Data Imports

Fall 2022

October 03 2022

Working Directories

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

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

To set working directory to your STAT 220 course folder

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

Useful Terminal Commands:

```
$ cd   # change directory
$ ls   # unix command to list files
$ pwd  # present working directory
```

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")
```

readr

- `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", ...)
```

Base R Imports

- R-base import functions
  - `read.csv()`
  - `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`

Advantages of `readr`

`readr` functions are:

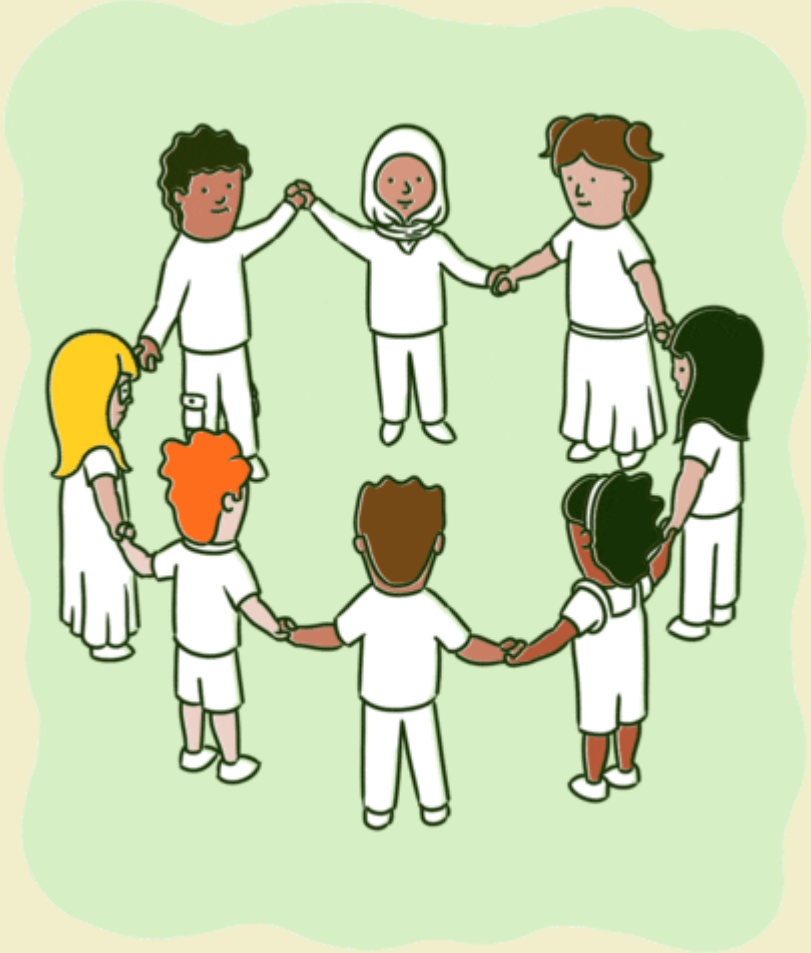
- ~ 10 times faster
- Return tibbles
- Have more intuitive defaults.
- No row names, no strings as factors.

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

10:00



- 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.csv",
  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.csv",
    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

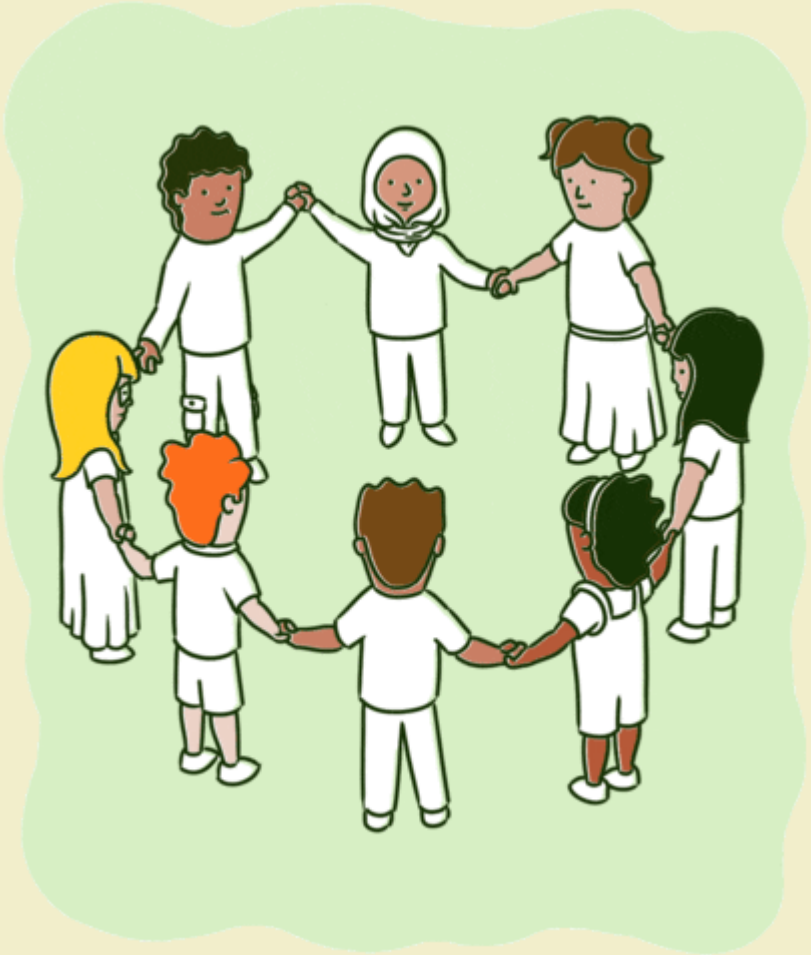
Type	`dplyr::glimpse()`	`readr::col_*()`
logical	`<lgl>`	`col_logical`
numeric	`<int>` or `<dbl>`	`col_number`
character	`<chr>`	`col_character`
factor	`<fct>`	`col_factor`
date	`<date>`	`col_date`

?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

Another example

```
energy <- read_csv("https://raw.githubusercontent.com/deepbas/statdatasets/main/energy.csv",
                   col_type = cols(
                    .default = col_double(),
                     Timestamp = col_datetime(format = ""),
                     dayWeek = col_factor(levels=c("Mon","Tues","Wed","Thurs","Fri","Sat","
                    ))
```

```
energy %>% slice_max(n=2, order_by = Timestamp)
# A tibble: 2 × 90
  Timestamp            year month weekOfYear dayOfMonth dayWeek timeHour timeM…¹
  <dttm>              <dbl> <dbl>      <dbl>      <dbl> <fct>       <dbl>   <dbl>
1 2016-08-31 23:45:00  2016     8         35         31 Wed            23      45
2 2016-08-31 23:30:00  2016     8         35         31 Wed            23      30
# … with 82 more variables: `100_Nevada_Street` <dbl>, `104_Maple_St.` <dbl>,
#   `106_Winona_St.` <dbl>, Allen_House <dbl>,
#   `Alumni_Guest_House/Johnson_House` <dbl>, Arboretum_Office <dbl>,
#   Art_Studios <dbl>, Benton_House <dbl>, Berg_House <dbl>, Bird_House <dbl>,
#   Boliou_Memorial_Art_Bldg. <dbl>, Burton_Hall <dbl>,
#   `Cassat_Hall_/_James_Hall` <dbl>,
#   `Center_for_Mathematics_&_Computing` <dbl>, Chaney_House <dbl>, …
```

Wide to Long

```
energy_narrow <- energy %>%
  pivot_longer(names_to = "building", values_to = "energyKWH", cols = `100_Nevada_Street`:Wi
```

```
energy_narrow
# A tibble: 2,880,578 × 10
   Timestamp            year month weekOfYear dayOfMonth dayWeek timeH…¹ timeM…²
   <dttm>              <dbl> <dbl>      <dbl>      <dbl> <fct>      <dbl>   <dbl>
 1 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
 2 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
 3 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
 4 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
 5 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
 6 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
 7 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
 8 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
 9 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
10 2015-09-01 00:00:00  2015     9         35          1 Tues           0       0
# … with 2,880,568 more rows, 2 more variables: building <chr>,
#   energyKWH <dbl>, and abbreviated variable names ¹timeHour, ²timeMinute
```

More dates and times manipulation

```
energy$Timestamp[1]
[1] "2015-09-01 UTC"
## [1] "2015-09-01 UTC"
as.numeric(energy$Timestamp[1])
[1] 1441065600
## [1] 1441065600

# 5th timestamp
( stamp5 <- energy$Timestamp[5] )
[1] "2015-09-01 01:00:00 UTC"
```

```
mdy("1/4/2021")
[1] "2021-01-04"

dmy_hms("01/04/2020-01-30-23")
[1] "2020-04-01 01:30:23 UTC"
```

```
make_datetime(year = 2022,
              month = 1,
              day = 5,
              hour = 10,
              sec = 20)
[1] "2022-01-05 10:00:20 UTC"
```

Another example: duration using lubriate

```
top_dest <- flights %>%
  count(dest) %>%
  slice_max(n, n = 10)

flights %>%
  semi_join(top_dest) %>%
  mutate(sch_datetime = make_datetime(
    year = year, month = month,
    day = day, hour = hour,
    min = minute)
    ) %>%
  select(dest, sch_datetime) %>%
  group_by(dest) %>%
  arrange(sch_datetime) %>%
  mutate(
    diff1 = (lag(sch_datetime) %--%
                    sch_datetime)/minutes(1),
    diff2 = interval(lag(sch_datetime),
                    sch_datetime)/minutes(1)) %>%
  summarize(medianMins1 = median(diff1,
                                 na.rm=TRUE),
            medianMins2 = median(diff2,
                                 na.rm=TRUE))
```

```
# A tibble: 10 × 3
   dest  medianMins1 medianMins2
   <chr>       <dbl>       <dbl>
 1 ATL            15          15
 2 BOS            17          17
 3 CLT            18          18
 4 DCA            34          34
 5 FLL            24          24
 6 LAX            19          19
 7 MCO            20          20
 8 MIA            25          25
 9 ORD            15          15
10 SFO            20          20
```