CPT_S 575 Data Science: Assignment 3

Reet Barik

September 19, 2019

Exercise 1

Print the first few values of the columns with a header including “sleep”. (head(), head())

```
msleep <- read.csv("https://scads.eecs.wsu.edu/wp-
content/uploads/2017/10/msleep_ggplot2.csv")
head(msleep)
```

```
##                         name      genus  vore        order conservation
## 1                    Cheetah   Acinonyx carni    Carnivora           lc
## 2                 Owl monkey      Aotus  omni     Primates         <NA>
## 3            Mountain beaver Aplodontia herbi     Rodentia           nt
## 4 Greater short-tailed shrew    Blarina  omni Soricomorpha           lc
## 5                        Cow        Bos herbi Artiodactyla domesticated
## 6           Three-toed sloth   Bradypus herbi       Pilosa         <NA>
##   sleep_total sleep_rem sleep_cycle awake brainwt  bodywt
## 1        12.1        NA          NA  11.9      NA  50.000
## 2        17.0       1.8          NA   7.0 0.01550   0.480
## 3        14.4       2.4          NA   9.6      NA   1.350
## 4        14.9       2.3   0.1333333   9.1 0.00029   0.019
## 5         4.0       0.7   0.6666667  20.0 0.42300 600.000
## 6        14.4       2.2   0.7666667   9.6      NA   3.850
```

a) Count the number of animals which weigh under 50 kilograms and sleep more than 16 hours a day. (filter(), query())

```
nrow(filter(msleep, bodywt < 50, sleep_total > 16))
```

```
## [1] 7
```

7 animals weigh under 50 kilograms and sleep for more than 16 hours

b) Print the name, order, sleep time and bodyweight of the animals with the 5 longest sleep times, in order of sleep time. (select(), arrange(), loc(), sort_values())

```
part_b = select(msleep, name, sleep_total, order, bodywt)
sleep_descending =  arrange(part_b, desc(sleep_total))
head(sleep_descending,5)
```

```
##                    name sleep_total           order bodywt
## 1       Little brown bat        19.9      Chiroptera  0.010
## 2          Big brown bat        19.7      Chiroptera  0.023
```

```
## 3   Thick-tailed opposum        19.4 Didelphimorphia  0.370
## 4        Giant armadillo        18.1       Cingulata 60.000
## 5 North American Opossum        18.0 Didelphimorphia  1.700
```

c) Add two new columns to the dataframe; wt_ratio with the ratio of brain size to body weight, rem_ratio with the ratio of rem sleep to sleep time. If you think they might be useful, feel free to extract more features than these, and describe what they are. (mutate(), assign())

```
msleep_ratio = mutate(msleep, wt_ratio = brainwt/bodywt, rem_ratio =
sleep_rem/sleep_total)
head(msleep_ratio[c('wt_ratio','rem_ratio')])
```

```
##     wt_ratio rem_ratio
## 1         NA        NA
## 2 0.03229167 0.1058824
## 3         NA 0.1666667
## 4 0.01526316 0.1543624
## 5 0.00070500 0.1750000
## 6         NA 0.1527778
```

d) Display the average, min and max sleep times for each order. (group_by(), summarise(), groupby(), agg())

```
groupDF = group_by(msleep,order)
summarize(groupDF, average_sleep = mean(sleep_total),
          min_sleep = min(sleep_total),
          max_sleep = max(sleep_total))
```

```
## # A tibble: 19 x 4
##    order           average_sleep min_sleep max_sleep
##    <fct>                   <dbl>     <dbl>     <dbl>
##  1 Afrosoricida            15.6       15.6      15.6
##  2 Artiodactyla             4.52       1.9       9.1
##  3 Carnivora               10.1        3.5      15.8
##  4 Cetacea                  4.5        2.7       5.6
##  5 Chiroptera              19.8       19.7      19.9
##  6 Cingulata               17.8       17.4      18.1
##  7 Didelphimorphia         18.7       18        19.4
##  8 Diprotodontia           12.4       11.1      13.7
##  9 Erinaceomorpha          10.2       10.1      10.3
## 10 Hyracoidea               5.67       5.3       6.3
## 11 Lagomorpha               8.4        8.4       8.4
## 12 Monotremata              8.6        8.6       8.6
## 13 Perissodactyla           3.47       2.9       4.4
## 14 Pilosa                  14.4       14.4      14.4
## 15 Primates                10.5        8        17
## 16 Proboscidea              3.6        3.3       3.9
## 17 Rodentia                12.5        7        16.6
```

```
## 18 Scandentia                8.9         8.9       8.9
## 19 Soricomorpha             11.1         8.4      14.9
```

e) Impute the missing brain weights as the average wt_ratio for that animal's order times the animal's weight. Make a second copy of your dataframe, but this time impute missing brain weights with the average brain weight for that animal's order. What assumptions do these data filling methods make? Which is the best way to impute the data, or do you see a better way, and why? You may impute or remove other variables as you find appropriate. Briefly explain your decisions. (group_by(), mutate(), groupby(),assign())

Imputing missing brain weights wtih (average weight ratio * body weight) of the animal:

```
method1 = msleep_ratio
method1 = method1 %>%
              select(name, order, brainwt, bodywt, wt_ratio) %>%
              group_by(order) %>%
              mutate(avg_wt_ratio=mean(wt_ratio, na.rm=TRUE)) %>%
              mutate(brainwt =
ifelse(is.na(brainwt),avg_wt_ratio*bodywt,brainwt))
head(method1)
```

```
## # A tibble: 6 x 6
## # Groups:   order [6]
##   name                  order         brainwt  bodywt wt_ratio
avg_wt_ratio
##   <fct>                 <fct>           <dbl>   <dbl>    <dbl>
<dbl>
## 1 Cheetah               Carnivora     0.371    50     NA
0.00743
## 2 Owl monkey            Primates      0.0155    0.48   3.23e-2      0.0186
## 3 Mountain beaver       Rodentia      0.0189    1.35  NA            0.0140
## 4 Greater short-tailed~ Soricomorp~   0.00029   0.019  1.53e-2      0.0166
## 5 Cow                   Artiodacty~   0.423   600      7.05e-4
0.00320
## 6 Three-toed sloth      Pilosa      NaN         3.85  NA          NaN
```

Imputing missing values of brain weights with average brain weight:

```
method2 = msleep_ratio
method2 = method2 %>%
               select(name, order, brainwt, bodywt, wt_ratio) %>%
               mutate(avg_brainwt =mean(wt_ratio, na.rm=TRUE)) %>%
               mutate(brainwt=ifelse(is.na(brainwt),avg_brainwt,brainwt))
head(method2)
```

```
##                          name        order    brainwt  bodywt   wt_ratio
## 1                     Cheetah    Carnivora 0.01035592  50.000         NA
## 2                  Owl monkey     Primates 0.01550000   0.480 0.03229167
## 3             Mountain beaver     Rodentia 0.01035592   1.350         NA
## 4 Greater short-tailed shrew Soricomorpha 0.00029000   0.019 0.01526316
```

```
## 5                         Cow Artiodactyla 0.42300000 600.000 0.00070500
## 6           Three-toed sloth       Pilosa 0.01035592   3.850         NA
##   avg_brainwt
## 1  0.01035592
## 2  0.01035592
## 3  0.01035592
## 4  0.01035592
## 5  0.01035592
## 6  0.01035592
```

I believe that replacing NA with the product of average brain weight and body weight (method1) takes into account another feature of the animal which is logically related to the feature with the missing values. It seems more sensible than just substituting the average brain weight.

Question 2

```
library("tidyr")
who_tidy = tidyr::who
who1 <- who_tidy %>%
  gather(key, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
  mutate(key = stringr::str_replace(key, "newrel", "new_rel")) %>%
  separate(key, c("new", "Type", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)

who1
```

```
## # A tibble: 76,046 x 6
##    country      year Type  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
```

a) Explain why this line " mutate(key = stringr::str_replace(key, “newrel”, “new_rel”))" is necessary to properly tidy the data. What happens if you skip this line?

The names become inconsistent because of newrel. All other values are in the form new_type. So while separating the new or old from the type, r would not be able to recognize where to separate the two as the sep factor provided in ‘_’. Thus, this step is necessary to tidy the data.

If this step was skipped, the 'new' would not have been separated from the 'rel' and the type column could have had NA entries, causing inconsistencies. Thus, newrel is replaced with new_rel.

b) How many entries are removed from the dataset when you set na.rm to true in the gather command (in this dataset)?

```
sum(is.na(who_tidy))
```

```
## [1] 329394
```

Number of entries removed from dataset: 329394

c) Explain the difference between an explicit and implicit missing value, in general. Can you find any implicit missing values in this dataset, if so where?

An **explicit** missing value is flagged as NA whereas an **implicit** missing value is simply not present in the data.

```
who_tidy %>%
  group_by(country) %>%
  summarise(min = min(year), max = max(year), distinct_years =
n_distinct(year)) %>%
  filter(min != 1980 | max != 2013 | distinct_years != 34)
```

```
## # A tibble: 9 x 4
##   country                              min   max distinct_years
##   <chr>                              <int> <int>          <int>
## 1 Bonaire, Saint Eustatius and Saba   2010  2013              4
## 2 Curacao                             2010  2013              4
## 3 Montenegro                          2005  2013              9
## 4 Netherlands Antilles                1980  2009             30
## 5 Serbia                              2005  2013              9
## 6 Serbia & Montenegro                 1980  2004             25
## 7 Sint Maarten (Dutch part)           2010  2013              4
## 8 South Sudan                         2011  2013              3
## 9 Timor-Leste                         2002  2013             12
```

d) Looking at the features (country, year, var, sex, age, cases) in the tidied data, are they all appropriately typed? Are there any features you think would be better suited as a different type? Why or why not?

```
who1
```

```
## # A tibble: 76,046 x 6
##    country      year Type  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
```

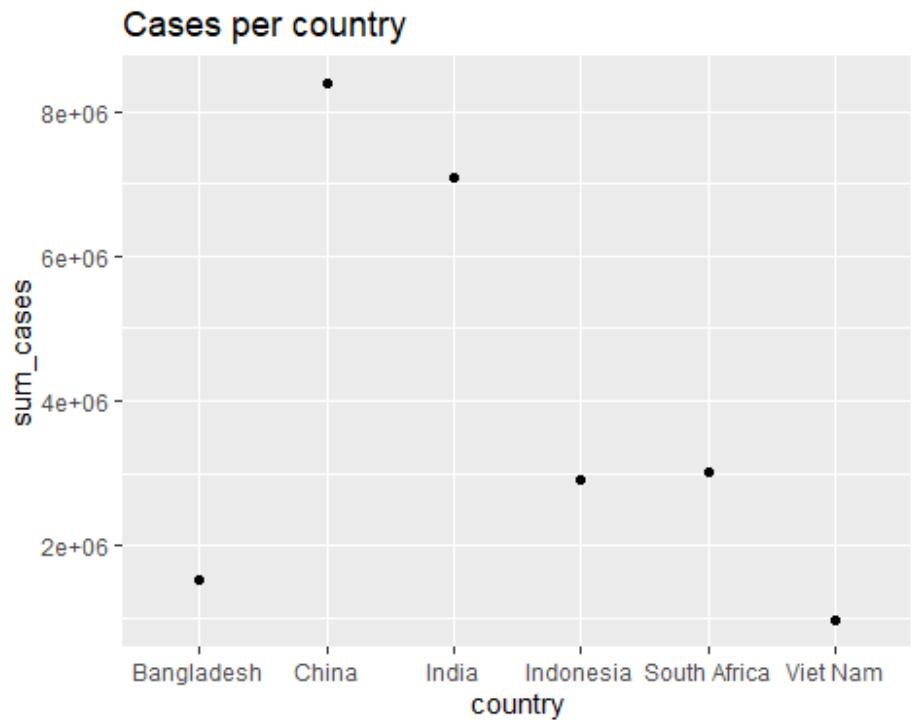
- Country : country of origin of the person
- Year : year in which tuberculosis was diagoised
- Type : type of tuberculosis
- sex : the gender of the person
- age : age of the person diagnosed
- value : count of the number of cases.

e) Explain in your own words what a gather operation is and give an example of a situation when it might be useful. Do the same for spread.

Gather is basically used to push data in columns into rows. It gathers multiple columns into key - value pairs. This function is needed when columns are not variables. The syntax of gather function is: gather(data, key = “key”, value = “value”, . . . , na.rm = FALSE, convert = FALSE, factor_key = FALSE For example, The column of a table are: ‘Year, quarter1, quarter2, quarter3, quarter4’ and the rows mention the year and give the revenue of each quarter. Using gather here to update the rows as Year, Quarter, Revenue makes more sense and the data looks more tidy and readable. Spread is complementary to the gather function. It spreads key-value pairs across multiple columns. The syntax of spread function is: spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL) This can be used to spread the revenue values in multiple quarter columns. This would get back the original table.

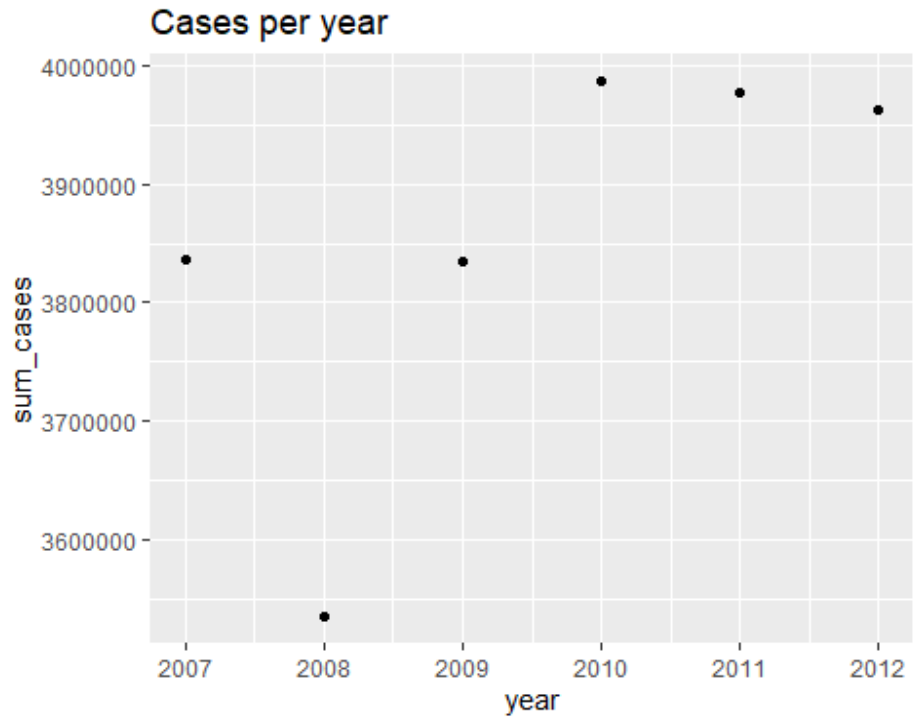
f) Generate an informative visualization, which shows something about the data. Give a brief description of what it shows, and why you thought it would be interesting to investigate.

Cases per country is analysed:



We can see that China has the highest number of cases followed by India.

Cases per year is analysed:



There was a drastic drop of cases in 2008, after which the number of cases have increased and remained high.

g) Suppose you have the following dataset called siteDemo: Site U30.F U30.M O30.F O30.M facebook 32 31 60 58 myspace 1 5 3 6 snapchat 6 4 3 2 twitter 17 23 12 17 You know that the U30.F column is the number of female users under 30 on the site, O30.M denotes the number of male users 30 or older on the site, etc. Construct this table, and show the code you would use to tidy this dataset (using gather(), separate() and mutate() or melt(), pivot(), and assign()) such that the columns are organized as: Site, AgeGroup, Gender and Count.

```
a = c("facebook","myspace", "snapchat", "twitter"  )
b = c( 32, 31, 60,58)
c = c( 1, 5, 3, 6)
d = c( 6, 4, 3,2)
e = c( 17, 23, 12,17)
siteDemo = data.frame( "Site" = a, "U30.F" = b, "U30.M" = c, "O30.F" =
d,"O30.M" = e)
siteDemo
```

```
##       Site U30.F U30.M O30.F O30.M
## 1 facebook    32     1     6    17
## 2  myspace    31     5     4    23
## 3 snapchat    60     3     3    12
## 4  twitter    58     6     2    17
```

```
siteDemo_tidy = siteDemo %>%
  gather(key, count, U30.F:O30.M) %>%
  separate(key, c("ageGroup", "gender")) %>%
  mutate(ageGroup = stringr::str_replace(ageGroup, "U30", "under 30")) %>%
  mutate(ageGroup = stringr::str_replace(ageGroup, "O30", "Over 30")) %>%
  mutate(gender = stringr::str_replace(gender, "F", "Female")) %>%
  mutate(gender = stringr::str_replace(gender, "M", "Male"))

siteDemo_tidy
```

```
##        Site ageGroup gender count
## 1  facebook under 30 Female    32
## 2   myspace under 30 Female    31
## 3  snapchat under 30 Female    60
## 4   twitter under 30 Female    58
## 5  facebook under 30   Male     1
## 6   myspace under 30   Male     5
## 7  snapchat under 30   Male     3
## 8   twitter under 30   Male     6
## 9  facebook  Over 30 Female     6
## 10  myspace  Over 30 Female     4
## 11 snapchat  Over 30 Female     3
## 12  twitter  Over 30 Female     2
```

```
## 13 facebook  Over 30   Male    17
## 14  myspace  Over 30   Male    23
## 15 snapchat  Over 30   Male    12
## 16  twitter  Over 30   Male    17
```