R6: Import data files

Data import from files with `tidyverse`

Andreas Reschreiter

Contents

1 Import data from text files

Import data from text files.

1.1 Required packages

```
library(tidyverse) ## includes packages: readr, magrittr, dplyr, stringr, tidyr, tibble
```

1.2 Delimited data files

The structure of the text file and data can be inspected with (ordinary) text editors.

Delimited text (data) files use a column separator. Use `read_delim()` and specify the column separator in the `delim=` option. Wrappers exist for frequently used delimiters:

- `read_delim()` sets the separator in `delim=`
- `read_csv()` for comma
- `read_csv2()` for semi-colons
- `read_tsv()` for tabulator
- `read_table()` for white space as separators

1.3 Inspect the text file

- Visually inspection `data/text1.txt` with a text editor
- Can also read-in the first few lines in the file `data/text1.txt`

```
read_lines("data/text1.txt", n_max = 5)
```

```
## [1] "Sepal.Length,Sepal.Width,Petal.Length,Petal.Width,Species"
## [2] "5.1,3.5,1.4,0.2,setosa"
## [3] "4.9,3,1.4,0.2,setosa"
## [4] "4.7,3.2,1.3,0.2,setosa"
## [5] "4.6,3.1,1.5,0.2,setosa"
```

This reveals as separator a `","`, as decimal point a `"."` and the presence of column headings (in the first row).

1.4 Load data file

```
x = read_delim("data/text1.txt", delim = ",")
 #x = read_csv("data/text1.txt")
x
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <chr>
##  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
## # ... with 140 more rows
```

1.5 Diagnose problems

Use `problems(x)` to diagnose issues:

```
problems(read_delim("data/text1.txt", delim = ","))   # NOTE Empty --> no problems
```

```
## # A tibble: 0 x 5
## # ... with 5 variables: row <int>, col <int>, expected <chr>, actual <chr>,
## #   file <chr>
```

1.6 Column type specification

The functions attempt to guess the data types automatically. If this does not work, they can be specified manually. With "-" columns can be excluded.

```
x = read_csv("data/text1.txt", col_types = "cdi-l")
x |> head()
```

```
## # A tibble: 6 x 4
##   Sepal.Length Sepal.Width Petal.Length Species
##   <chr>              <dbl>        <int> <lgl>
## 1 5.1                  3.5           NA NA
## 2 4.9                  3             NA NA
## 3 4.7                  3.2           NA NA
## 4 4.6                  3.1           NA NA
## 5 5                    3.6           NA NA
## 6 5.4                  3.9           NA NA
```

Use `problems(x)` to diagnose issues:

```
problems(x) |> head()     # NOTE not empty when problems
```

```
## # A tibble: 6 x 5
##     row   col expected           actual file
##   <int> <int> <chr>              <chr>  <chr>
## 1     2     3 an integer         1.4    C:/Users/reschrei/Documents/Teaching/_D~
## 2     2     3 an integer         1.3    C:/Users/reschrei/Documents/Teaching/_D~
## 3     2     3 an integer         1.5    C:/Users/reschrei/Documents/Teaching/_D~
## 4     2     3 an integer         1.7    C:/Users/reschrei/Documents/Teaching/_D~
## 5     2     5 1/0/T/F/TRUE/FALSE setosa C:/Users/reschrei/Documents/Teaching/_D~
## 6     3     3 an integer         1.4    C:/Users/reschrei/Documents/Teaching/_D~
```

```
x = read_csv("data/text1.txt", col_types = "ccddc")
x |> problems()
```

```
## # A tibble: 0 x 5
## # ... with 5 variables: row <int>, col <int>, expected <chr>, actual <chr>,
## #   file <chr>
```

Character columns can be transformed with `mutate()`:

```
x %>% head  # NOTE wiht magritr pipe can use head instead of head()
```

```
## # A tibble: 6 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <chr>        <chr>              <dbl>       <dbl> <chr>
## 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
```

```
x |> mutate(Sepal.Length = parse_double(Sepal.Length),
            Sepal.Width = parse_double(Sepal.Width)) %>%
  head(2)
```

```
## # A tibble: 2 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <chr>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
```

1.7 Fixed-width data

Visual inspection of `data/text2.txt` (with a text editor) shows three text columns with fixed widths with spaces in the text and a varying number of spaces to separate the text columns.

Using the space separator `delim = " "` yields problems, because the text contains spaces the column are not quoted properly):

```
x = read_delim("data/text2.txt", delim = " ", col_names = F)
x %>%  problems
```

```
## # A tibble: 1 x 5
##     row   col expected   actual     file
##   <int> <int> <chr>      <chr>      <chr>
## 1     2    13 18 columns 13 columns C:/Users/reschrei/Documents/Teaching/_DE_MD~
```

```
x
```

```
## # A tibble: 2 x 18
##   X1    X2    X3    X4    X5    X6    X7    X8       X9 X10     X11   X12    X13
##   <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <dbl> <chr> <dbl> <dbl>  <dbl>
## 1 David Meyer <NA>  Höch~ 6,    1200  Wien  <NA>     NA <NA>     NA    NA     NA
## 2 Hugo  H.    Wolf  An    den   lang~ Lüss~ 47;    1190 Wien     43  4545 454545
## # ... with 5 more variables: X14 <lgl>, X15 <chr>, X16 <dbl>, X17 <lgl>,
## #   X18 <lgl>
```

```
x = read_table("data/text2.txt", col_names = F)
x %>%  problems
```

```
## # A tibble: 1 x 5
##     row col   expected  actual     file
##   <int> <chr> <chr>     <chr>      <chr>
## 1     2 <NA>  9 columns 13 columns 'data/text2.txt'
```

```
x
```

```
## # A tibble: 2 x 9
##   X1    X2    X3             X4    X5    X6     X7     X8           X9
##   <chr> <chr> <chr>          <chr> <chr> <chr>  <chr>  <chr>     <dbl>
## 1 David Meyer Höchstädtplatz 6,    1200  Wien   0699   12345674     NA
## 2 Hugo  H.    Wolf           An    den   langen Lüssen 47;        1190
```

Use `read.fwf()` for data with a fixed-width for each column.

```
data <- read_fwf("data/text2.txt", fwf_cols(Name = 12, Address = 37, Tel = 15))
data <- read_fwf("data/text2.txt", fwf_cols(Name = 13, Address = 35, Tel = 15))
data
```

```
## # A tibble: 2 x 3
##   Name          Address                             Tel
##   <chr>         <chr>                               <chr>
## 1 David Meyer   Höchstädtplatz 6, 1200 Wien         0699 12345674
## 2 Hugo H. Wolf  An den langen Lüssen 47; 1190 Wien +43 4545 45454
```

The file `data/text2.txt` contains in the rows from character 1-12 the name and in character 13 a single space character. The length of the first column has to be at least 12 characters and at most 13 characters. The address in character 14-47 has a length of 34 characters is followed by a single space character and the telephone number from character 48-62.

2 Separating and joining columns

Separate with `separate()` text in one column into different columns:

```
data2 <- data %>%
  mutate(Address = str_replace(Address, ",", ";")) %>%
  separate(Address, c("Street", "ZIPCity"), sep = "; ") %>%
  separate(ZIPCity, c("ZIP", "City"), sep = " ")
data2
```

```
## # A tibble: 2 x 5
##   Name         Street                 ZIP   City  Tel
##   <chr>        <chr>                  <chr> <chr> <chr>
## 1 David Meyer  Höchstädtplatz 6       1200  Wien  0699 12345674
## 2 Hugo H. Wolf An den langen Lüssen 47 1190  Wien  +43 4545 45454
```

Join with `unite()` text in different columns into one column:

```
data2 %>%
  unite(ZIPCity, ZIP, City, sep = " ") %>%
  unite(Address, ZIPCity, Street, sep = ", ")
```

```
## # A tibble: 2 x 3
##   Name         Address                           Tel
##   <chr>        <chr>                             <chr>
## 1 David Meyer  1200 Wien, Höchstädtplatz 6        0699 12345674
## 2 Hugo H. Wolf 1190 Wien, An den langen Lüssen 47 +43 4545 45454
```

3 Missing data

Explicitly missing data is present in the data file with a special entry for a missing data point. Implicitly missing data is not present in the data file (absent data points).

Inspect the file:

```
read_lines("data/text3.txt")
```

```
##  [1] "Class Sex    Age    Died  Survived         "
##  [2] "1st   Male   Child  0      5"
##  [3] "             Adult  118    57"
##  [4] "      Female Child  0      1 "
##  [5] "             Adult  4      140"
##  [6] "2nd   Male   Child  0      11"
##  [7] "             Adult  154    14"
##  [8] "      Female Child  0      13"
##  [9] "             Adult  13     80"
## [10] "3rd   Male   Child  35     13"
## [11] "             Adult  387    75"
## [12] "      Female Child  17     14"
## [13] "             Adult  89     76"
## [14] "Crew  Male   Child  0      ??"
## [15] "             Adult  670    192"
## [16] "      Female Child  0      ??"
## [17] "             Adult  3      20"
```

The file `text3.txt` contains explicitly missing data via `??` entries:

```
data1 <- read_table("data/text3.txt", col_names = TRUE, na = "??" )
data <- read_fwf(file="data/text3.txt",
                na = "??",
                skip = 1,
                col_positions = fwf_empty(file="data/text3.txt",
                                           col_names = names(data1))
                )
data
```

```
## # A tibble: 16 x 5
##    Class  Sex      Age    Died Survived
##    <chr>  <chr>    <chr> <dbl>    <dbl>
##  1 "1st"  "Male"   Child     0        5
##  2 ""     ""       Adult   118       57
##  3 ""     "Female" Child     0        1
##  4 ""     ""       Adult     4      140
##  5 "2nd"  "Male"   Child     0       11
##  6 ""     ""       Adult   154       14
##  7 ""     "Female" Child     0       13
##  8 ""     ""       Adult    13       80
##  9 "3rd"  "Male"   Child    35       13
## 10 ""     ""       Adult   387       75
## 11 ""     "Female" Child    17       14
## 12 ""     ""       Adult    89       76
## 13 "Crew" "Male"   Child     0       NA
## 14 ""     ""       Adult   670      192
## 15 ""     "Female" Child     0       NA
## 16 ""     ""       Adult     3       20
```

3.1 Find missing data (rows)

```
data %>% filter(!complete.cases(.)) # find all rows with missings
```

```
## # A tibble: 2 x 5
##   Class  Sex    Age    Died Survived
##   <chr>  <chr>  <chr> <dbl>    <dbl>
## 1 "Crew" Male   Child     0       NA
## 2 ""     Female Child     0       NA
```

3.2 Drop missing data (rows)

```
data %>% drop_na() ## either drop them ...
```

```
## # A tibble: 14 x 5
##    Class Sex      Age    Died Survived
##    <chr> <chr>    <chr> <dbl>    <dbl>
##  1 "1st" "Male"   Child     0        5
##  2 ""    ""       Adult   118       57
##  3 ""    "Female" Child     0        1
##  4 ""    ""       Adult     4      140
##  5 "2nd" "Male"   Child     0       11
##  6 ""    ""       Adult   154       14
##  7 ""     "Female" Child     0       13
##  8 ""     ""       Adult    13       80
##  9 "3rd"  "Male"   Child    35       13
## 10 ""     ""       Adult   387       75
## 11 ""     "Female" Child    17       14
## 12 ""     ""       Adult    89       76
## 13 ""     ""       Adult   670      192
## 14 ""     ""       Adult     3       20
```

3.3 Replace missing data points

```
data %>% mutate(Survived = replace_na(Survived, 0)) ## ... or replace them
```

```
## # A tibble: 16 x 5
##    Class  Sex      Age    Died Survived
##    <chr>  <chr>    <chr> <dbl>    <dbl>
##  1 "1st"  "Male"   Child     0        5
##  2 ""     ""       Adult   118       57
##  3 ""     "Female" Child     0        1
##  4 ""     ""       Adult     4      140
##  5 "2nd"  "Male"   Child     0       11
##  6 ""     ""       Adult   154       14
##  7 ""     "Female" Child     0       13
##  8 ""     ""       Adult    13       80
##  9 "3rd"  "Male"   Child    35       13
## 10 ""     ""       Adult   387       75
## 11 ""     "Female" Child    17       14
## 12 ""     ""       Adult    89       76
## 13 "Crew" "Male"   Child     0        0
## 14 ""     ""       Adult   670      192
## 15 ""     "Female" Child     0        0
## 16 ""     ""       Adult     3       20
```

3.4 Converting entries into missing values

```
data <- data %>%
  mutate_all(na_if, "")
data
```

```
## # A tibble: 16 x 5
##    Class Sex    Age    Died Survived
##    <chr> <chr>  <chr> <dbl>    <dbl>
##  1 1st   Male   Child     0        5
##  2 <NA>  <NA>   Adult   118       57
##  3 <NA>  Female Child     0        1
##  4 <NA>  <NA>   Adult     4      140
##  5 2nd   Male   Child     0       11
##  6 <NA>  <NA>   Adult   154       14
##  7 <NA>  Female Child     0       13
##  8 <NA>  <NA>   Adult    13       80
##  9 3rd   Male   Child    35       13
## 10 <NA>  <NA>   Adult   387       75
## 11 <NA>  Female Child    17       14
## 12 <NA>  <NA>   Adult    89       76
## 13 Crew  Male   Child     0       NA
## 14 <NA>  <NA>   Adult   670      192
## 15 <NA>  Female Child     0       NA
## 16 <NA>  <NA>   Adult     3       20
```

3.5 Fill missing data points

In `text3.txt` for the first two columns the entry in the previous row of the same column applies. The entries in the columns `Class` and `Sex` can be filled with the values from the previous row (`direction = "down"`):

```
data <- data %>%
  fill(Class, Sex, .direction = "down")
data
```

```
## # A tibble: 16 x 5
##    Class Sex    Age    Died Survived
##    <chr> <chr>  <chr> <dbl>    <dbl>
##  1 1st   Male   Child     0        5
##  2 1st   Male   Adult   118       57
##  3 1st   Female Child     0        1
##  4 1st   Female Adult     4      140
##  5 2nd   Male   Child     0       11
##  6 2nd   Male   Adult   154       14
##  7 2nd   Female Child     0       13
##  8 2nd   Female Adult    13       80
##  9 3rd   Male   Child    35       13
## 10 3rd   Male   Adult   387       75
## 11 3rd   Female Child    17       14
## 12 3rd   Female Adult    89       76
## 13 Crew  Male   Child     0       NA
## 14 Crew  Male   Adult   670      192
## 15 Crew  Female Child     0       NA
## 16 Crew  Female Adult     3       20
```

4 Import from Excel

4.1 Required packages

```
library(tidyverse)
library(readxl)
```

4.2 Get sheet names

First, get the sheet names:

```
path = "data/EU_Stockmarkets.xlsx"
sheets = excel_sheets(path)
sheets
```

```
## [1] "1991" "1992" "1993" "1994" "1995" "1996" "1997" "1998"
```

4.3 Import sheets

Try one sheet:

```
read_excel(path, sheet = "1991")
```

```
## # A tibble: 131 x 5
##    ...1    DAX   SMI   CAC  FTSE
##    <chr> <dbl> <dbl> <dbl> <dbl>
##  1 1     1629. 1678. 1773. 2444.
##  2 2     1614. 1688. 1750. 2460.
##  3 3     1607. 1679. 1718  2448.
##  4 4     1621. 1684. 1708. 2470.
##  5 5     1618. 1687. 1723. 2485.
##  6 6     1611. 1672. 1714. 2467.
##  7 7     1631. 1683. 1734. 2488.
##  8 8     1640. 1704. 1757. 2508.
##  9 9     1635. 1698. 1754  2510.
## 10 10    1646. 1716. 1754. 2497.
## # ... with 121 more rows
```

Loop over all sheets:

```
data = lapply(sheets, read_excel, path = path)
names(data) = sheets
```

Combine subtables. Add sheet names as extra column. Fix name of index column.

```
data = bind_rows(data, .id = "Year") %>%
  rename("Period" = "...1") %>%
  mutate(Period = parse_integer(Period),
         Year = parse_integer(Year))
data
```

```
## # A tibble: 1,867 x 6
##     Year Period   DAX   SMI   CAC  FTSE
##    <int>  <int> <dbl> <dbl> <dbl> <dbl>
##  1  1991      1 1629. 1678. 1773. 2444.
##  2  1991      2 1614. 1688. 1750. 2460.
##  3  1991      3 1607. 1679. 1718  2448.
##  4  1991      4 1621. 1684. 1708. 2470.
##  5  1991      5 1618. 1687. 1723. 2485.
##  6  1991      6 1611. 1672. 1714. 2467.
##  7  1991      7 1631. 1683. 1734. 2488.
##  8  1991      8 1640. 1704. 1757. 2508.
##  9  1991      9 1635. 1698. 1754  2510.
## 10  1991     10 1646. 1716. 1754. 2497.
## # ... with 1,857 more rows
```