Assignment 4 Textbook Exercises

Doug Goon

April 27, 2018

10.5

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

```
is.tibble(mtcars)
```

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

2. Compare and contrast the following operations on a data.frame and equivalent tibble. What is different? Tibble doesn't allow for partial matching when subsetting.

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

```
#partial matching
df$x
```

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

```
#doesn't do partial matching
df_tibble$x
```

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

```
## NULL
```

```
#becomes factor
df[, "xyz"]
```

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

```
#returns tibble
df_tibble [,"xyz"]
```

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

```
#returns as factor type
df[, c("abc", "xyz")]
```

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

```
#return as character type
df_tibble[, c("abc", "xyz")]
```

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

Why might the default data frame behaviours cause you frustration? It might cause frustration because the functions might not work.

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 <- "abc"
df_tibble[[var]]
```

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

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

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

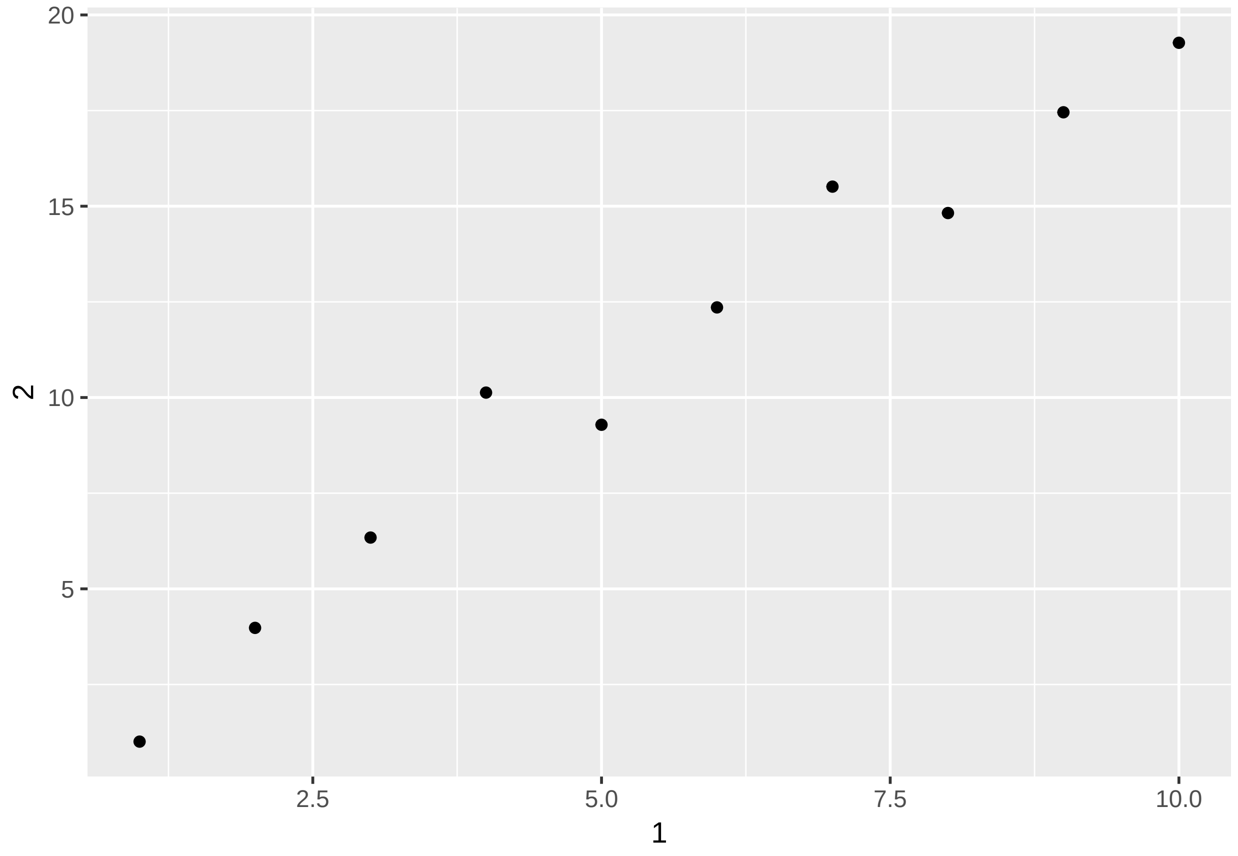
Extracting the variable called 1.

```
annoying$`1`
```

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

Plotting a scatterplot of 1 vs 2.

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



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

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

Renaming the columns to one, two and three.

```
dplyr::rename(annoying, One =`1`, Two = `2`, Three = `3`)
```

```
## # A tibble: 10 x 3
##      One       Two    Three
##    <int>     <dbl>    <dbl>
##  1     1  1.006414 1.006414
##  2     2  3.979418 1.989709
##  3     3  6.339763 2.113254
##  4     4 10.128598 2.532149
##  5     5  9.288020 1.857604
##  6     6 12.355255 2.059209
##  7     7 15.512519 2.216074
##  8     8 14.822050 1.852756
##  9     9 17.455482 1.939498
## 10    10 19.271924 1.927192
```

5. What does tibble::enframe() do?

tibble::enframe() converts named atomic vectors or lists to two-column data frames

When might you use it?

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

```
## # 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
```

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

```
?print.tbl_df
```

```
## starting httpd help server ... done
```

```
print(as_tibble(mtcars), n = 3)
```

```
## # 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   160   110  3.90 2.620 16.46     0     1     4     4
## 2  21.0     6   160   110  3.90 2.875 17.02     0     1     4     4
## 3  22.8     4   108    93  3.85 2.320 18.61     1     1     4     1
## # ... with 29 more rows
```

12.6.1

```r
tidyr::who
```

```
## # A tibble: 7,240 x 60
##        country  iso2  iso3  year new_sp_m014 new_sp_m1524 new_sp_m2534
##          <chr> <chr> <chr> <int>       <int>        <int>        <int>
##  1 Afghanistan    AF   AFG  1980          NA           NA           NA
##  2 Afghanistan    AF   AFG  1981          NA           NA           NA
##  3 Afghanistan    AF   AFG  1982          NA           NA           NA
##  4 Afghanistan    AF   AFG  1983          NA           NA           NA
##  5 Afghanistan    AF   AFG  1984          NA           NA           NA
##  6 Afghanistan    AF   AFG  1985          NA           NA           NA
##  7 Afghanistan    AF   AFG  1986          NA           NA           NA
##  8 Afghanistan    AF   AFG  1987          NA           NA           NA
##  9 Afghanistan    AF   AFG  1988          NA           NA           NA
## 10 Afghanistan    AF   AFG  1989          NA           NA           NA
## # ... with 7,230 more rows, and 53 more variables: new_sp_m3544 <int>,
## #   new_sp_m4554 <int>, new_sp_m5564 <int>, new_sp_m65 <int>,
## #   new_sp_f014 <int>, new_sp_f1524 <int>, new_sp_f2534 <int>,
## #   new_sp_f3544 <int>, new_sp_f4554 <int>, new_sp_f5564 <int>,
## #   new_sp_f65 <int>, new_sn_m014 <int>, new_sn_m1524 <int>,
## #   new_sn_m2534 <int>, new_sn_m3544 <int>, new_sn_m4554 <int>,
## #   new_sn_m5564 <int>, new_sn_m65 <int>, new_sn_f014 <int>,
## #   new_sn_f1524 <int>, new_sn_f2534 <int>, new_sn_f3544 <int>,
## #   new_sn_f4554 <int>, new_sn_f5564 <int>, new_sn_f65 <int>,
## #   new_ep_m014 <int>, new_ep_m1524 <int>, new_ep_m2534 <int>,
## #   new_ep_m3544 <int>, new_ep_m4554 <int>, new_ep_m5564 <int>,
## #   new_ep_m65 <int>, new_ep_f014 <int>, new_ep_f1524 <int>,
## #   new_ep_f2534 <int>, new_ep_f3544 <int>, new_ep_f4554 <int>,
## #   new_ep_f5564 <int>, new_ep_f65 <int>, newrel_m014 <int>,
## #   newrel_m1524 <int>, newrel_m2534 <int>, newrel_m3544 <int>,
## #   newrel_m4554 <int>, newrel_m5564 <int>, newrel_m65 <int>,
## #   newrel_f014 <int>, newrel_f1524 <int>, newrel_f2534 <int>,
## #   newrel_f3544 <int>, newrel_f4554 <int>, newrel_f5564 <int>,
## #   newrel_f65 <int>
```

```r
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
```

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

```
who %>%
  select(1:3) %>%
  sapply(function(x){length(unique(x))})
```

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

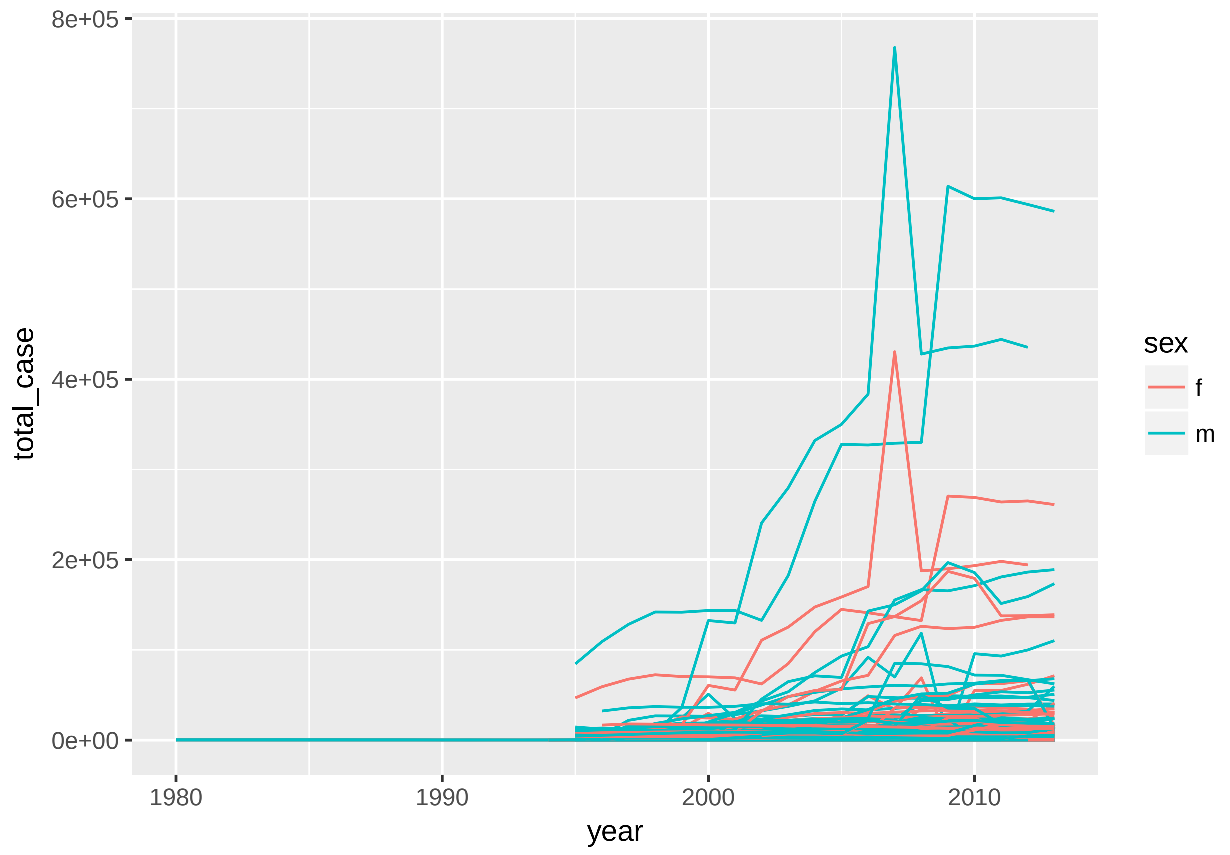
```
who %>% select(1:3) %>%
  unite(combined, 1:3) %>%
  select(combined) %>%
  distinct() %>%
  nrow()
```

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

Claim confirmed.

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