Homework 4

Yue Wu

2/20/2018

10.5 Exercises

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

A tibble will print with the comment "A tible: n x m" when called. In addition, it automatically display variable types. A R-built-in data.frame shows none of the above when called.

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
```

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

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

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

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

tibble:

```
dft <- tibble(abc = 1, xyz = "a")
dft$x
```

```
## NULL
```

```
print(dft[, "xyz"])
```

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

```
1
```

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

```
print(dft[, c("abc", "xyz")])
```

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

Using data.frame, the outputs consist of both data value and "levels", which repeat each other. Using tibble, the outputs are in neat table format, without redundant information.

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?

Double [[]] extracts data from the reference.

```
var <- "mpg"
dft[[var]]
```

```
## NULL
```

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

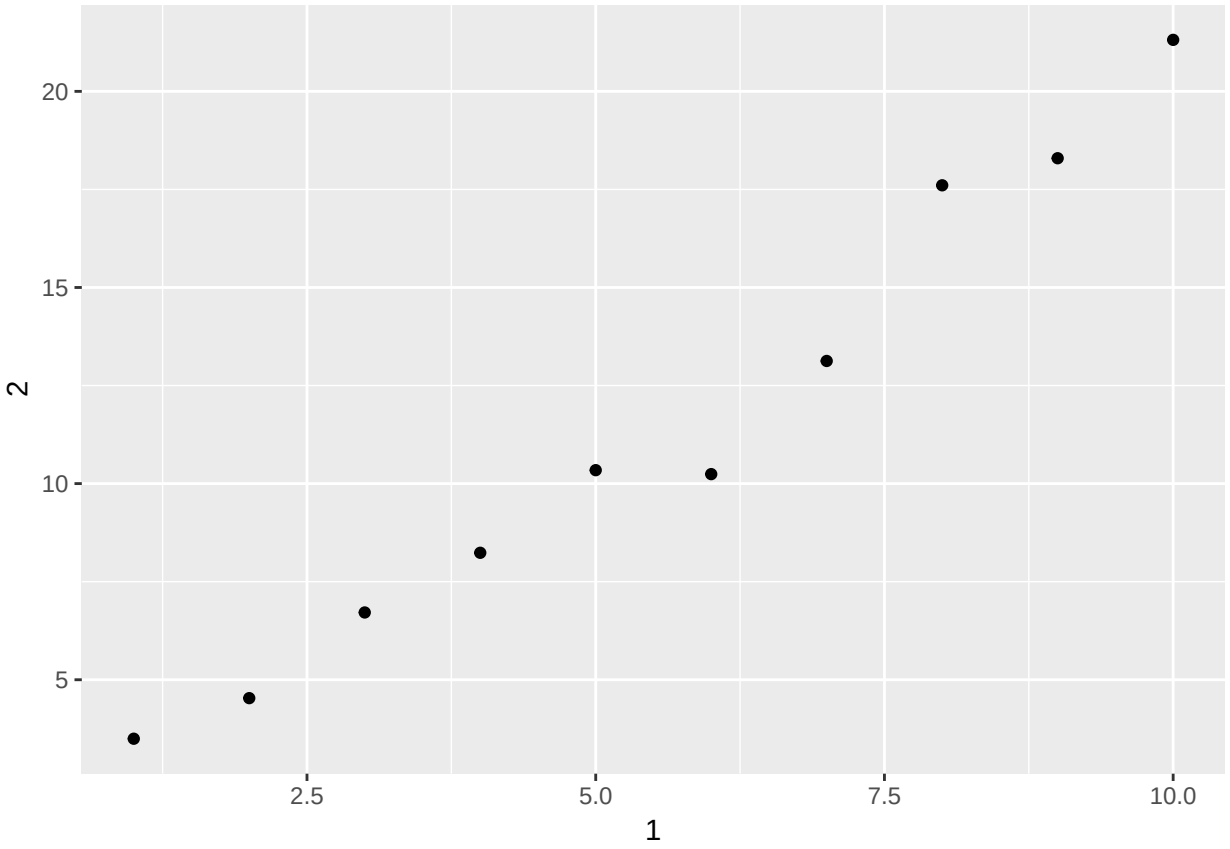
Extracting the variable called 1.

```
annoying <- tibble(
  `1` = 1:10,
  `2` = `1` * 2 + rnorm(length(`1`))
)
annoying %>% .$'1'
```

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

Plotting a scatterplot of 1 vs 2.

```
ggplot(annoying) +
  geom_point(aes(`1`,`2`))
```



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

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

```
## # A tibble: 10 x 3
##      `1`   `2`   `3`
##    <int> <dbl> <dbl>
##  1     1  3.50  3.50
##  2     2  4.53  2.27
##  3     3  6.72  2.24
##  4     4  8.24  2.06
##  5     5 10.3   2.07
##  6     6 10.2   1.71
##  7     7 13.1   1.88
##  8     8 17.6   2.20
##  9     9 18.3   2.03
## 10    10 21.3   2.13
```

Renaming the columns to one, two and three.

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

```
## # A tibble: 10 x 3
##      one   two three
##    <int> <dbl> <dbl>
##  1     1  3.50  3.50
##  2     2  4.53  2.27
##  3     3  6.72  2.24
##  4     4  8.24  2.06
##  5     5 10.3   2.07
##  6     6 10.2   1.71
##  7     7 13.1   1.88
##  8     8 17.6   2.20
##  9     9 18.3   2.03
## 10    10 21.3   2.13
```

```
annoying
```

```
## # A tibble: 10 x 3
##      `1`   `2`   `3`
##    <int> <dbl> <dbl>
##  1     1  3.50  3.50
##  2     2  4.53  2.27
##  3     3  6.72  2.24
##  4     4  8.24  2.06
##  5     5 10.3   2.07
##  6     6 10.2   1.71
##  7     7 13.1   1.88
##  8     8 17.6   2.20
##  9     9 18.3   2.03
## 10    10 21.3   2.13
```

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

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

```
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?

print(x,...n_extra = n) The example prints a tibble table with 2 additional column names in the footnote:

```
ex106 <- tibble(
  a= c(1:10), b= c(11:20), c= c(21:30), d= c(31:40), e= c(41:50), f= c(51:60), g= c(61:70), h= c(71:80)
  a1= c(1:10), b1= c(11:20), c1= c(21:30), d1= c(31:40), e1= c(41:50), f1= c(51:60), g1= c(61:70), h1= c
  a2= c(1:10), b2= c(11:20), c2= c(21:30), d2= c(31:40), e2= c(41:50), f2= c(51:60), g2= c(61:70), h2= c
)
print(ex106, n_extra =2)
```

```
## # A tibble: 10 x 27
##        a     b     c     d     e     f     g     h     i    a1    b1    c1
##    <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
##  1     1    11    21    31    41    51    61    71    81     1    11    21
##  2     2    12    22    32    42    52    62    72    82     2    12    22
##  3     3    13    23    33    43    53    63    73    83     3    13    23
##  4     4    14    24    34    44    54    64    74    84     4    14    24
##  5     5    15    25    35    45    55    65    75    85     5    15    25
##  6     6    16    26    36    46    56    66    76    86     6    16    26
##  7     7    17    27    37    47    57    67    77    87     7    17    27
##  8     8    18    28    38    48    58    68    78    88     8    18    28
##  9     9    19    29    39    49    59    69    79    89     9    19    29
## 10    10    20    30    40    50    60    70    80    90    10    20    30
## # ... with 15 more variables: d1 <int>, e1 <int>, ...
```

12.6.1 Exercises

Repeat the case study

```
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?

There is difference between NA and zero in this case. Country and year with 0 cases are not listed as NA nor vise versa. Zero means no cases instead of missing entry. It does not change any information to omit the NA ones here.

```
who1 %>%
  filter(cases == 0)
```

```
## # A tibble: 11,080 x 6
##    country     iso2  iso3   year key         cases
##    <chr>       <chr> <chr> <int> <chr>       <int>
##  1 Afghanistan AF    AFG    1997 new_sp_m014     0
##  2 Albania     AL    ALB    1995 new_sp_m014     0
##  3 Albania     AL    ALB    1997 new_sp_m014     0
##  4 Albania     AL    ALB    1999 new_sp_m014     0
##  5 Albania     AL    ALB    2002 new_sp_m014     0
##  6 Albania     AL    ALB    2003 new_sp_m014     0
##  7 Albania     AL    ALB    2005 new_sp_m014     0
##  8 Albania     AL    ALB    2007 new_sp_m014     0
##  9 Albania     AL    ALB    2009 new_sp_m014     0
## 10 Albania     AL    ALB    2010 new_sp_m014     0
## # ... with 11,070 more rows
```

```
whoex1 <-  who %>%
  gather(new_sp_m014:newrel_f65, key = "key", value = "cases", na.rm = F) %>%
  filter(is.na(cases))
whoex1
```

```
## # A tibble: 329,394 x 6
##    country     iso2  iso3   year key         cases
##    <chr>       <chr> <chr> <int> <chr>       <int>
##  1 Afghanistan AF    AFG    1980 new_sp_m014    NA
##  2 Afghanistan AF    AFG    1981 new_sp_m014    NA
##  3 Afghanistan AF    AFG    1982 new_sp_m014    NA
```

```
##  4 Afghanistan AF    AFG    1983 new_sp_m014    NA
##  5 Afghanistan AF    AFG    1984 new_sp_m014    NA
##  6 Afghanistan AF    AFG    1985 new_sp_m014    NA
##  7 Afghanistan AF    AFG    1986 new_sp_m014    NA
##  8 Afghanistan AF    AFG    1987 new_sp_m014    NA
##  9 Afghanistan AF    AFG    1988 new_sp_m014    NA
## 10 Afghanistan AF    AFG    1989 new_sp_m014    NA
## # ... with 329,384 more rows
```

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

```
whoex2 <- who %>%
          gather(code, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
          separate(code, c("new", "type", "sexage"))
filter(whoex2, new == "newrel")
```

```
## # A tibble: 2,580 x 8
##    country             iso2  iso3   year new    type  sexage value
##    <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
##  7 Antigua and Barbuda AG    ATG    2013 newrel m014  <NA>       1
##  8 Argentina           AR    ARG    2013 newrel m014  <NA>     462
##  9 Armenia             AM    ARM    2013 newrel m014  <NA>      25
## 10 Australia           AU    AUS    2013 newrel m014  <NA>      28
## # ... with 2,570 more rows
```

There is warning for missing pieces, and all sexage filled with NA. For all 'newrel' observations, type = m014.

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

None of the countries have multiple iso2 or iso3 codes.

```
whoex3 <- select(who3, country, iso2, iso3) %>%
           group_by(country)
n_groups(whoex3)
```

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

```
whoex3_2 <-  whoex3 %>% group_by(country,iso2,iso3)
n_groups(whoex3_2)
```

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

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

```
whoex4 <- who5 %>%
  group_by(country, year, sex) %>%
  summarise(cases = sum(cases))
whoex4
```

```
## # A tibble: 6,921 x 4
## # Groups:   country, year [?]
##    country      year sex   cases
##    <chr>       <int> <chr> <int>
##  1 Afghanistan  1997 f       102
##  2 Afghanistan  1997 m        26
##  3 Afghanistan  1998 f      1207
##  4 Afghanistan  1998 m       571
##  5 Afghanistan  1999 f       517
##  6 Afghanistan  1999 m       228
##  7 Afghanistan  2000 f      1751
##  8 Afghanistan  2000 m       915
##  9 Afghanistan  2001 f      3062
## 10 Afghanistan  2001 m      1577
## # ... with 6,911 more rows
```

```
whoex4 %>% filter(year>1990) %>%
  ggplot(aes(x = year, y = cases, group = country, color= country))   +geom_line()+ facet_wrap(~sex)+ tl
```