Assignment 4, Part 1

Brianna Kincaid

February 20, 2018

10.5 Exercises

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

```
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
```

```
class(mtcars)
```

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

```
class(as_tibble(mtcars))
```

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

Tibbles show the class of each column and also do not show all the rows.They also have the tbl_df and tbl_.

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 frame:

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

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

This can accidentally return the wrong result. It gives df$xyz.

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

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

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

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

Tibble:

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

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

```
## NULL
```

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

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

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

```
## # A tibble: 1 x 2
##     abc xyz
##   <dbl> <fct>
## 1  1.00 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?

You use the double bracket.

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

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

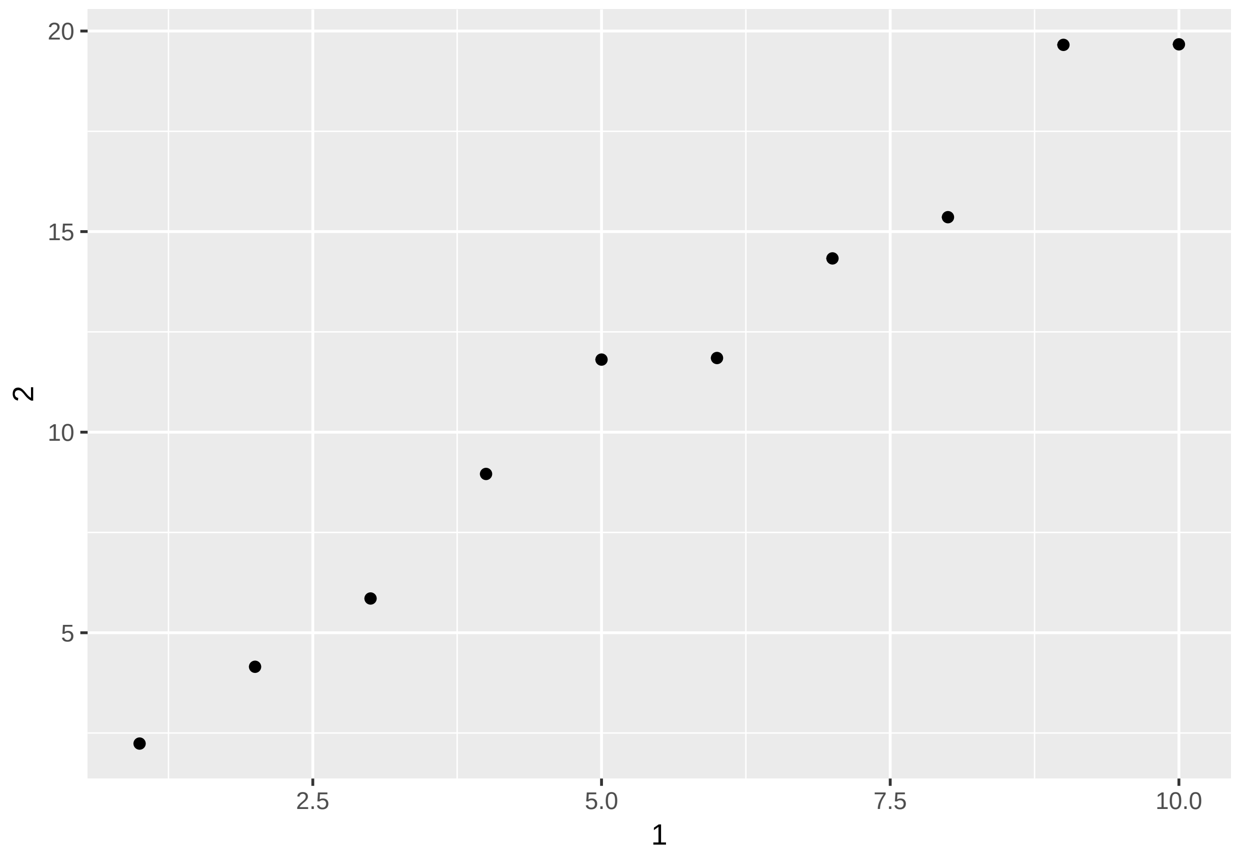
(a) Extracting the variable called 1.

```
annoying[["1"]]
```

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

(b) Plotting a scatterplot of 1 vs 2.

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



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

```r
annoying[["3"]] <- annoying[["2"]] / annoying[["1"]]
```

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

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

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

It converts named vectors or lists to two-column data frames.

```r
enframe(c(a = 5, b = 7))
```

```
## # A tibble: 2 x 2
##   name  value
##   <chr> <dbl>
## 1 a      5.00
## 2 b      7.00
```

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

The n_extra option in the print function, print.tbl_df, determines the number of additional column names printed at the footer of the tibble.

12.6.1 Exercises

The tidyr::who dataset contains tuberculosis (TB) cases broken down by year, country, age, gender, and diagnosis method.

```
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= "_")

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)
```

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?

Removing the missing values (NA) is reasonable because we can reasonably treat explicitly missing values the same as implicitly missing values. Zero's explicitly indicate no cases of TB, while NA represents missing data.

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

separate emits the warning "too few values", and if we check the rows for keys beginning with "newrel_", we see that sexage is messing, and type = m014.

```
who3a <- who1 %>%
  separate(key, c("new", "type", "sexage"), sep = "_")
```

```
## 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, ...].
```

```
filter(who3a, new == "newrel") %>% head()
```

```
## # A tibble: 6 x 8
##   country     iso2  iso3   year new    type  sexage cases
##   <chr>       <chr> <chr> <int> <chr>  <chr> <chr>  <int>
## 1 Afghanistan AF    AFG    2013 newrel m014  <NA>    1705
## 2 Albania     AL    ALB    2013 newrel m014  <NA>      14
## 3 Algeria     DZ    DZA    2013 newrel m014  <NA>      25
## 4 Andorra     AD    AND    2013 newrel m014  <NA>       0
## 5 Angola      AO    AGO    2013 newrel m014  <NA>     486
## 6 Anguilla    AI    AIA    2013 newrel m014  <NA>       0
```

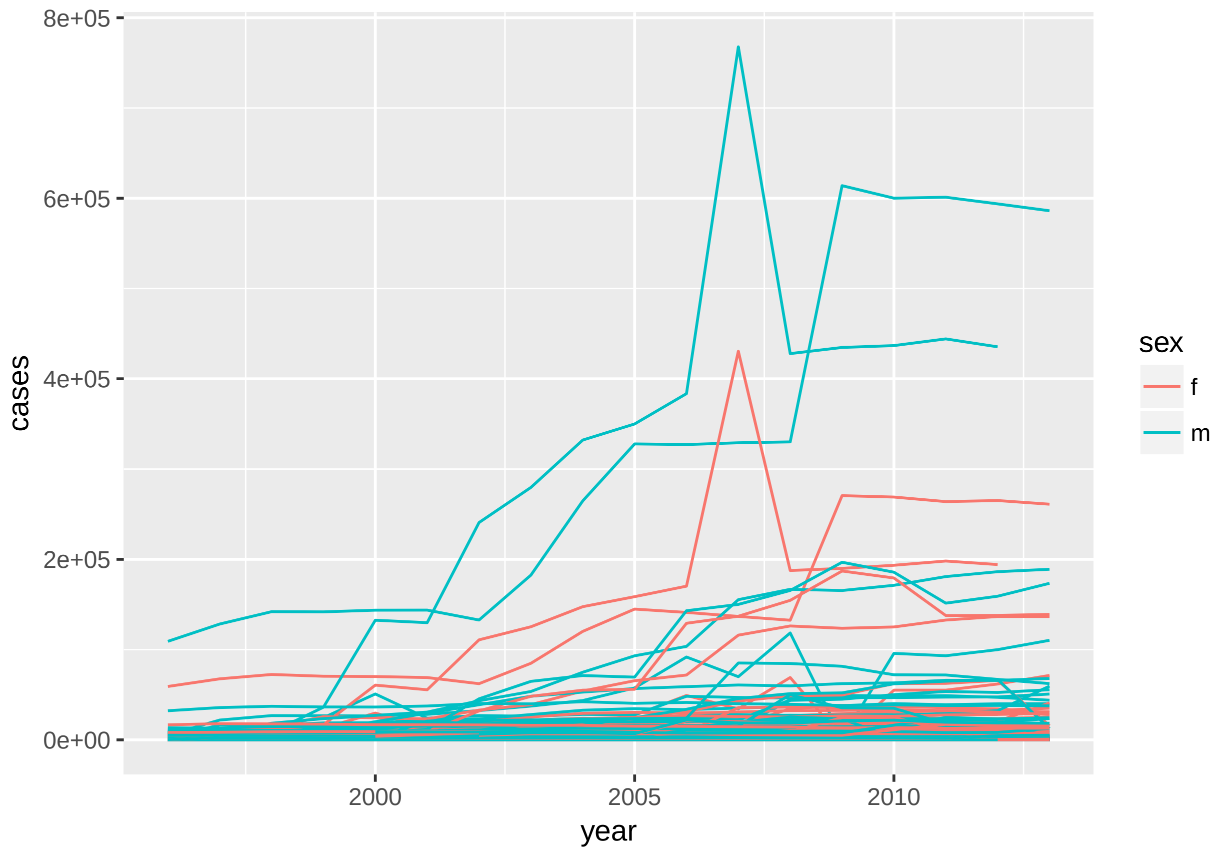
3. I claimed that iso2 and iso3 were redundant with country. Confirm this claim.

```
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.

```
who5 %>%
  group_by(country, year, sex) %>%
  filter(year > 1995) %>%
  summarise(cases = sum(cases)) %>%
  unite(country_sex, country, sex, remove = FALSE) %>%
  ggplot(aes(x = year, y = cases, group = country_sex, colour = sex)) +
  geom_line()
```


8e+05
6e+05
4e+05
2e+05
0e+00
cases
year
sex
f
m