Data Imports

Spring 23

April 13 2023

Clean messy column names with janitor::clean_names()

```
library(tidyverse)
library(janitor)
bar <- tibble("First Name" = c("Yi", "Do"),
              "last init" = c("C", "R"),
              "% in" = c(0.1, 0.5),
              "ñ$$$" = 1:2,
              " " = 3:2,
              " hi" = c("a", "b"),
              "null" = c(NA, NA))
bar
# A tibble: 2 × 7
  `First Name` `last init` `% in` `ñ$$$`   ` ` ` hi` null
  <chr>        <chr>        <dbl>  <int> <int> <chr> <lgl>
1 Yi           C              0.1      1     3 a     NA
2 Do           R              0.5      2     2 b     NA
```

```
cleaned_bar <- bar %>%
  clean_names() %>%
  remove_empty(c("rows", "cols"))
cleaned_bar
# A tibble: 2 × 6
  first_name last_init percent_in     n     x hi
  <chr>      <chr>          <dbl> <int> <int> <chr>
1 Yi         C                0.1     1     3 a
2 Do         R                0.5     2     2 b
```

Putting It All Together

```
foo %>%
  left_join(cleaned_bar,
            by = c("name" = "first_name")) %>%
  drop_na(height) %>%
  mutate(years = replace_na(years, 0),
         height = round(height, 1)) %>%
  select(id, name, height, years, last_init)
```

```
# A tibble: 2 × 5
     id name  height years last_init
  <int> <chr>  <dbl> <dbl> <chr>
1     3 Yi       3      50 C
2     5 Do       2.8     0 R
```

Working Directories

The working directory is where R looks for files and saves files by default.

```r
getwd() # see working directory
setwd() # change your working directory
```

To set working directory to your STAT 220 course folder

```r
setwd("path/to/stat220-folder/")  # set
getwd()     # check
```

Useful Terminal Commands:

```
$ cd   # change directory
$ ls   # unix command to list files
$ pwd  # present working directory
$ grep  # search for patterns in files
$ mkdir # create a new directory
$ mv # move or rename files or directories
```

Web imports

To your working environment:

```
url <- "https://raw.githubusercontent.com/deepbas/statdatasets/main/murders.csv"
dat <- read.csv(url)
```

To download file to working folder:

```
download.file(url, "murders.csv")
```

Reading and Writing Files: In base R

Reading CSV files:

```
data <- read.csv("data.csv")
```

Writing CSV files:

```
write.csv(data, "output.csv", row.names = FALSE)
```

- Other R-base import functions
  - `read.table()`
  - `read.delim()`

- Generate data frames rather than tibbles
- Character variables are converted to factors
  - Can be avoided by setting the argument `stringsAsFactors=FALSE`

readr package

- readr is a part of tidyverse library
- Includes functions for reading data stored in text file spreadsheets into R.
- Functions in the package include read_csv(), read_tsv(), read_delim() and more.
- These differ by the delimiter they use to split columns.

readr functions

function	reads
read_csv()	Comma separated values
read_csv2()	Semi-colon separated values
read_delim()	General delimited files
read_fwf()	Fixed width files
read_log()	Apache log files
read_table()	Space separated
read_tsv()	Tab delimited values

Basic syntax

All readr functions share a common syntax

```
df <- read_csv(file = "path/to/file.csv", ...)
```

Advantages of `readr`

`readr` functions are:

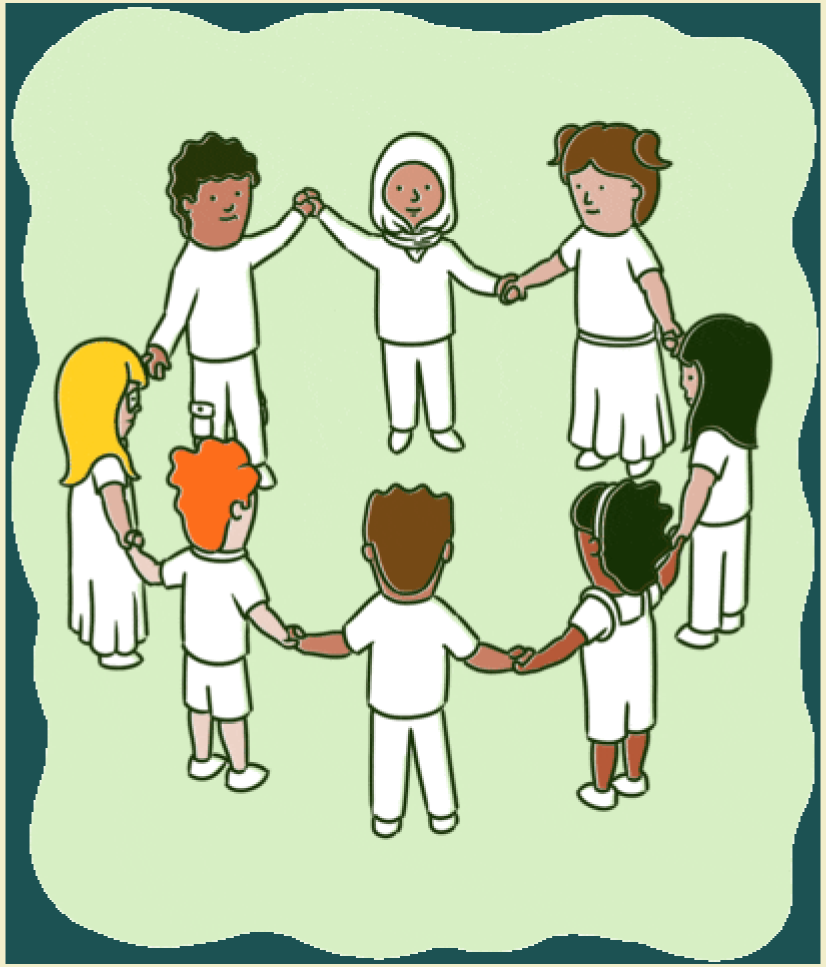
- ~ 10 times faster
- Return tibbles
- Have more intuitive defaults.
- No row names, no strings as factors.

Data frames and tibbles Conversion

- `as_tibble()` - convert a data frame to a tibble
- `as.data.frame()` - convert a tibble to a data frame

GROUP ACTIVITY 1

05:00



- Let's go over to maize server/ local Rstudio and our class moodle
- Get the class activity 10.Rmd file
- Skim through problem 1

Did it work as expected?

```
Rows: 549
Columns: 16
$ series                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ episode                <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, …
$ baker                  <chr> "Annetha", "David", "Edd", "Jasminder", "Jonatha…
$ technical              <chr> "2nd", "3rd", "1st", "N/A", "9th", "N/A", "8th",…
$ result                 <chr> "IN", "IN", "IN", "IN", "IN", "IN", "IN", "IN", …
$ uk_airdate             <chr> "17 August 2010", "17 August 2010", "17 August 2…
$ us_season              <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …
$ us_airdate             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …
$ showstopper_chocolate <chr> "chocolate", "chocolate", "no chocolate", "no ch…
$ showstopper_dessert   <chr> "other", "other", "other", "other", "other", "ca…
$ showstopper_fruit     <chr> "no fruit", "no fruit", "no fruit", "no fruit", …
$ showstopper_nut       <chr> "no nut", "no nut", "no nut", "no nut", "almond"…
$ signature_chocolate   <chr> "no chocolate", "chocolate", "no chocolate", "no…
$ signature_dessert     <chr> "cake", "cake", "cake", "cake", "cake", "cake", …
$ signature_fruit       <chr> "no fruit", "fruit", "fruit", "fruit", "fruit", …
$ signature_nut         <chr> "no nut", "no nut", "no nut", "no nut", "no nut"…
```

We want `technical` to be numerical and `uk_airdate` to be date

The `col_types` argument

By default, looks at first 1000 rows to guess variable data types (`guess_max`)

```
desserts <- read_csv(
  "https://raw.githubusercontent.com/deepbas/statdatasets/main/desserts
  col_types = list(
    technical = col_number(),
    uk_airdate = col_date()
  )
)
```

Looking for `problems`

List of potential problems parsing the file

```
problems(desserts)
# A tibble: 556 × 5
     row   col expected        actual         file
   <int> <int> <chr>           <chr>          <chr>
 1     2     6 date in ISO8601 17 August 2010 ""
 2     3     6 date in ISO8601 17 August 2010 ""
 3     4     6 date in ISO8601 17 August 2010 ""
 4     5     4 a number        N/A            ""
 5     5     6 date in ISO8601 17 August 2010 ""
 6     6     6 date in ISO8601 17 August 2010 ""
 7     7     4 a number        N/A            ""
 8     7     6 date in ISO8601 17 August 2010 ""
 9     8     6 date in ISO8601 17 August 2010 ""
10     9     4 a number        N/A            ""
# … with 546 more rows
```

Date formatting

```
# A tibble: 556 × 5
    row   col expected        actual         file
  <int> <int> <chr>           <chr>          <chr>
1     2     6 date in ISO8601 17 August 2010 ""
2     3     6 date in ISO8601 17 August 2010 ""
3     4     6 date in ISO8601 17 August 2010 ""
4     5     4 a number        N/A            ""
5     5     6 date in ISO8601 17 August 2010 ""
# … with 551 more rows
```

ISO8601 format: 2021-10-04

What we have: 17 August 2010

Adding `format` instructions

```
desserts <- read_csv(
  "https://raw.githubusercontent.com/deepbas/statdatasets/main/desserts
    col_types = list(
    technical = col_number(),
    uk_airdate = col_date(format = "%d %B %Y")
  )
)
```

- Year: "%Y" (4 digits). "%y" (2 digits)
- Month: "%m" (2 digits), "%b" (abbreviated name in current locale), "%B" (full name in current locale).
- Day: "%d" (2 digits), "%e" (optional leading space)

Looking for more `problems`

List of potential problems parsing the file

```
problems(desserts)
# A tibble: 7 × 5
    row   col expected actual file 
  <int> <int> <chr>    <chr>  <chr>
1     5     4 a number N/A    ""   
2     7     4 a number N/A    ""   
3     9     4 a number N/A    ""   
4    11     4 a number N/A    ""   
5    35     4 a number N/A    ""   
6    36     4 a number N/A    ""   
7    37     4 a number N/A    ""   
```

Addressing missing values

By default `na = c("", "NA")` are the recognized missing values

```
desserts <- read_csv(
  "https://raw.githubusercontent.com/deepbas/statdatasets/main/desserts.csv",
  col_types = list(
    technical = col_number(),
    uk_airdate = col_date(format = "%d %B %Y")
  ),
  na = c("", "NA", "N/A")
)
```

No more `problems`

```
problems(desserts)
# A tibble: 0 × 5
# … with 5 variables: row <int>, col <int>, expected <chr>, actual <chr>,
#   file <chr>
```

The Dataset

```
# A tibble: 549 × 16
   series episode baker     techn…¹ result uk_airdate us_se…² us_airdate shows…³
    <dbl>   <dbl> <chr>       <dbl> <chr>  <date>       <dbl> <date>     <chr>  
 1      1       1 Annetha         2 IN     2010-08-17      NA NA         chocol…
 2      1       1 David           3 IN     2010-08-17      NA NA         chocol…
 3      1       1 Edd             1 IN     2010-08-17      NA NA         no cho…
 4      1       1 Jasminder      NA IN     2010-08-17      NA NA         no cho…
 5      1       1 Jonathan        9 IN     2010-08-17      NA NA         no cho…
 6      1       1 Louise         NA IN     2010-08-17      NA NA         chocol…
 7      1       1 Miranda         8 IN     2010-08-17      NA NA         chocol…
 8      1       1 Ruth           NA IN     2010-08-17      NA NA         chocol…
 9      1       1 Lea            10 OUT    2010-08-17      NA NA         chocol…
10      1       1 Mark           NA OUT    2010-08-17      NA NA         chocol…
# … with 539 more rows, 7 more variables: showstopper_dessert <chr>,
#   showstopper_fruit <chr>, showstopper_nut <chr>, signature_chocolate <chr>,
#   signature_dessert <chr>, signature_fruit <chr>, signature_nut <chr>, and
#   abbreviated variable names ¹technical, ²us_season, ³showstopper_chocolate
```

Column casting functions

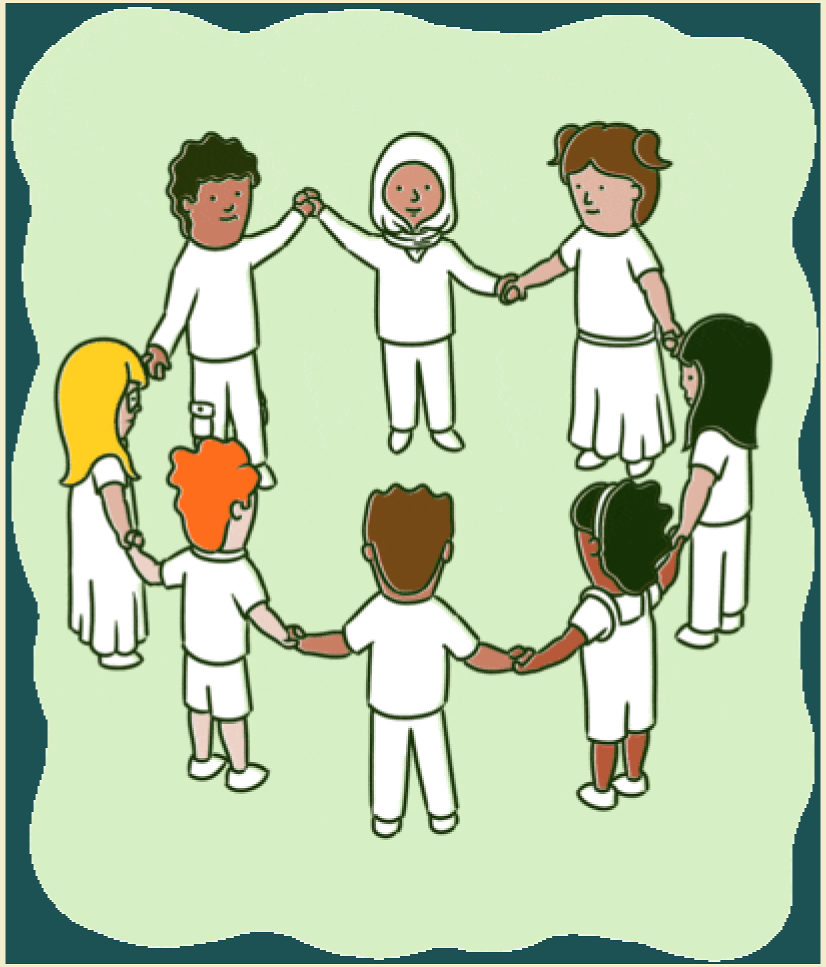
Type	`dplyr::glimpse()`	`readr::col_*()`
logical	`<lgl>`	`col_logical`
numeric	`<int>` or `<dbl>`	`col_number`
character	`<chr>`	`col_character`
factor	`<fct>`	`col_factor`
date	`<date>`	`col_date`

?read_csv

```
read_csv(file,
         col_names = TRUE,
         col_types = NULL,
         locale = default_locale(),
         na = c("", "NA"),
         quoted_na = TRUE,
         quote = "\"",
         comment = "",
         trim_ws = TRUE,
         skip = 0,
         n_max = Inf,
         guess_max = min(1000, n_max),
         progress = show_progress())
```

GROUP ACTIVITY 2

10:00



- ***Work on problem 2 to fix some messy data***
- ***Ask me questions***