Stat. 450 Section 1 or 2: Homework 7

Prof. Eric A. Suess

So how should you complete your homework for this class?

- First thing to do is type all of your information about the problems you do in the text part of your R Notebook.
- Second thing to do is type all of your R code into R chunks that can be run.
- If you load the tidyverse in an R Notebook chunk, be sure to include the "message = FALSE" in the {r}, so {r message = FALSE}.
- Last thing is to spell check your R Notebook. Edit > Check Spelling... or hit the F7 key.

Homework 7:

```
Read: Chapter 9, Chapter 10, Chapter 11
Do 10.5 Exercises 1, 2
Do 11.2.2 Exercise 2
Do 11.3.5 Exercises 6, 7
```

```r
library(tidyverse)
```

10.5

1.

At the Console, all of the variables are printed out. Note the labeling of the rows.

In a notebook data.frames are printed in the same way as a tibble, but the row labels are not printed.

You can use is_tibble() and class() functions to check what a data.frame is.

```r
library(tidyverse)

is_tibble(mtcars)
```

```
## [1] FALSE
```

```r
class(mtcars)
```

```
## [1] "data.frame"
```

```r
mtcars
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
```

```
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```r
as.tibble(mtcars)
```

```
## # A tibble: 32 x 11
##      mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##  * <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  21       6  160    110  3.9   2.62  16.5     0     1     4     4
##  2  21       6  160    110  3.9   2.88  17.0     0     1     4     4
##  3  22.8     4  108     93  3.85  2.32  18.6     1     1     4     1
##  4  21.4     6  258    110  3.08  3.22  19.4     1     0     3     1
##  5  18.7     8  360    175  3.15  3.44  17.0     0     0     3     2
##  6  18.1     6  225    105  2.76  3.46  20.2     1     0     3     1
##  7  14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
##  8  24.4     4  147.    62  3.69  3.19  20       1     0     4     2
##  9  22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2
## 10  19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4
## # ... with 22 more rows
```

```r
library(nycflights13)

is_tibble(flights)
```

```
## [1] TRUE
```

```r
is_tibble(planes)
```

```
## [1] TRUE
```

```r
is_tibble(airports)
```

```
## [1] TRUE
```

```r
is_tibble(weather)
```

```
## [1] TRUE
```

```r
class(flights)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

2.

The main thing that is different is that with data.frame the reference to the variable can use only the first letter, the rest are assumed. This could lead to problems because more than one variable name may start with the same letter.

The tibble returns a tibble all of the time, regardless of selecting one column or more than one column. In a data.frame if a single column is selected, a vector is returned, otherwise a data.frame is retured. This behavior could cause problems.

```r
df <- data.frame(abc = 1, xyz = "a")
df$x
```

```
## [1] a
## Levels: a
```

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

```
## [1] a
## Levels: a
```

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

```
##   abc xyz
## 1   1   a
```

Converting the data.frame to a tibble.

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

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

```
## NULL
```

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

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

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

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

11.2.2

2.

Read the help files, it appears they have all of the same options.

- col_names = TRUE
- col_types = NULL
- locale = default_locale()
- na = c("", "NA")
- quoted_na = TRUE
- quote = "\"
- trim_ws = TRUE
- n_max = Inf
- guess_max = min(1000, n_max)
- progress = show_progress()

```
?read_csv
?read_tsv

union(names(formals(read_csv)), names(formals(read_tsv)))
```

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

```
intersect(names(formals(read_csv)), names(formals(read_tsv)))
```

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

11.3.5

6.

These solutions are from the R for Data Science Solutions. Note the problem number has changed.

UTF-8 is standard now, and ASCII has been around forever.

For Asian languages Arabic and Vietnamese have ISO and Windows standards. The other major Asian scripts have their own:

- Japanese: JIS X 0208, Shift JIS, ISO-2022-JP
- Chinese: GB 2312, GBK, GB 18030
- Korean: KS X 1001, EUC-KR, ISO-2022-KR

7.

Generate the correct format strings.

```
d1 <- "January 1, 2010"
d2 <- "2015-Mar-07"
d3 <- "06-Jun-2017"
d4 <- c("August 19 (2015)", "July 1 (2015)")
```

```r
d5 <- "12/30/14" # Dec 30, 2014
t1 <- "1705"
t2 <- "11:15:10.12 PM"
```

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

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

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

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

```r
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
```

```r
parse_time(t2, "%H:%M:%OS %p")
```

```
## 23:15:10.12
```