Assignment4

Junpei Xiao

2018-2-20

10.5 Exercise

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

```
tb <- as_tibble(mtcars)
print(tb)
```

```
## # 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
```

```
##tibble never changes the type of the inputs such as it never change strings to factors!), and it neve

### tibble
```

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?

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

df$x
```

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

```
class(df[,"xyz"])   ### return a factor
```

```
## [1] "factor"
```

```
class(df[, c("abc", "xyz")]) ### return a data.frame
```

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

```
tb_df <- as_tibble(df)
### tb_df$x             ### warning Unknown or uninitialised column: 'x'.NULL
```

```
class(tb_df[, "xyz"])   ### return a tibber
```

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

```
class(tb_df[, c("abc", "xyz")]) ### return a tibber
```

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

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?

```
tb_cars <- as_tibble(mtcars)
var <- "mpg"
tb_cars[var]
```

```
## # A tibble: 32 x 1
##      mpg
##    <dbl>
##  1  21.0
##  2  21.0
##  3  22.8
##  4  21.4
##  5  18.7
##  6  18.1
##  7  14.3
##  8  24.4
##  9  22.8
## 10  19.2
## # ... with 22 more rows
```

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

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

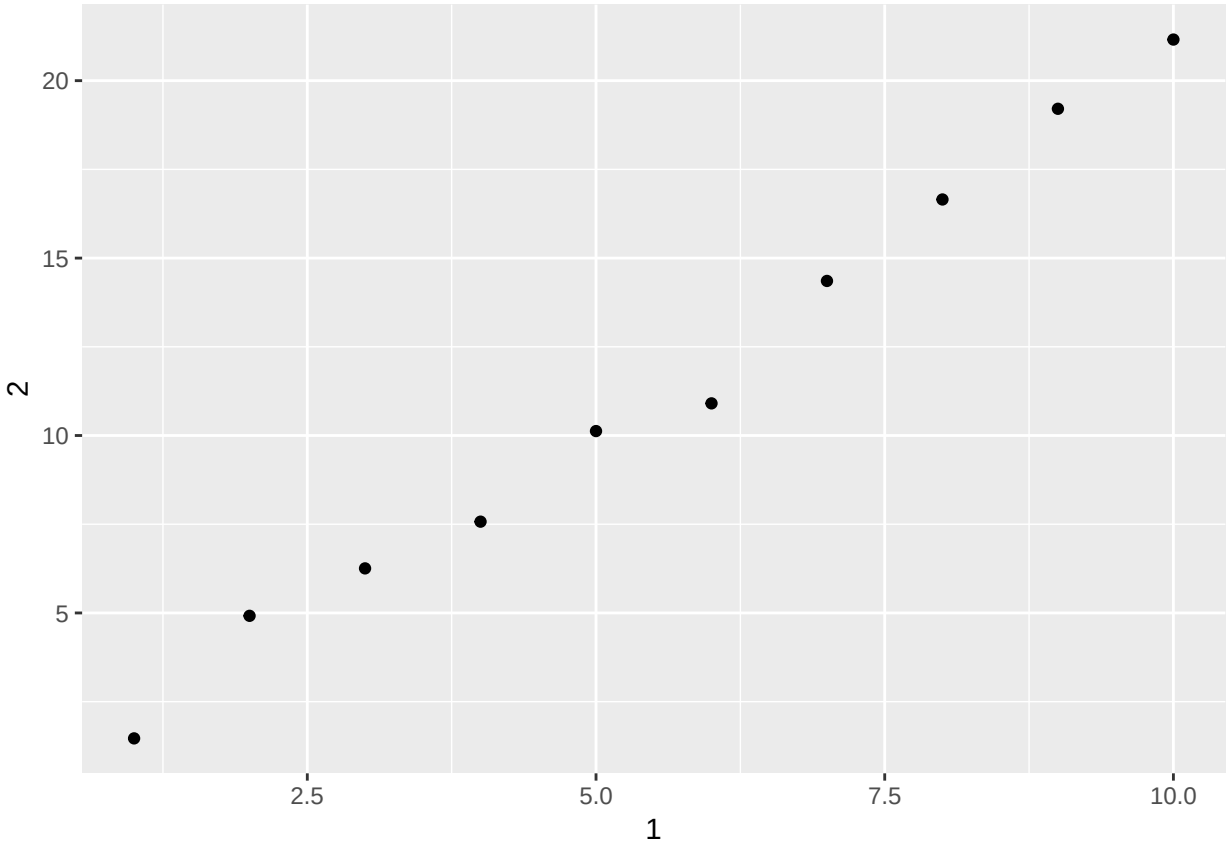
1. Extracting the variable called 1.

```
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 %>%mutate(`3` = `2`/`1`)
```

4 Renaming the columns to one, two and three.

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

```
## # A tibble: 10 x 3
##      one   two three
##    <int> <dbl> <dbl>
##  1     1  1.47  1.47
##  2     2  4.92  2.46
##  3     3  6.26  2.09
##  4     4  7.57  1.89
##  5     5 10.1   2.03
##  6     6 10.9   1.82
##  7     7 14.4   2.05
##  8     8 16.7   2.08
##  9     9 19.2   2.13
## 10    10 21.2   2.12
```

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

```
sample <- letters[1:10]
enframe(sample)
```

```
## # A tibble: 10 x 2
##     name value
##    <int> <chr>
##  1     1 a
##  2     2 b
##  3     3 c
##  4     4 d
##  5     5 e
##  6     6 f
##  7     7 g
##  8     8 h
##  9     9 i
## 10    10 j
```

```
### convert vectors to data frames, and vice versa.
```

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

```
### tibble.max_extra_cols
### Number of extra columns printed in reduced form. Default: 100.
```

12.6.1 Exercises

```
### This part code copied from 12.6 Case study, becasue it is needed by exercise

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

```
## Observations: 76,046
## Variables: 6
## $ country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanis...
## $ iso2    <chr> "AF", "AF", "AF", "AF", "AF", "AF", "AF", "AF", "AF", ...
## $ iso3    <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG"...
## $ year    <int> 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, ...
## $ key     <chr> "new_sp_m014", "new_sp_m014", "new_sp_m014", "new_sp_m...
## $ cases   <int> 0, 30, 8, 52, 129, 90, 127, 139, 151, 193, 186, 187, 2...
```

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

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

```
## # A tibble: 76,046 x 8
##    country     iso2  iso3   year new   type  sexage cases
```

```
##    <chr>       <chr> <chr> <int> <chr> <chr> <chr>  <int>
##  1 Afghanistan AF    AFG    1997 new   sp    m014       0
##  2 Afghanistan AF    AFG    1998 new   sp    m014      30
##  3 Afghanistan AF    AFG    1999 new   sp    m014       8
##  4 Afghanistan AF    AFG    2000 new   sp    m014      52
##  5 Afghanistan AF    AFG    2001 new   sp    m014     129
##  6 Afghanistan AF    AFG    2002 new   sp    m014      90
##  7 Afghanistan AF    AFG    2003 new   sp    m014     127
##  8 Afghanistan AF    AFG    2004 new   sp    m014     139
##  9 Afghanistan AF    AFG    2005 new   sp    m014     151
## 10 Afghanistan AF    AFG    2006 new   sp    m014     193
## # ... with 76,036 more rows
```

```
who3 %>%
count(new)
```

```
## # A tibble: 1 x 2
##   new       n
##   <chr> <int>
## 1 new   76046
```

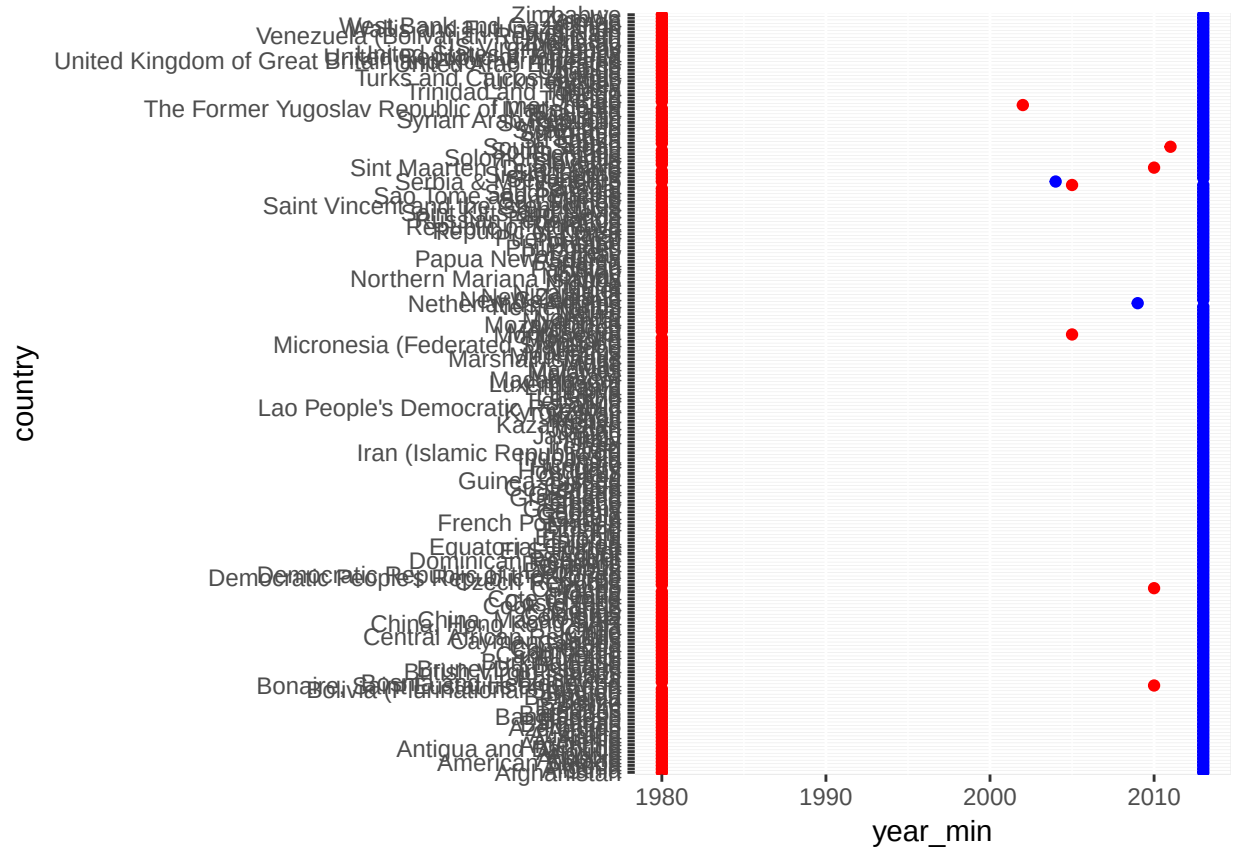
```
who4 <- who3 %>%
select(-new, -iso2, -iso3)

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

```
## # A tibble: 76,046 x 6
##    country      year type  sex   age   cases
##    <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?

```
who %>%
group_by(country) %>%
summarize(year_min = min(year), year_max = max(year)) %>%
ggplot() +
geom_point(mapping = aes(x = country, y = year_min), color = 'red') +
geom_point(mapping = aes(x = country , y= year_max), color = 'blue') +
coord_flip()
```




```
### From the plot we can see there are some implicit missing values

### Cheack zero and NA.
sum(who %>% select(-c(1:4)) == 0, na.rm = TRUE)
```

```
## [1] 11080
```

```
who %>% select(-c(1:4)) %>%sapply(function(x){sum(is.na(x))})
```

```
##  new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544 new_sp_m4554
##         4067         4031         4034         4021         4017
## new_sp_m5564   new_sp_m65  new_sp_f014 new_sp_f1524 new_sp_f2534
##         4022         4031         4066         4046         4040
## new_sp_f3544 new_sp_f4554 new_sp_f5564   new_sp_f65  new_sn_m014
##         4041         4036         4045         4043         6195
## new_sn_m1524 new_sn_m2534 new_sn_m3544 new_sn_m4554 new_sn_m5564
##         6210         6218         6215         6213         6219
##   new_sn_m65  new_sn_f014 new_sn_f1524 new_sn_f2534 new_sn_f3544
##         6220         6200         6218         6224         6220
## new_sn_f4554 new_sn_f5564   new_sn_f65  new_ep_m014 new_ep_m1524
##         6222         6223         6221         6202         6214
## new_ep_m2534 new_ep_m3544 new_ep_m4554 new_ep_m5564   new_ep_m65
##         6220         6216         6220         6225         6222
##  new_ep_f014 new_ep_f1524 new_ep_f2534 new_ep_f3544 new_ep_f4554
##         6208         6219         6219         6219         6223
## new_ep_f5564   new_ep_f65  newrel_m014 newrel_m1524 newrel_m2534
##         6223         6226         7050         7058         7057
```

```
## newrel_m3544 newrel_m4554 newrel_m5564   newrel_m65  newrel_f014
##         7056         7056         7055         7058         7050
## newrel_f1524 newrel_f2534 newrel_f3544 newrel_f4554 newrel_f5564
##         7056         7058         7057         7057         7057
##   newrel_f65
##         7055
```

```
### Zero simply means no case, and NA means missing values
```

2. What happens if you neglect the mutate() step? (mutate(key = stringr::str_replace(key, "newrel", "new_rel")))

```
### The code will not be separated correctly into the three columns sexage,new,var.
```

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

```
### Check unique values in country,iso2,iso3
who %>% select(1:3) %>% sapply(function(x){length(unique(x))})
```

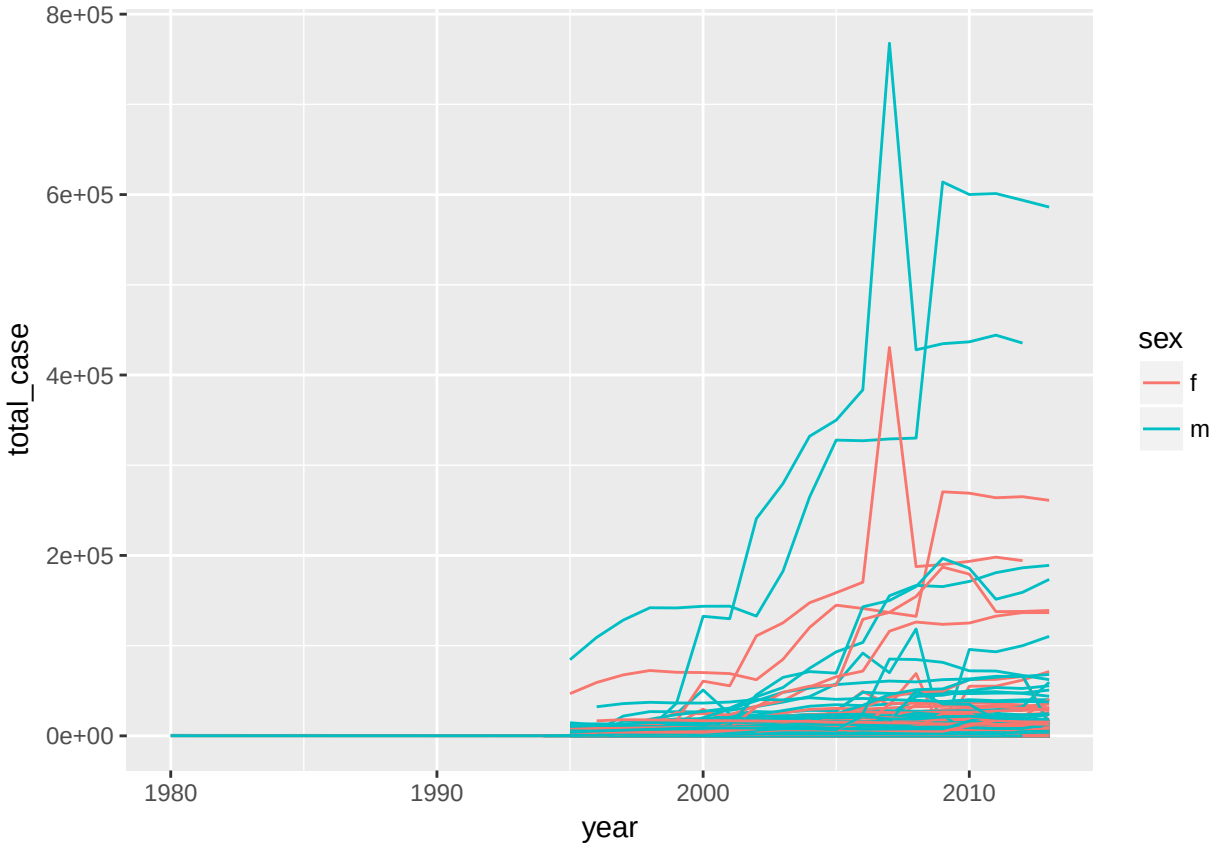
```
## country    iso2    iso3
##     219     219     219
```

```
### check the unique combination of these three columns
who %>% select(1:3) %>%
  unite(combined, 1:3) %>%
  select(combined) %>%
  distinct() %>%
  nrow()
```

```
## [1] 219
```

4.For each country, year, and sex compute the total number of cases of TB. Make an informative visualisation of the data

```
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) %>%
  group_by(country, year, sex) %>%
  summarize(total_case = sum(value)) %>%
  unite(country_sex, country, sex, remove = FALSE) %>%
  ggplot() +
  geom_line(mapping = aes(x = year, y = total_case, color = sex,
                          group = country_sex))
```


8e+05
6e+05
4e+05
2e+05
0e+00
total_case
1980
1990
2000
2010
year
sex
f
m