Exercise 10.5

Xiru Lyu

2/20/2018

```
# load in required package
library(tidyverse)
```

1. How can you tell if an object is a tibble?

```
# an object is a data frame
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
# an object is a tibble
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  21.0  6.00   160 110    3.90  2.62  16.5 0      1.00  4.00  4.00
##  2  21.0  6.00   160 110    3.90  2.88  17.0 0      1.00  4.00  4.00
##  3  22.8  4.00   108  93.0  3.85  2.32  18.6 1.00   1.00  4.00  1.00
##  4  21.4  6.00   258 110    3.08  3.22  19.4 1.00   0     3.00  1.00
##  5  18.7  8.00   360 175    3.15  3.44  17.0 0      0     3.00  2.00
##  6  18.1  6.00   225 105    2.76  3.46  20.2 1.00   0     3.00  1.00
##  7  14.3  8.00   360 245    3.21  3.57  15.8 0      0     3.00  4.00
##  8  24.4  4.00   147  62.0  3.69  3.19  20.0 1.00   0     4.00  2.00
##  9  22.8  4.00   141  95.0  3.92  3.15  22.9 1.00   0     4.00  2.00
## 10  19.2  6.00   168 123    3.92  3.44  18.3 1.00   0     4.00  4.00
## # ... with 22 more rows
```

If an object is a tibble, we would clearly see the first row of the output would say "A tibble:" while a data frame doesn't having this kind of label. Moreover, the third row of the tibble object indicates the class for each column in the table, but a data frame object doesn't have such labeling.

2. Compare and contrast the following operations on a `data.frame` and equivalent tibble. What is different? Why might the default data frame behaviours cause you frustration?

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

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

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

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

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

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

```
# operate on a 'tibble'
tb <- tibble(abc=1,xyz='a')
tb$x # this would generate a warning message since the column named 'x' doen't exist
```

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

```
## NULL
```

```
tb[['xyz']]
```

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

```
tb %>% select('abc','xyz')
```

```
## # A tibble: 1 x 2
##     abc xyz
##   <dbl> <chr>
## 1  1.00 a
```

`df$x` refers to the column named 'x' in the data frame. Even though there is no column with such a column name, this command would return a column that contains 'x'. However, if we have a tibble object, the same command would generate a warning message because it cannot find a column that exactly matches this name. This characteristic for the data frame object will be troublesome when we have columns that have very similar names, so that when mistyped something in the column name, it would return us unexpected results. The same thing wouldn't happen for a tibble object since it only searches for columns with exact matching names.

3. If you have the name of a variable stored in an object, e.g. `var <- "mpg"`, how can you extract the reference variable from a tibble?

There are several ways of doing so. For example, the name for the tibble is 'tb'. You can extract the reference variable using `tb$var`, `tb[[var]]`.

4. Practice referring to non-syntactic names in the following data frame by:

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

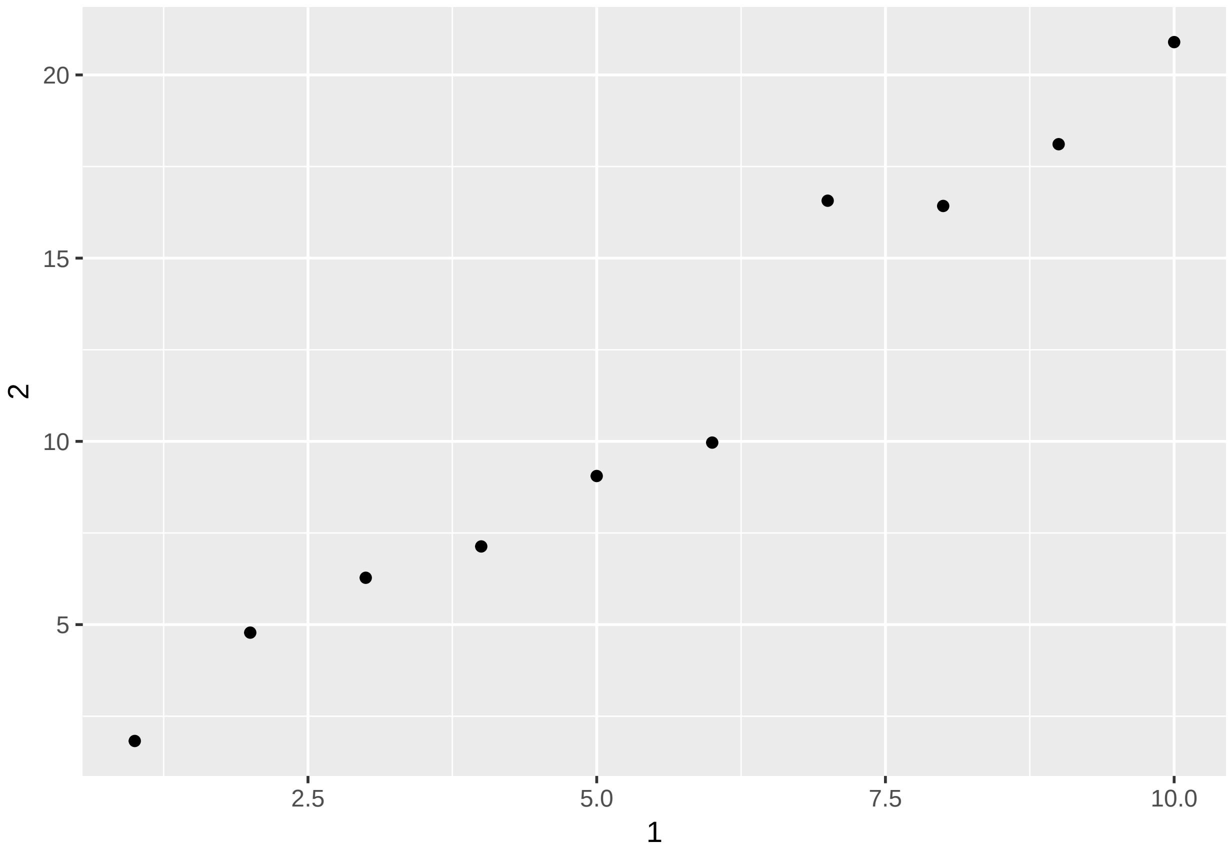
1. Extracting the variable called 1.

```
annoying$`1`
```

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

2. Plotting a scatterplot of 1 vs 2.

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



3. Creating a new column called 3 which is 2 divided by 1.

```
annoying <- annoying %>% add_column(`3`=annoying$`2`/annoying$`1`)
annoying
```

```
## # A tibble: 10 x 3
##      `1`   `2`   `3`
##    <int> <dbl> <dbl>
##  1     1  1.82  1.82
##  2     2  4.78  2.39
##  3     3  6.28  2.09
##  4     4  7.13  1.78
##  5     5  9.05  1.81
##  6     6  9.97  1.66
##  7     7 16.6   2.37
##  8     8 16.4   2.05
##  9     9 18.1   2.01
## 10    10 20.9   2.09
```

4. Renaming the columns to `one`, `two` and `three`.

```
annoying <- annoying %>% rename('one'=`1`,'two'=`2`,'three'=`3`)
annoying
```

```
## # A tibble: 10 x 3
##      one   two three
##    <int> <dbl> <dbl>
##  1     1  1.82  1.82
##  2     2  4.78  2.39
##  3     3  6.28  2.09
##  4     4  7.13  1.78
##  5     5  9.05  1.81
##  6     6  9.97  1.66
##  7     7 16.6   2.37
##  8     8 16.4   2.05
##  9     9 18.1   2.01
## 10    10 20.9   2.09
```

5. What does `tibble::enframe()` do? When might you use it?

`enframe()` takes a vector or a list and then convert it into a tibble. I can use the command if I have a vector or a list and turn it into a tibble. It would give me a tibble with a column "name" and another column "value", which stores values in my vector or list.

```
# two examples
enframe(1:3)
```

```
## # A tibble: 3 x 2
##    name value
##   <int> <int>
## 1     1     1
## 2     2     2
## 3     3     3
```

```
item <- c(a=3,b=5)
enframe(item)
```

```
## # A tibble: 2 x 2
##   name  value
##   <chr> <dbl>
## 1 a      3.00
## 2 b      5.00
```

6. What option controls how many additional column names are printed at the footer of a tibble?

For example, if we want to print additional 5 columns , we can use `options(tibble.max_extra_cols = 5)`. If we want to show all columns, then we can use `options(tibble.width = Inf)`