Assignment 3: Data Transformation and Tidying

WenLi

2020/9/16

1 Question 1

For this question you will be using either the dplyr package from R or the Pandas library in python to manipulate and clean up a dataset called msleep

1.1 Load the data into R and check abnormalities

```
Msleep  = read.csv("msleep_ggplot2.csv", header = TRUE)
head (as_tibble(Msleep))
```

```
## # A tibble: 6 x 11
##   name  genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##   <chr> <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
## 1 Chee~ Acin~ carni Carn~ lc                  12.1      NA        NA      11.9
## 2 Owl ~ Aotus omni  Prim~ <NA>                17         1.8      NA       7
## 3 Moun~ Aplo~ herbi Rode~ nt                  14.4       2.4      NA       9.6
## 4 Grea~ Blar~ omni  Sori~ lc                  14.9       2.3       0.133   9.1
## 5 Cow   Bos   herbi Arti~ domesticated         4         0.7       0.667  20
## 6 Thre~ Brad~ herbi Pilo~ <NA>                14.4       2.2       0.767   9.6
## # ... with 2 more variables: brainwt <dbl>, bodywt <dbl>
```

1.2 Print the first few values of the columns with "sleep"

```
Sleep = select(Msleep, contains("sleep"))
head (as_tibble(Sleep))
```

```
## # A tibble: 6 x 3
##   sleep_total sleep_rem sleep_cycle
##         <dbl>     <dbl>       <dbl>
## 1        12.1      NA        NA
## 2        17         1.8      NA
## 3        14.4       2.4      NA
## 4        14.9       2.3       0.133
## 5         4         0.7       0.667
## 6        14.4       2.2       0.767
```

1.3 Count the number of animals

```
Animals = filter(Msleep, bodywt < 1, sleep_total > 14)
count(Animals)
```

```
##    n
## 1 14
```

1.4 Print table in order of sleep time

```
Animals = Msleep%>%
          select("name", "order", "sleep_total", "bodywt")%>%
          arrange(desc(sleep_total))
head (as_tibble(Animals), n = 6)
```

```
## # A tibble: 6 x 4
##   name                  order           sleep_total bodywt
##   <chr>                 <chr>                 <dbl>  <dbl>
## 1 Little brown bat      Chiroptera             19.9  0.01
## 2 Big brown bat         Chiroptera             19.7  0.023
## 3 Thick-tailed opposum  Didelphimorphia        19.4  0.37
## 4 Giant armadillo       Cingulata              18.1 60
## 5 North American Opossum Didelphimorphia       18    1.7
## 6 Long-nosed armadillo  Cingulata             17.4  3.5
```

1.5 Print table in order of sleep time

```
NewCols = Msleep%>%
          select("name","sleep_total","sleep_rem","awake","bodywt","brainwt")%>%
          mutate(wt_ratio = brainwt/bodywt,
                 rem_ratio = sleep_rem/sleep_total,
                 ak_ratio=awake/sleep_total)
head(as_tibble(NewCols))
```

```
## # A tibble: 6 x 9
##   name  sleep_total sleep_rem awake  bodywt  brainwt wt_ratio rem_ratio ak_ratio
##   <chr>       <dbl>     <dbl> <dbl>   <dbl>    <dbl>    <dbl>     <dbl>    <dbl>
## 1 Chee~        12.1      NA    11.9  50     NA       NA          NA        0.983
## 2 Owl ~        17         1.8   7     0.48   0.0155   3.23e-2     0.106    0.412
## 3 Moun~        14.4       2.4   9.6   1.35  NA       NA           0.167    0.667
## 4 Grea~        14.9       2.3   9.1   0.019  0.00029  1.53e-2     0.154    0.611
## 5 Cow           4         0.7  20   600      0.423    7.05e-4     0.175    5
## 6 Thre~        14.4       2.2   9.6   3.85  NA       NA           0.153    0.667
```

```
#ak_ratio: the ratio of awake time to sleep time
```

1.6 Display the average, min and max sleep times for each order

```
SumOrders = Msleep%>%
            group_by(order)%>%
            summarise(aveargе=mean(sleep_total), min=min(sleep_total),
                      max=max(sleep_total), total=n())%>%
print(SumOrders)
```

```
## # A tibble: 19 x 5
##    order           aveargе   min   max total
##    <chr>             <dbl> <dbl> <dbl> <int>
##  1 Afrosoricida      15.6   15.6  15.6     1
##  2 Artiodactyla       4.52   1.9   9.1     6
##  3 Carnivora         10.1    3.5  15.8    12
##  4 Cetacea            4.5    2.7   5.6     3
##  5 Chiroptera        19.8   19.7  19.9     2
##  6 Cingulata         17.8   17.4  18.1     2
```

```
##  7 Didelphimorphia   18.7   18    19.4     2
##  8 Diprotodontia     12.4   11.1  13.7     2
##  9 Erinaceomorpha    10.2   10.1  10.3     2
## 10 Hyracoidea         5.67   5.3   6.3     3
## 11 Lagomorpha         8.4    8.4   8.4     1
## 12 Monotremata        8.6    8.6   8.6     1
## 13 Perissodactyla     3.47   2.9   4.4     3
## 14 Pilosa            14.4   14.4  14.4     1
## 15 Primates          10.5    8    17      12
## 16 Proboscidea        3.6    3.3   3.9     2
## 17 Rodentia          12.5    7    16.6    22
## 18 Scandentia         8.9    8.9   8.9     1
## 19 Soricomorpha      11.1    8.4  14.9     5
```

1.7 Impute the missing brain weights

```
# Impute the missing brain weights as the average wt_ratio
AvgWtRatio = Msleep%>%
  group_by(order)%>%
  select("name", "order", "brainwt", "bodywt")%>%
  mutate(brainwt=ifelse(is.na(brainwt),
                        ifelse(is.nan(mean(brainwt, na.rm = TRUE)),0,
                  mean(brainwt,na.rm = TRUE))/mean(bodywt, na.rm = TRUE)*bodywt,
                        brainwt))%>%
  ungroup(order)
head(as_tibble(AvgWtRatio))
```

```
## # A tibble: 6 x 4
##   name                        order        brainwt  bodywt
##   <chr>                       <chr>          <dbl>   <dbl>
## 1 Cheetah                     Carnivora    0.0854   50
## 2 Owl monkey                  Primates     0.0155    0.48
## 3 Mountain beaver             Rodentia     0.0167    1.35
## 4 Greater short-tailed shrew Soricomorpha 0.00029   0.019
## 5 Cow                         Artiodactyla 0.423   600
## 6 Three-toed sloth            Pilosa       0         3.85
```

```
# Impute the missing brain weights as the average brain weight
AvgBwt = Msleep%>%
         group_by(order)%>%
         select("name", "order", "brainwt", "bodywt")%>%
         mutate(brainwt=ifelse(is.na(brainwt),
                               ifelse(is.nan(mean(brainwt,na.rm = TRUE)),0,
                                      mean(brainwt,na.rm = TRUE)),
                               brainwt))%>%
         ungroup(order)
head(as_tibble(AvgBwt))
```

```
## # A tibble: 6 x 4
##   name                        order        brainwt  bodywt
##   <chr>                       <chr>          <dbl>   <dbl>
## 1 Cheetah                     Carnivora    0.0986   50
## 2 Owl monkey                  Primates     0.0155    0.48
## 3 Mountain beaver             Rodentia     0.00357   1.35
## 4 Greater short-tailed shrew Soricomorpha 0.00029   0.019
```

```
## 5 Cow                          Artiodactyla 0.423   600
## 6 Three-toed sloth             Pilosa       0         3.85
```

When we impute the missing brain weights as average wt_ratio, we assume that in a specific order, wt_ratio of the samples are similar; while choosing the average brain weight, the brain weight of the samples are similar. Here imputation by average wt_ratio is a better way for the brain weight which usually associates with the body weight. But usually this "Mean Imputation" is not a good choice in practice. there are many other better solutions (e.g., "Hot deck imputation", "Regression imputation").

```
# Impute the missing sleep_rem as average sleep rem
SleepRem = Msleep%>%
           group_by(order)%>%
           select("name", "order", "sleep_rem")%>%
           mutate(sleep_rem=ifelse(is.na(sleep_rem),
                                   ifelse(is.nan(mean(sleep_rem, na.rm = TRUE)),0,
                                          mean(sleep_rem,na.rm = TRUE)),
                                    sleep_rem))%>%
           ungroup(order)
head(as_tibble(SleepRem))
```

```
## # A tibble: 6 x 3
##   name                          order        sleep_rem
##   <chr>                         <chr>            <dbl>
## 1 Cheetah                       Carnivora         1.87
## 2 Owl monkey                    Primates          1.8
## 3 Mountain beaver               Rodentia          2.4
## 4 Greater short-tailed shrew    Soricomorpha      2.3
## 5 Cow                           Artiodactyla      0.7
## 6 Three-toed sloth              Pilosa            2.2
```

Question 2

For this question, you will first need to read section 12.6 in the R for Data Science book, here (http://r4ds.had.co.nz/tidy-data.html#case-study). Grab the dataset from the tidyr package (tidyr::who), and tidy it as shown in the case study before answering the following questions.

1.1 Load the data into R and Tidy the dataset

```
Who = tidyr::who
TidyDataNaF = Who %>%
              gather(key, value, new_sp_m014:newrel_f65, na.rm = FALSE) %>%
              mutate(key = stringr::str_replace(key, "newrel", "new_rel")) %>%
              separate(key, c("new", "var", "sexage")) %>%
              select(-new, -iso2, -iso3) %>%
              separate(sexage, c("sex", "age"), sep = 1)
TidyDataNaF
```

```
## # A tibble: 405,440 x 6
##    country      year var   sex   age   value
##    <chr>       <int> <chr> <chr> <chr> <int>
##  1 Afghanistan  1980 sp    m     014      NA
##  2 Afghanistan  1981 sp    m     014      NA
##  3 Afghanistan  1982 sp    m     014      NA
##  4 Afghanistan  1983 sp    m     014      NA
##  5 Afghanistan  1984 sp    m     014      NA
##  6 Afghanistan  1985 sp    m     014      NA
```

```
##  7 Afghanistan  1986 sp    m     014      NA
##  8 Afghanistan  1987 sp    m     014      NA
##  9 Afghanistan  1988 sp    m     014      NA
## 10 Afghanistan  1989 sp    m     014      NA
## # ... with 405,430 more rows
```

1.2 Explain mutate(key = stringr::str_replace(key, "newrel", "new_rel")).

It replace the header with name "newrel" into "new_rel", which can make the format of all headers consistent. If it is skipped, we can not extract detailed features by simply separate the header name. For example, given a header name of "new_sp_f5564" we can apply separate function on it and extract three new features:new, sp, f5564, while in case of "newrel_m014" we can not.

1.3 How many entries are removed from the dataset when you set na.rm to true

```
TidyDataNaT = Who %>%
              gather(key, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
              mutate(key = stringr::str_replace(key, "newrel", "new_rel")) %>%
              separate(key, c("new", "var", "sexage")) %>%
              select(-new, -iso2, -iso3) %>%
              separate(sexage, c("sex", "age"), sep = 1)
# When set na.rm=TRUE, the remove entries can be computed as:
count(TidyDataNaF) - count(TidyDataNaT)
```

```
##        n
## 1 329394
```

1.4 Explain the difference between an explicit and implicit missing value

According to the "R for Data Science, An explicit missing value is the presence of an absence; an implicit missing value is the absence of a presence", That means for explicit missing, there will be a specific representation to indicate the missing of the value (e.g., NA). While for implicit missing, there will be no specific representation for the value.

In this dataset, we can consider the value == 0 as the implicit missing value, we can get the sub-samples with following command.

```
Who %>%
gather(key, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
filter(value == 0)
```

```
## # A tibble: 11,080 x 6
##    country     iso2  iso3   year key         value
##    <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
```

1.5 Looking at the featuresin the tidied data, are they all appropriately typed?

```
head(as_tibble(TidyDataNaT))
```

```
## # A tibble: 6 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
```

```
sapply(TidyDataNaT, class)
```

```
##     country        year         var         sex         age       value
## "character"   "integer" "character" "character" "character"   "integer"
```

As shown above, all features are typed appropriately except the age which is typed as character. Usually it is better to type this feature as integer, when we try to analyze this feature (e.g., distribution of the age, average age), we must compute the results with integer.

1.6 Explain in your own words what a gather operation is.

gather During data tidy, when we get a data frame where there are lots of variable with the same features, we can use gather to translate these variables as data values (translate columns into rows). for example, the following data frame present the variation of fruit price.

```
Fruits = tibble(time = as.Date('2020-09-01') + 0:2,
                apple  = rnorm(3, 0, 1),
                peach  = rnorm(3, 0, 2),
                banana = rnorm(3, 0, 3))
print (Fruits)
```

```
## # A tibble: 3 x 4
##   time        apple peach banana
##   <date>      <dbl> <dbl>  <dbl>
## 1 2020-09-01 -0.328  1.12  2.25
## 2 2020-09-02 -1.37   3.52 -0.200
## 3 2020-09-03  0.100 -1.52  3.24
```

In this data frame, the three variable apple, peach and banana are all fruits and this format is usually not what we want, so we can apply gather function on it as following:

```
gather(Fruits, key = "fruit", value = "price_var", "apple":"banana")
```

```
## # A tibble: 9 x 3
##   time       fruit  price_var
##   <date>     <chr>      <dbl>
## 1 2020-09-01 apple     -0.328
## 2 2020-09-02 apple     -1.37
## 3 2020-09-03 apple      0.100
## 4 2020-09-01 peach      1.12
## 5 2020-09-02 peach      3.52
## 6 2020-09-03 peach     -1.52
## 7 2020-09-01 banana     2.25
## 8 2020-09-02 banana    -0.200
```

```
## 9 2020-09-03 banana     3.24
```

spread During data tidy, spread has the similar functionality as gather but it translate the data in rows into columns, which is opposite to gather. for example:

```
Fruits = tibble( fruit = c('apple', 'apple', 'peach', 'peach', 'banana', 'banana'),
                 time  = c('2020', '2020', '2019', '2019', '2018', '2018'),
                 key   = c('yield', 'price', 'yield', 'price', 'yield', 'price'),
                 value = c(100, 2, 150, 1, 300, 3))
print (Fruits)
```

```
## # A tibble: 6 x 4
##   fruit  time  key   value
##   <chr>  <chr> <chr> <dbl>
## 1 apple  2020  yield   100
## 2 apple  2020  price     2
## 3 peach  2019  yield   150
## 4 peach  2019  price     1
## 5 banana 2018  yield   300
## 6 banana 2018  price     3
```

In this data frame, the variable "key" has only two different values and the variable itself is not meaningful, in this case we can translate the data frame with spread as following:

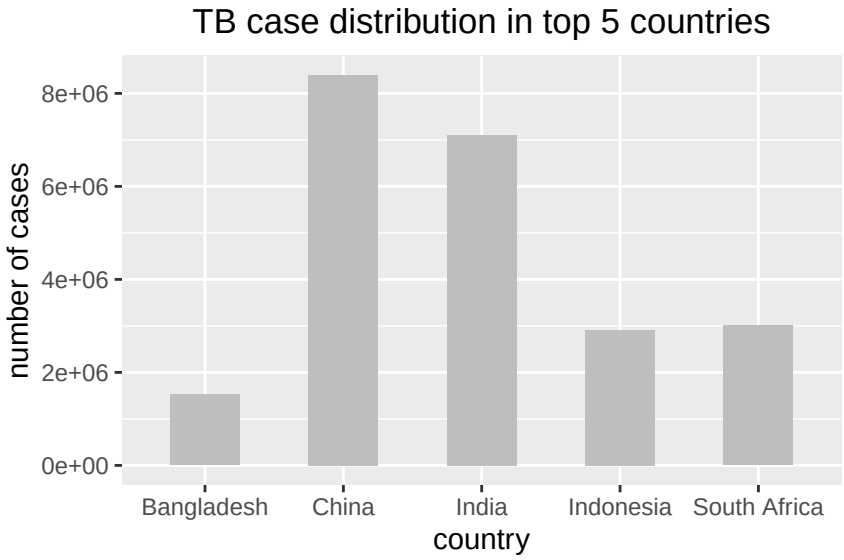
```
spread(Fruits, key, value)
```

```
## # A tibble: 3 x 4
##   fruit  time  price yield
##   <chr>  <chr> <dbl> <dbl>
## 1 apple  2020      2   100
## 2 banana 2018      3   300
## 3 peach  2019      1   150
```

1.7 Generate an informative visualization.

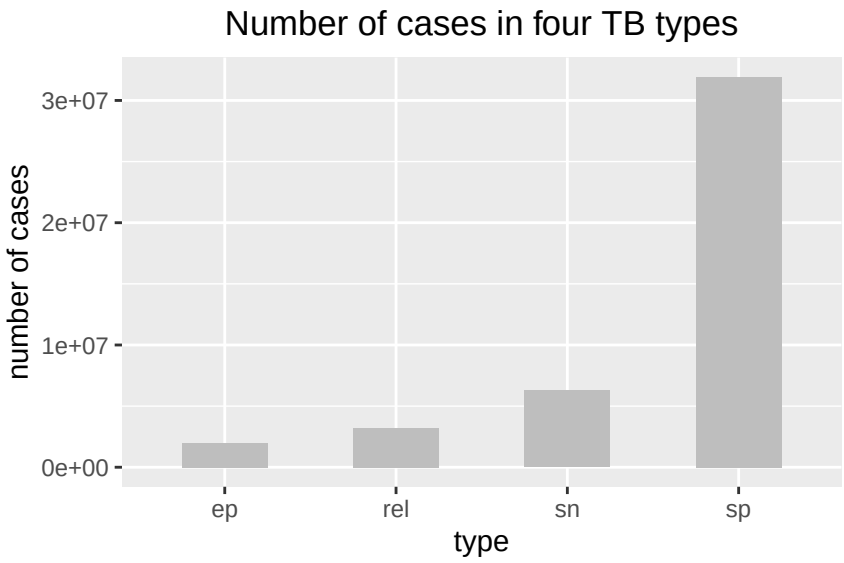
Look at the data grouped by country In this case, top TB cases distribution by countries are shown as following graph, from which we can know which countries are worse off.

```
library(ggplot2)
VDCountry = TidyDataNaT%>%
            group_by(country)%>%
            tally(value)%>%
            top_n(5)
ggplot(data=VDCountry, aes(x=country, y=n)) +
       geom_bar(stat="identity",width=0.5,fill="gray") +
       labs(title="TB case distribution in top 5 countries",
            x="country", y="number of cases") +
       theme(plot.title = element_text(hjust = 0.5))
```



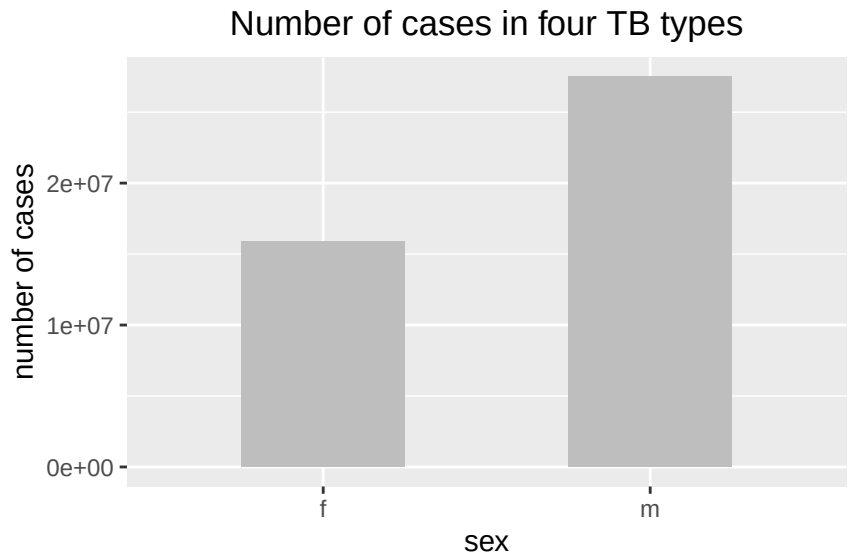
Look at the data grouped by TB type the following graph shows TB case distribution among four TB types, sp is the most common type in call cases.

```
library(ggplot2)
VDType = TidyDataNaT%>%
         group_by(var)%>%
         tally(value)
ggplot(data=VDType, aes(x=var, y=n)) +
       geom_bar(stat="identity",width=0.5,fill="gray") +
       labs(title="Number of cases in four TB types",
            x="type", y="number of cases") +
       theme(plot.title = element_text(hjust = 0.5))
```



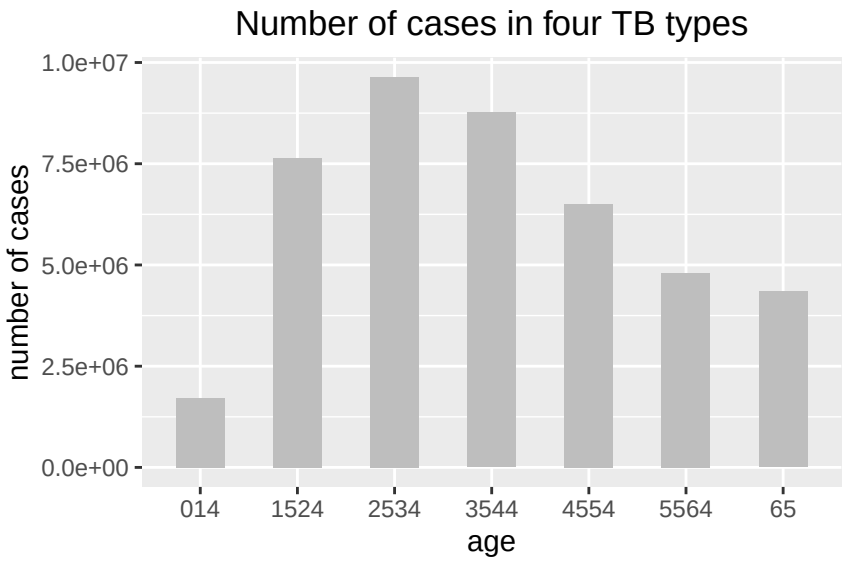
Look at the data grouped by sex the following graph shows TB case distribution among female and male, it indicates that male are more easily affected.

```
library(ggplot2)
VDSex = TidyDataNaT%>%
        group_by(sex)%>%
        tally(value)
ggplot(data=VDSex, aes(x=sex, y=n)) +
       geom_bar(stat="identity",width=0.5,fill="gray") +
       labs(title="Number of cases in four TB types",
            x="sex", y="number of cases") +
       theme(plot.title = element_text(hjust = 0.5))
```



Look at the data grouped by age the following graph shows TB case distribution among different ages, it indicates that people with young and the middle-aged are more possible to be affected.

```
library(ggplot2)
VDAge = TidyDataNaT%>%
        group_by(age)%>%
        tally(value)
ggplot(data=VDAge, aes(x=age, y=n)) +
       geom_bar(stat="identity",width=0.5,fill="gray") +
       labs(title="Number of cases in four TB types",
            x="age", y="number of cases") +
       theme(plot.title = element_text(hjust = 0.5))
```



1.8 Suppose you have the following dataset called siteDemo

```
siteDemo = tibble(Site = c('facebook', 'myspace', 'snapchat', 'twitter', 'tiktok'),
                  U30.F = c(30, 1, 6, 18, 44),
                  U30.M = c(35, 2, 5, 23, 60),
                  O30.F = c(66, 3, 3, 12, 2),
                  O30.M = c(58, 6, 2, 28, 7))
print (siteDemo)
```

Construct the data frame:

```
## # A tibble: 5 x 5
##   Site     U30.F U30.M O30.F O30.M
##   <chr>    <dbl> <dbl> <dbl> <dbl>
## 1 facebook    30    35    66    58
## 2 myspace      1     2     3     6
## 3 snapchat     6     5     3     2
## 4 twitter     18    23    12    28
## 5 tiktok      44    60     2     7
```

```
TidyData = siteDemo %>%
           gather(key, Count, U30.F:O30.M, na.rm = TRUE) %>%
           separate(key, c("AgeGroup", "Gender"))
TidyData
```

organize the data frame as: Site, AgeGroup, Gender and Count.

```
## # A tibble: 20 x 4
##    Site     AgeGroup Gender Count
##    <chr>    <chr>    <chr>  <dbl>
##  1 facebook U30      F         30
##  2 myspace  U30      F          1
##  3 snapchat U30      F          6
##  4 twitter  U30      F         18
##  5 tiktok   U30      F         44
```

```
##  6 facebook U30      M         35
##  7 myspace  U30      M          2
##  8 snapchat U30      M          5
##  9 twitter  U30      M         23
## 10 tiktok   U30      M         60
## 11 facebook O30      F         66
## 12 myspace  O30      F          3
## 13 snapchat O30      F          3
## 14 twitter  O30      F         12
## 15 tiktok   O30      F          2
## 16 facebook O30      M         58
## 17 myspace  O30      M          6
## 18 snapchat O30      M          2
## 19 twitter  O30      M         28
## 20 tiktok   O30      M          7
```