# Lecture Assignment 10

Viraj Vijaywargiya

2022-05-03

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

**Part 10.5**

**Question 1**

```
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
```

```
is_tibble(mtcars)
```

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

```
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
```

Tibbles have a refined print method that shows only the first 10 rows, and all the columns that fit on screen. Also, printing mtcars, which is not a tibble, shows the description of the data as "df[32 x 11]" where df means data frame. After converting the object to a tibble, using as_tibble(), printing it shows the description of the data as "A tibble: 32 x 11". Furthermore, by using is_tibble(), we can check if the object is a tibble. In this case, using is_tibble(mtcars) gives FALSE.

**Question 2**

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

```
## [1] "a"
```

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

```
## [1] "a"
```

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

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

```
tbl <- as_tibble(df)
tbl$x
```

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

```
## NULL
```

```
tbl[, "xyz"]
```

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

```
tbl[, c("abc", "xyz")]
```

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

Using the $ operator with data.frame matches any column name that starts with the name following it, therefore, df*xexapndstodf*xyz. This feature can be frustrating as you might end up using a different column than the one you expected. However, tibble is strict with these kind of issues as they never do partial matching, and they will generate a warning if the column you are trying to access does not exist. This prevents the user from using the wrong column. Moreover, using "[" with data.frame returns a type of object that differs on the number of columns. It will return a data.frame if there's more than one column, else, it will return a vector. This is also frustrating as what the code does depends on the length of the variable, and it would require us to write a code to handle such situations.

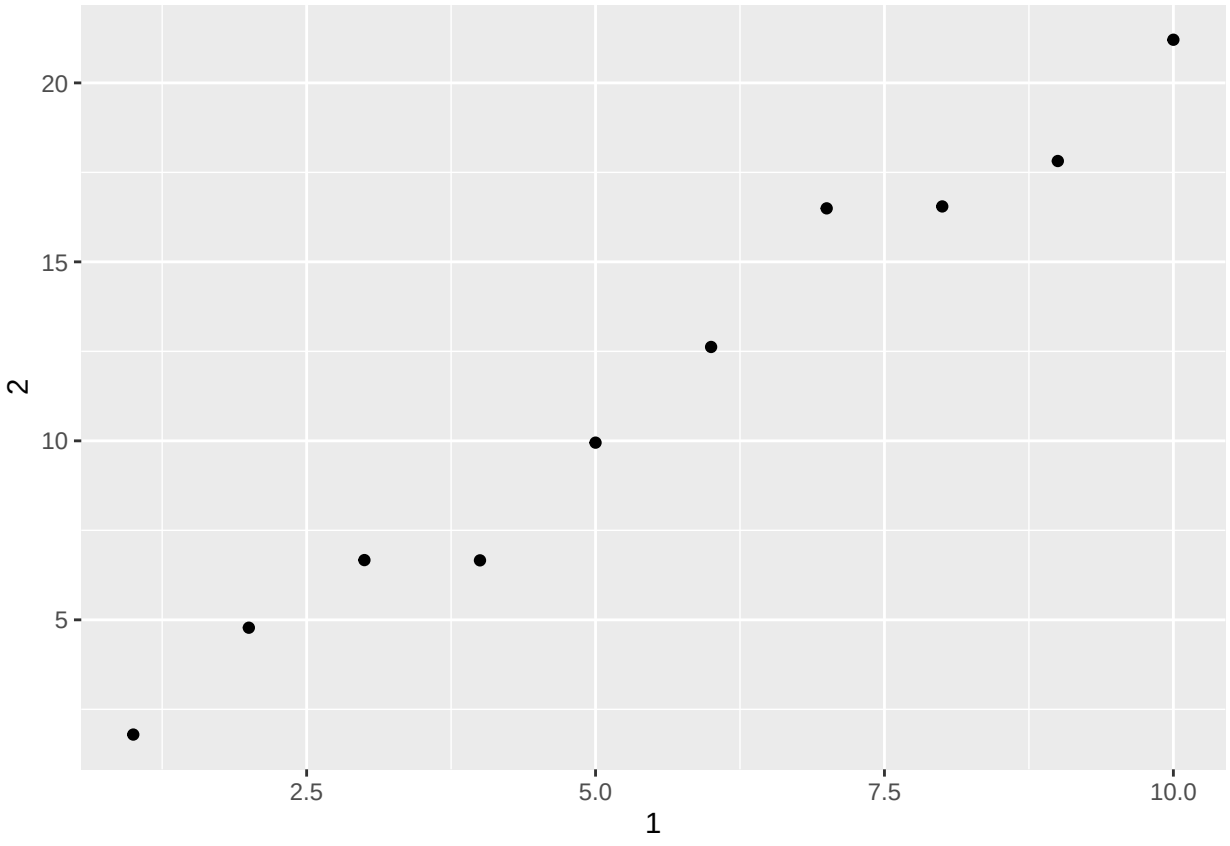
**Question 4**

```
annoying <- tibble(
  `1` = 1:10,
  `2` = `1` * 2 + rnorm(length(`1`))
)
```

```
annoying$`1`
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
ggplot(annoying, aes(x = `1`, y = `2`))+
  geom_point()
```



```
annoying <- mutate(annoying, `3` = `2`/`1`)
print(annoying)
```

```
## # A tibble: 10 x 3
##      `1`   `2`   `3`
##    <int> <dbl> <dbl>
##  1     1  1.79  1.79
##  2     2  4.78  2.39
##  3     3  6.67  2.22
##  4     4  6.66  1.67
##  5     5  9.95  1.99
##  6     6 12.6   2.10
##  7     7 16.5   2.36
##  8     8 16.5   2.07
##  9     9 17.8   1.98
## 10    10 21.2   2.12
```

```
annoying <- rename(annoying, one = `1`, two = `2`, three = `3`)
print(annoying)
```

```
## # A tibble: 10 x 3
##      one   two three
##    <int> <dbl> <dbl>
##  1     1  1.79  1.79
##  2     2  4.78  2.39
##  3     3  6.67  2.22
##  4     4  6.66  1.67
##  5     5  9.95  1.99
##  6     6 12.6   2.10
##  7     7 16.5   2.36
##  8     8 16.5   2.07
##  9     9 17.8   1.98
## 10    10 21.2   2.12
```

**Question 5**

enframe() converts named atomic vectors or lists to one- or two-column data frames. For a list, the result will be a nested tibble with a column of type list. For unnamed vectors, the natural sequence is used as name column. For example,

```
enframe(c(a = 10, b = 15))
```

```
## # A tibble: 2 x 2
##   name  value
##   <chr> <dbl>
## 1 a        10
## 2 b        15
```

**Question 6**

You can explicitly print() the data frame and control the number of rows (n) and the width of the display. Additional column names to be printed can be controlled using "print(width = )".