# MA 415: Assignment 4

*Kayla Ippongi*

### Section 10.5 Exercises

1. How can you tell if an object is a tibble? (Hint: try printing mtcars, which is a regular data frame).

```
x = c(1,2,3,4)
print(is.tibble(x)) #will return false because x is not a tibble
```

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

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?
   - data frames seem to return values in levels
   - while a tibble returns the actual x value

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

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

```
tib$x
```

```
## [1] "abc"
```

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

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

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

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

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?

```
var <- "mpg"
tib <- tibble(var, y = "a")
select(tib,var)
```

```
## # A tibble: 1 x 1
##   var
##   <chr>
## 1 mpg
```

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

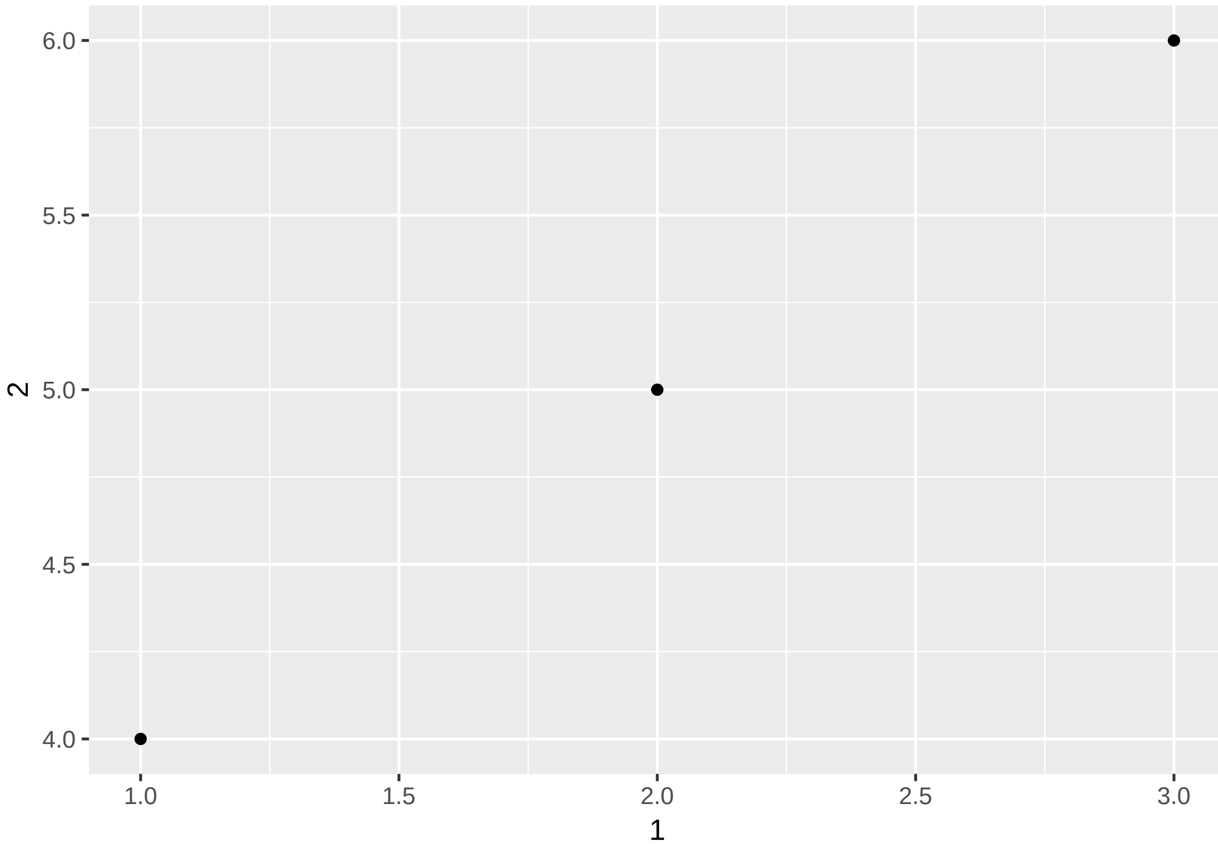
a) Extracting the variable called 1.

```
tib <- tibble('1' = "abc", '2' = "test")
tib$'1'
```

```
## [1] "abc"
```

b) Plotting a scatterplot of 1 vs 2.

```
tib <- tibble('1' = 1:3, '2' = 4:6)
ggplot(tib, aes(x = `1`, y = `2`)) + geom_point()
```




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

```
tib <-
  tib %>%
  mutate(`3` = `2`/`1`)
```

d) Renaming the columns to one, two and three.

```
tib %>%
  rename(one = `1`,
         two = `2`,
         three = `3`)
```

```
## # A tibble: 3 x 3
##     one   two three
##   <int> <int> <dbl>
## 1     1     4  4.00
## 2     2     5  2.50
## 3     3     6  2.00
```

5. What does tibble::enframe() do? When might you use it? enframe() takes a vector or list and converts it to a dataframe

```
x <- c(1:5)
enframe(x)
```

```
## # A tibble: 5 x 2
##    name value
##   <int> <int>
## 1     1     1
## 2     2     2
## 3     3     3
## 4     4     4
## 5     5     5
```

```
c <- enframe(list(a = 2, b = 3))
```

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

- tibble.max_extra_cols controls how many additional columns are printed at the end

### 12.6.1 Exercises

```
library(tidyverse)
who1 <- who %>%
  gather(new_sp_m014:newrel_f65, key = "key", value = "cases", na.rm = TRUE)

who2 <- who1 %>%
  mutate(key = stringr::str_replace(key, "newrel", "new_rel"))

who3 <- who2 %>%
  separate(key, c("new", "type", "sexage"), sep = "_")

who4 <- who3 %>%
  select(-new, -iso2, -iso3)

who5 <- who4 %>%
  separate(sexage, c("sex", "age"), sep = 1)

who %>%
  gather(code, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
  mutate(code = stringr::str_replace(code, "newrel", "new_rel")) %>%
  separate(code, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

```
## # A tibble: 76,046 x 6
##    country      year var   sex   age   value
##    <chr>       <int> <chr> <chr> <chr> <int>
##  1 Afghanistan  1997 sp    m     014       0
##  2 Afghanistan  1998 sp    m     014      30
##  3 Afghanistan  1999 sp    m     014       8
##  4 Afghanistan  2000 sp    m     014      52
##  5 Afghanistan  2001 sp    m     014     129
##  6 Afghanistan  2002 sp    m     014      90
##  7 Afghanistan  2003 sp    m     014     127
##  8 Afghanistan  2004 sp    m     014     139
##  9 Afghanistan  2005 sp    m     014     151
## 10 Afghanistan  2006 sp    m     014     193
## # ... with 76,036 more rows
```

1. In this case study I set na.rm = TRUE just to make it easier to check that we had the correct values. Is this reasonable? Think about how missing values are represented in this dataset. Are there implicit missing values? What's the difference between an NA and zero? Sure, it's reasonable. Setting na.rm to True allows those 'missing' values to be represented as not applicable, so users can see what values are actually missing. Therefore, the difference between NA and Zero is NA would imply a missing value, while zero is the actual value

2. What happens if you neglect the mutate() step? (mutate(key = stringr::str_replace(key, "newrel", "new_rel"))) If the mutate step is skipped, the missing data set values are not properly replaced and you're given an error message similar to before. To avoid that error the mutate() step seems to be replacing those strings

```
who %>%
  gather(code, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
  #mutate(code = stringr::str_replace(code, "newrel", "new_rel")) %>%
  separate(code, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 2580 rows
## [73467, 73468, 73469, 73470, 73471, 73472, 73473, 73474, 73475, 73476,
## 73477, 73478, 73479, 73480, 73481, 73482, 73483, 73484, 73485, 73486, ...].
```

```
## # A tibble: 76,046 x 6
##    country      year var   sex   age   value
##  * <chr>       <int> <chr> <chr> <chr> <int>
##  1 Afghanistan  1997 sp    m     014       0
##  2 Afghanistan  1998 sp    m     014      30
##  3 Afghanistan  1999 sp    m     014       8
##  4 Afghanistan  2000 sp    m     014      52
##  5 Afghanistan  2001 sp    m     014     129
##  6 Afghanistan  2002 sp    m     014      90
##  7 Afghanistan  2003 sp    m     014     127
##  8 Afghanistan  2004 sp    m     014     139
##  9 Afghanistan  2005 sp    m     014     151
## 10 Afghanistan  2006 sp    m     014     193
## # ... with 76,036 more rows
```

3. I claimed that iso2 and iso3 were redundant with country. Confirm this claim.

```
select(who3, country, iso2, iso3) %>%
  distinct() %>%
  group_by(country) %>%
  filter(n() > 1)
```

```
## # A tibble: 0 x 3
## # Groups:   country [0]
## # ... with 3 variables: country <chr>, iso2 <chr>, iso3 <chr>
```

4. For each country, year, and sex compute the total number of cases of TB. Make an informative visualisation of the data. We can use the data from who5, which holds the sex and age variables to grab those variables and group by country,year and sex

```
who5 %>%
  group_by(country, year, sex) %>%
  filter(year > 2000) %>%
  summarise(cases = sum(cases)) %>%
  unite(country_sex, country, sex, remove=FALSE) %>%
```

```r
ggplot(aes(x = year, y = cases, group = country_sex, colour = sex)) +
geom_line()
```