <- c("August 19 (2015)", "July 1 (2015)")
d5 <- "12/30/14" # Dec 30, 2014
t1 <- "1705"
t2 <- "11:15:10.12 PM"
```

Following format is correct;

```
parse_date(d1, "%B %d, %Y")
```

```
## [1] "2010-01-01"
```

```
parse_date(d2, "%Y-%b-%d")
```

```
## [1] "2015-03-07"
```

```
parse_date(d3, "%d-%b-%Y")
```

```
## [1] "2017-06-06"
```

```r
parse_date(d4, "%B %d (%Y)")
```

```
## [1] "2015-08-19" "2015-07-01"
```

```r
parse_date(d5, "%m/%d/%y")
```

```
## [1] "2014-12-30"
```

```r
parse_time(t1, "%H%M")
```

```
## 17:05:00
```