ds4da Cookbook

Recipes for Success

Brendan Knapp and Christopher Callaghan

2020-09-30

Contents

Welcome

Test

`<- == !=`

```
test <- "face"
```

Preface

init

Part I

Setup

Chapter 1

R and RStudio

1.1 R

1.1.1 Installation

https://cran.r-project.org/

1.2 RStudio

1.2.1 Installation

https://rstudio.com/products/rstudio/download/

Part II

Reading and Writing Data

Chapter 2

Tabular Data

- Aliases:
  - Tabular files
  - Flat
  - Delimited
- Includes:
  - Comma-Separated Value (.csv)
  - Tab-Separated Value (.tsv)

2.1 Basics

```r
library(readr)
```

Here's some example data, modified from http://www.gapminder.org/data/

```
country,continent,year,lifeExp,pop,gdpPercap       # header/column names, separated by commas
Afghanistan,Asia,1952,28.801,8425333,779.4453145
Afghanistan,Asia,1957,30.332,9240934,820.8530296   # comma-separated values
Afghanistan,Asia,1962,31.997,10267083,853.10071
Afghanistan,Asia,1967,34.02,11537966,836.1971382
Afghanistan,Asia,1972,36.088,13079460,739.9811058
Afghanistan,Asia,1977,38.438,14880372,786.11336
Afghanistan,Asia,1982,39.854,12881816,978.0114388
Afghanistan,Asia,1987,40.822,13867957,852.3959448
```

```
csv_text <-
'country,continent,year,lifeExp,pop,gdpPercap
Afghanistan,Asia,1952,28.801,8425333,779.4453145
Afghanistan,Asia,1957,30.332,9240934,820.8530296
Afghanistan,Asia,1962,31.997,10267083,853.10071
Afghanistan,Asia,1967,34.02,11537966,836.1971382
Afghanistan,Asia,1972,36.088,13079460,739.9811058
Afghanistan,Asia,1977,38.438,14880372,786.11336
Afghanistan,Asia,1982,39.854,12881816,978.0114388
Afghanistan,Asia,1987,40.822,13867957,852.3959448'

csv_file <- tempfile(fileext = ".csv")
csv_file # a temporary file path
```

```
## [1] "/tmp/Rtmpu2PRwa/file5fe27f20420f.csv"
```

```
writeLines(text = csv_text, con = csv_file) # write `csv_text` to `csv_file`
```

```
read_csv(file = csv_file)
```

```
## Parsed with column specification:
## cols(
##   country = col_character(),
##   continent = col_character(),
##   year = col_double(),
##   lifeExp = col_double(),
##   pop = col_double(),
##   gdpPercap = col_double()
## )
```

```
## # A tibble: 8 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
## 7 Afghanistan Asia       1982    39.9 12881816      978.
## 8 Afghanistan Asia       1987    40.8 13867957      852.
```

You may encounter Tab-Delimited data where values are separated by `\t` instead of `,`. Instead of `readr::read_csv()`, we can use `readr::read_tsv()`.

```
tsv_text <-
'country\tcontinent\tyear\tlifeExp\tpop\tgdpPercap
Afghanistan\tAsia\t1952\t28.801\t8425333\t779.4453145
Afghanistan\tAsia\t1957\t30.332\t9240934\t820.8530296
Afghanistan\tAsia\t1962\t31.997\t10267083\t853.10071
Afghanistan\tAsia\t1967\t34.02\t11537966\t836.1971382
Afghanistan\tAsia\t1972\t36.088\t13079460\t739.9811058
Afghanistan\tAsia\t1977\t38.438\t14880372\t786.11336
Afghanistan\tAsia\t1982\t39.854\t12881816\t978.0114388
Afghanistan\tAsia\t1987\t40.822\t13867957\t852.3959448'

tsv_file <- tempfile(fileext = ".tsv")
writeLines(text = tsv_text, con = tsv_file)
```

```
read_tsv(file = tsv_file)
```

```
## Parsed with column specification:
## cols(
##   country = col_character(),
##   continent = col_character(),
##   year = col_double(),
##   lifeExp = col_double(),
##   pop = col_double(),
##   gdpPercap = col_double()
## )
```

```
## # A tibble: 8 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
## 7 Afghanistan Asia       1982    39.9 12881816      978.
## 8 Afghanistan Asia       1987    40.8 13867957      852.
```

If we find ourselves reading delmited data that uses something other than `\t` or `,` to separate values, we can use `readr::read_delim()`.

```
pipe_separated_values_text <-
'country|continent|year|lifeExp|pop|gdpPercap
Afghanistan|Asia|1952|28.801|8425333|779.4453145
```

```
Afghanistan|Asia|1957|30.332|9240934|820.8530296
Afghanistan|Asia|1962|31.997|10267083|853.10071
Afghanistan|Asia|1967|34.02|11537966|836.1971382
Afghanistan|Asia|1972|36.088|13079460|739.9811058
Afghanistan|Asia|1977|38.438|14880372|786.11336
Afghanistan|Asia|1982|39.854|12881816|978.0114388
Afghanistan|Asia|1987|40.822|13867957|852.3959448'

psv_file <- tempfile(fileext = ".tsv")
writeLines(text = pipe_separated_values_text, con = psv_file)
```

```
read_delim(file = psv_file, delim = "|")
```

```
## Parsed with column specification:
## cols(
##   country = col_character(),
##   continent = col_character(),
##   year = col_double(),
##   lifeExp = col_double(),
##   pop = col_double(),
##   `gdpPercap     ` = col_double()
## )

## # A tibble: 8 x 6
##   country     continent  year lifeExp      pop `gdpPercap     `
##   <chr>       <chr>     <dbl>   <dbl>    <dbl>            <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333             779.
## 2 Afghanistan Asia       1957    30.3  9240934             821.
## 3 Afghanistan Asia       1962    32.0 10267083             853.
## 4 Afghanistan Asia       1967    34.0 11537966             836.
## 5 Afghanistan Asia       1972    36.1 13079460             740.
## 6 Afghanistan Asia       1977    38.4 14880372             786.
## 7 Afghanistan Asia       1982    39.9 12881816             978.
## 8 Afghanistan Asia       1987    40.8 13867957             852.
```

```
country,continent,year,lifeExp,pop,gdpPercap       # header/column names
Afghanistan,Asia,1952,28.801,8425333,779.4453145
Afghanistan,Asia,1957,30.332,9240934,820.8530296
Afghanistan,Asia,1962,31.997,10267083,853.10071
Afghanistan,Asia,1967,34.02,11537966,836.1971382
Afghanistan,Asia,1972,36.088,13079460,739.9811058
Afghanistan,Asia,1977,38.438,14880372,786.11336
Afghanistan,Asia,1982,39.854,12881816,978.0114388
Afghanistan,Asia,1987,40.822,13867957,852.3959448
Afghanistan,,,N/A,,                                 # notice that we're missing values
```

```
csv_text <-
'country,continent,year,lifeExp,pop,gdpPercap
Afghanistan,Asia,1952,28.801,8425333,779.4453145
Afghanistan,Asia,1957,30.332,9240934,820.8530296
Afghanistan,Asia,1962,31.997,10267083,853.10071
Afghanistan,Asia,1967,34.02,11537966,836.1971382
Afghanistan,Asia,1972,36.088,13079460,739.9811058
Afghanistan,Asia,1977,38.438,14880372,786.11336
Afghanistan,Asia,1982,39.854,12881816,978.0114388
Afghanistan,Asia,1987,40.822,13867957,852.3959448
Afghanistan,,,N/A,,'

csv_file <- tempfile(fileext = ".csv")
writeLines(text = csv_text, con = csv_file)
```

2.2 Common Pitfalls

2.2.1 Incorrect Column Types

```
data_frame_from_csv <- read_csv(file = csv_file)
```

```
## Parsed with column specification:
## cols(
##   country = col_character(),
##   continent = col_character(),
##   year = col_double(),
##   lifeExp = col_character(),
##   pop = col_double(),
##   gdpPercap = col_double()
## )
```

```
data_frame_from_csv
```

```
## # A tibble: 9 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <chr>       <chr>     <dbl> <chr>      <dbl>     <dbl>
## 1 Afghanistan Asia       1952 28.801   8425333      779.
## 2 Afghanistan Asia       1957 30.332   9240934      821.
## 3 Afghanistan Asia       1962 31.997  10267083      853.
## 4 Afghanistan Asia       1967 34.02   11537966      836.
## 5 Afghanistan Asia       1972 36.088  13079460      740.
## 6 Afghanistan Asia       1977 38.438  14880372      786.
## 7 Afghanistan Asia       1982 39.854  12881816      978.
## 8 Afghanistan Asia       1987 40.822  13867957      852.
## 9 Afghanistan <NA>         NA N/A           NA        NA
```

Notice that our `year` column says `<dbl>`, referring to it being of type `double`, yet all of our `year` values are whole numbers.

```
typeof(data_frame_from_csv$year)
```

```
## [1] "double"
```

```
data_frame_from_csv$year
```

```
## [1] 1952 1957 1962 1967 1972 1977 1982 1987   NA
```

We also have `"N/A"` in our `lifeExp` column, forcing R to interpret all `lifeExp` values as `characters` (`<chr>`).

```
typeof(data_frame_from_csv$lifeExp)
```

```
## [1] "character"
```

```
data_frame_from_csv$lifeExp
```

```
## [1] "28.801" "30.332" "31.997" "34.02"  "36.088" "38.438" "39.854" "40.822" "N/A"
```

2.2.1.1 Solution

```
read_csv(
  file = csv_file,
  col_types = cols(
    country = col_character(),
    continent = col_character(),
    year = col_integer(),        # read `year` as `integer`
    lifeExp = col_double(),      # read `lifeExp` as `double`
    pop = col_double(),
    gdpPercap = col_double()
  ),
  na = c("", "N/A")              # be explicit about how `csv_file` represents missing
)
```

```
## # A tibble: 9 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <chr>       <chr>     <int>   <dbl>    <dbl>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
## 7 Afghanistan Asia       1982    39.9 12881816      978.
## 8 Afghanistan Asia       1987    40.8 13867957      852.
## 9 Afghanistan <NA>         NA    NA         NA       NA
```

Part III

Data Frames

Chapter 3

Manipulating Data Frames

```r
library(tidyverse, warn.conflicts = FALSE)
```

```
## -- Attaching packages ------------------------------------------------------------------------
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ----------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
df <- tibble(
  group = c("a", "a", "b", "b", "b"),
  a = c(1, 4, NA, 3, 5),
  b = c(9, NA, 8, 10, 7),
  c = c(TRUE, FALSE, NA, FALSE, TRUE),
  d = c(LETTERS[1:3], NA, LETTERS[[5]]),
  e = factor(1:5, labels = c("tiny", "small", "medium", "big", "huge")),
  f_col = c(as.Date(NA), as.Date("2020-09-23") + c(3, 2, 1, 4)),
  g_col = c(as.POSIXct("2020-09-23 00:00:00") + 1:4 * 60 * 60 * 24 * 1.1, NA),
  col_h = list(c(1, 10), c(2, NA), c(3, 8), c(4, 7), c(5, 6)),
  col_i = list(NULL, pi, month.abb[6:10], iris, as.matrix(mtcars))
)

df
```

```
## # A tibble: 5 x 10
##   group     a     b c     d     e      f_col      g_col               col_h     col_
##   <chr> <dbl> <dbl> <lgl> <chr> <fct>  <date>     <dttm>              <list>    <li
## 1 a         1     9 TRUE  A     tiny   NA         2020-09-24 02:24:00 <dbl [2]> <NU
## 2 a         4    NA FALSE B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <db
## 3 b        NA     8 NA    C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <ch
## 4 b         3    10 FALSE <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df
## 5 b         5     7 TRUE  E     huge   2020-09-27 NA                  <dbl [2]> <db
```

```
glimpse(df)
```

```
## Rows: 5
## Columns: 10
## $ group <chr> "a", "a", "b", "b", "b"
## $ a     <dbl> 1, 4, NA, 3, 5
## $ b     <dbl> 9, NA, 8, 10, 7
## $ c     <lgl> TRUE, FALSE, NA, FALSE, TRUE
## $ d     <chr> "A", "B", "C", NA, "E"
## $ e     <fct> tiny, small, medium, big, huge
## $ f_col <date> NA, 2020-09-26, 2020-09-25, 2020-09-24, 2020-09-27
## $ g_col <dttm> 2020-09-24 02:24:00, 2020-09-25 04:48:00, 2020-09-26 07:12:00, 2020-0
## $ col_h <list> [<1, 10>, <2, NA>, <3, 8>, <4, 7>, <5, 6>]
## $ col_i <list> [NULL, 3.141593, <"Jun", "Jul", "Aug", "Sep", "Oct">, <data.frame[15
```

3.1 select() Columns

3.1.1 by Name

```
df %>%
  select(a)
```

```
## # A tibble: 5 x 1
##       a
##   <dbl>
## 1     1
## 2     4
## 3    NA
## 4     3
## 5     5
```

```
df %>%
  select(a, c, e)
```

```
## # A tibble: 5 x 3
##       a c     e
##   <dbl> <lgl> <fct>
## 1     1 TRUE  tiny
## 2     4 FALSE small
## 3    NA NA    medium
## 4     3 FALSE big
## 5     5 TRUE  huge
```

```
df %>%
  select(b, d, f_col)
```

```
## # A tibble: 5 x 3
##       b d     f_col
##   <dbl> <chr> <date>
## 1     9 A     NA
## 2    NA B     2020-09-26
## 3     8 C     2020-09-25
## 4    10 <NA>  2020-09-24
## 5     7 E     2020-09-27
```

```
df %>%
  select(b, c, everything())
```

```
## # A tibble: 5 x 10
##       b c     group     a d     e      f_col      g_col               col_h     col_i
##   <dbl> <lgl> <chr> <dbl> <chr> <fct>  <date>     <dttm>              <list>    <list>
## 1     9 TRUE  a         1 A     tiny   NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2    NA FALSE a         4 B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 3     8 NA    b        NA C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
## 4    10 FALSE b         3 <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x
## 5     7 TRUE  b         5 E     huge   2020-09-27 NA                  <dbl [2]> <dbl[,11] [32
```

```
df %>%
  select(b, c, everything(), -a)
```

```
## # A tibble: 5 x 9
##       b c     group d     e      f_col      g_col               col_h     col_i
##   <dbl> <lgl> <chr> <chr> <fct>  <date>     <dttm>              <list>    <list>
## 1      9 TRUE  a     A     tiny   NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2     NA FALSE a     B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 3      8 NA    b     C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
## 4     10 FALSE b     <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5]  [
## 5      7 TRUE  b     E     huge   2020-09-27 NA                  <dbl [2]> <dbl[,11]
```

```r
cols_to_select <- c("a", "c", "e")
df %>%
  select(all_of(cols_to_select))
```

```
## # A tibble: 5 x 3
##       a c     e
##   <dbl> <lgl> <fct>
## 1     1 TRUE  tiny
## 2     4 FALSE small
## 3    NA NA    medium
## 4     3 FALSE big
## 5     5 TRUE  huge
```

3.1.2 by Index

```r
df %>%
  select(1L)
```

```
## # A tibble: 5 x 1
##   group
##   <chr>
## 1 a
## 2 a
## 3 b
## 4 b
## 5 b
```

```r
df %>%
  select(1, 3, 5)
```

```
## # A tibble: 5 x 3
##   group     b d
##   <chr> <dbl> <chr>
## 1 a         9 A
## 2 a        NA B
## 3 b         8 C
## 4 b        10 <NA>
## 5 b         7 E
```

```
df %>%
  select(2, 4, 6)
```

```
## # A tibble: 5 x 3
##       a c     e
##   <dbl> <lgl> <fct>
## 1     1 TRUE  tiny
## 2     4 FALSE small
## 3    NA NA    medium
## 4     3 FALSE big
## 5     5 TRUE  huge
```

```
df %>%
  select(2:3, everything())
```

```
## # A tibble: 5 x 10
##       a     b group c     d     e      f_col      g_col               col_h     col_i
##   <dbl> <dbl> <chr> <lgl> <chr> <fct>  <date>     <dttm>              <list>    <list>
## 1     1     9 a     TRUE  A     tiny   NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2     4    NA a     FALSE B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 3    NA     8 b     NA    C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
## 4     3    10 b     FALSE <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x
## 5     5     7 b     TRUE  E     huge   2020-09-27 NA                  <dbl [2]> <dbl[,11] [32
```

```
df %>%
  select(2:3, everything(), -1)
```

```
## # A tibble: 5 x 9
##       a     b c     d     e      f_col      g_col               col_h     col_i
##   <dbl> <dbl> <lgl> <chr> <fct>  <date>     <dttm>              <list>    <list>
## 1     1     9 TRUE  A     tiny   NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2     4    NA FALSE B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 3    NA     8 NA    C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
## 4     3    10 FALSE <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x 5]>
## 5     5     7 TRUE  E     huge   2020-09-27 NA                  <dbl [2]> <dbl[,11] [32 x 11]>
```

```
cols_to_select <- c(1, 3, 5)
df %>%
  select(all_of(cols_to_select))
```

```
## # A tibble: 5 x 3
##   group     b d
##   <chr> <dbl> <chr>
## 1 a         9 A
## 2 a        NA B
## 3 b         8 C
## 4 b        10 <NA>
## 5 b         7 E
```

```
cols_to_select <- c(1, 3, 5, 1000)
df %>%
  select(any_of(cols_to_select))
```

```
## # A tibble: 5 x 3
##   group     b d
##   <chr> <dbl> <chr>
## 1 a         9 A
## 2 a        NA B
## 3 b         8 C
## 4 b        10 <NA>
## 5 b         7 E
```

3.1.3 by Name Pattern

`contains()` selects a column if *any* part of its name contains `match=`.

```
df %>%
  select(contains(match = "col"))
```

```
## # A tibble: 5 x 4
##   f_col      g_col               col_h     col_i
##   <date>     <dttm>              <list>    <list>
## 1 NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2 2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 3 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
## 4 2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x 5]>
## 5 2020-09-27 NA                  <dbl [2]> <dbl[,11] [32 x 11]>
```

`starts_with()` selects a column if its name starts with `match=`.

```
df %>%
  select(starts_with("col_"))
```

```
## # A tibble: 5 x 2
##   col_h     col_i
##   <list>    <list>
## 1 <dbl [2]> <NULL>
## 2 <dbl [2]> <dbl [1]>
## 3 <dbl [2]> <chr [5]>
## 4 <dbl [2]> <df[,5] [150 x 5]>
## 5 <dbl [2]> <dbl[,11] [32 x 11]>
```

`starts_with()` selects a column if its name ends with `match=`.

```
df %>%
  select(ends_with("_col"))
```

```
## # A tibble: 5 x 2
##   f_col      g_col
##   <date>     <dttm>
## 1 NA         2020-09-24 02:24:00
## 2 2020-09-26 2020-09-25 04:48:00
## 3 2020-09-25 2020-09-26 07:12:00
## 4 2020-09-24 2020-09-27 09:36:00
## 5 2020-09-27 NA
```

`matches()`s Selects a column if its name matches a regular expression pattern.

```
df %>%
  select(matches("(^\\w_)?col(_\\w)?"))
```

```
## # A tibble: 5 x 4
##   f_col      g_col               col_h     col_i
##   <date>     <dttm>              <list>    <list>
## 1 NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2 2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 3 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
## 4 2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x 5]>
## 5 2020-09-27 NA                  <dbl [2]> <dbl[,11] [32 x 11]>
```

3.1.4 by Data Type

```
df %>%
  select(where(is.factor))
```

```
## # A tibble: 5 x 1
##   e
##   <fct>
## 1 tiny
## 2 small
## 3 medium
## 4 big
## 5 huge
```

```
df %>%
  select_if(is.factor)
```

```
## # A tibble: 5 x 1
##   e
##   <fct>
## 1 tiny
## 2 small
## 3 medium
## 4 big
## 5 huge
```

```
df %>%
  select(where(is.factor), f_col)
```

```
## # A tibble: 5 x 2
##   e      f_col
##   <fct>  <date>
## 1 tiny   NA
## 2 small  2020-09-26
## 3 medium 2020-09-25
## 4 big    2020-09-24
## 5 huge   2020-09-27
```

```
df %>%
  select(a, !where(is.integer))
```

```
## # A tibble: 5 x 10
##       a group     b c     d     e      f_col      g_col               col_h     col_
##   <dbl> <chr> <dbl> <lgl> <chr> <fct>  <date>     <dttm>              <list>    <lis
## 1     1 a         9 TRUE  A     tiny   NA         2020-09-24 02:24:00 <dbl [2]> <NU
## 2     4 a        NA FALSE B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <db
## 3    NA b         8 NA    C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <ch
## 4     3 b        10 FALSE <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df
## 5     5 b         7 TRUE  E     huge   2020-09-27 NA                  <dbl [2]> <db
```

```r
df %>%
  select(where(is.character) | where(is.factor))
```

```
## # A tibble: 5 x 3
##   group d     e
##   <chr> <chr> <fct>
## 1 a     A     tiny
## 2 a     B     small
## 3 b     C     medium
## 4 b     <NA>  big
## 5 b     E     huge
```

```r
df %>%
  select(where(~ is.double(.) | is.list(.)))
```

```
## # A tibble: 5 x 6
##       a     b f_col      g_col               col_h     col_i
##   <dbl> <dbl> <date>     <dttm>              <list>    <list>
## 1     1     9 NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2     4    NA 2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 3    NA     8 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
## 4     3    10 2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x 5]>
## 5     5     7 2020-09-27 NA                  <dbl [2]> <dbl[,11] [32 x 11]>
```

```r
df %>%
  select_if(~ is.character(.x) | is.factor(.x))
```

```
## # A tibble: 5 x 3
##   group d     e
##   <chr> <chr> <fct>
## 1 a     A     tiny
## 2 a     B     small
## 3 b     C     medium
## 4 b     <NA>  big
## 5 b     E     huge
```

3.2 filter() Rows

3.2.1 by row_number()

```
df %>%
  filter(row_number() == 1)
```

```
## # A tibble: 1 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <list
## 1 a         1     9 TRUE  A     tiny  NA         2020-09-24 02:24:00 <dbl [2]> <NULL
```

```
df %>%
  filter(row_number() > 1)
```

```
## # A tibble: 4 x 10
##   group     a     b c     d     e      f_col      g_col               col_h     col_
##   <chr> <dbl> <dbl> <lgl> <chr> <fct>  <date>     <dttm>              <list>    <lis
## 1 a         4    NA FALSE B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl
## 2 b        NA     8 NA    C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr
## 3 b         3    10 FALSE <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df
## 4 b         5     7 TRUE  E     huge   2020-09-27 NA                  <dbl [2]> <dbl
```

3.2.2 by Name

```
df %>%
  filter(a == 2)
```

```
## # A tibble: 0 x 10
## # ... with 10 variables: group <chr>, a <dbl>, b <dbl>, c <lgl>, d <chr>, e <fct>, f
```

```
df %>%
  filter(a != 2)
```

```
## # A tibble: 4 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <list
## 1 a         1     9 TRUE  A     tiny  NA         2020-09-24 02:24:00 <dbl [2]> <NULL
## 2 a         4    NA FALSE B     small 2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl
## 3 b         3    10 FALSE <NA>  big   2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[
## 4 b         5     7 TRUE  E     huge  2020-09-27 NA                  <dbl [2]> <dbl
```

```
df %>%
  filter(c)
```

```
## # A tibble: 2 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <list>
## 1 a         1     9 TRUE  A     tiny  NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2 b         5     7 TRUE  E     huge  2020-09-27 NA                  <dbl [2]> <dbl[,11] [32 x
```

```
df %>%
  filter(!c)
```

```
## # A tibble: 2 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <list>
## 1 a         4    NA FALSE B     small 2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 2 b         3    10 FALSE <NA>  big   2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x
```

```
df %>%
  filter(a == 5, d == "E")
```

```
## # A tibble: 1 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <list>
## 1 b         5     7 TRUE  E     huge  2020-09-27 NA                  <dbl [2]> <dbl[,11] [32 x
```

```
df %>%
  filter(a >= 3 | f_col == "2020-09-24")
```

```
## # A tibble: 3 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <list>
## 1 a         4    NA FALSE B     small 2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 2 b         3    10 FALSE <NA>  big   2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x
## 3 b         5     7 TRUE  E     huge  2020-09-27 NA                  <dbl [2]> <dbl[,11] [32 x
```

```
df %>%
  filter(a < 2 | c)
```

```
## # A tibble: 2 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <list>
## 1 a         1     9 TRUE  A     tiny  NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2 b         5     7 TRUE  E     huge  2020-09-27 NA                  <dbl [2]> <dbl[,11] [32 x
```

```
df %>%
  filter(!is.na(a), !is.na(b), !is.na(d))
```

```
## # A tibble: 2 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <lis
## 1 a         1     9 TRUE  A     tiny  NA         2020-09-24 02:24:00 <dbl [2]> <NUL
## 2 b         5     7 TRUE  E     huge  2020-09-27 NA                  <dbl [2]> <dbl
```

3.2.3 by Type

```
df %>%
  filter(across(where(is.numeric), ~ .x >= 5))
```

```
## # A tibble: 1 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <lis
## 1 b         5     7 TRUE  E     huge  2020-09-27 NA                  <dbl [2]> <dbl
```

```
df %>%
  filter_if(is.numeric, ~ .x >= 5)
```

```
## # A tibble: 1 x 10
##   group     a     b c     d     e     f_col      g_col               col_h     col_
##   <chr> <dbl> <dbl> <lgl> <chr> <fct> <date>     <dttm>              <list>    <lis
## 1 b         5     7 TRUE  E     huge  2020-09-27 NA                  <dbl [2]> <dbl
```

```
df %>%
  filter_if(is.list, ~ map_lgl(.x, ~ !is.null(.x)))
```

```
## # A tibble: 4 x 10
##   group     a     b c     d     e      f_col      g_col               col_h     col
##   <chr> <dbl> <dbl> <lgl> <chr> <fct>  <date>     <dttm>              <list>    <li
## 1 a         4    NA FALSE B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <db
## 2 b        NA     8 NA    C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <ch
## 3 b         3    10 FALSE <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df
## 4 b         5     7 TRUE  E     huge   2020-09-27 NA                  <dbl [2]> <db
```

3.3 arrange() Rows

```
df %>%
  arrange(a)
```

```
## # A tibble: 5 x 10
##   group     a     b c     d     e      f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct>  <date>     <dttm>              <list>    <list>
## 1 a         1     9 TRUE  A     tiny   NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 2 b         3    10 FALSE <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x
## 3 a         4    NA FALSE B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 4 b         5     7 TRUE  E     huge   2020-09-27 NA                  <dbl [2]> <dbl[,11] [32
## 5 b        NA     8 NA    C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
```

```
df %>%
  arrange(desc(a))
```

```
## # A tibble: 5 x 10
##   group     a     b c     d     e      f_col      g_col               col_h     col_i
##   <chr> <dbl> <dbl> <lgl> <chr> <fct>  <date>     <dttm>              <list>    <list>
## 1 b         5     7 TRUE  E     huge   2020-09-27 NA                  <dbl [2]> <dbl[,11] [32
## 2 a         4    NA FALSE B     small  2020-09-26 2020-09-25 04:48:00 <dbl [2]> <dbl [1]>
## 3 b         3    10 FALSE <NA>  big    2020-09-24 2020-09-27 09:36:00 <dbl [2]> <df[,5] [150 x
## 4 a         1     9 TRUE  A     tiny   NA         2020-09-24 02:24:00 <dbl [2]> <NULL>
## 5 b        NA     8 NA    C     medium 2020-09-25 2020-09-26 07:12:00 <dbl [2]> <chr [5]>
```