assign3.R

harry

2020-09-19

```
library("tidyverse")
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(dplyr)
library(imager)
```

```
## Loading required package: magrittr

##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract

##
## Attaching package: 'imager'

## The following object is masked from 'package:magrittr':
##
##     add

## The following object is masked from 'package:stringr':
##
##     boundary
```

```
## The following object is masked from 'package:tidyr':
##
##     fill

## The following objects are masked from 'package:stats':
##
##     convolve, spectrum

## The following object is masked from 'package:graphics':
##
##     frame

## The following object is masked from 'package:base':
##
##     save.image
```

```
options(scipen = 999)
```

Question 1. (60 pts total) 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 (mammals sleep) that is available on the course webpage at https://scads.eecs.wsu.edu/wp-content/uploads/2017/10/msleep_ggplot2.csv Below are the tasks to perform. Before you begin, print the first few values of the columns with a header including "sleep". (head(), head())

```
msleep<-read.csv("msleep_ggplot2.csv", header=TRUE)
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) (10 pts) Count the number of animals which weigh under 1 kilogram and sleep more than 14 hours a day. (filter(), query())

```
nrow(filter(msleep, msleep$bodywt < 1 & msleep$sleep_total > 14))
```

```
## [1] 14
```

```
#The number of animals that weigh under 1 kilo and sleep for more than 14 hours a day is 14
```

b) (10 pts) Print the name, order, sleep time and bodyweight of the animals with the 6 longest sleep times, in order of sleep time. (select(), arrange(), loc(), sort_values())

```
print(msleep[order(decreasing = TRUE,msleep$sleep_total),]
      [1:6,][,c("name","order","sleep_total","bodywt")])
```

```
##                       name           order sleep_total bodywt
## 43       Little brown bat      Chiroptera        19.9  0.010
## 22          Big brown bat      Chiroptera        19.7  0.023
## 37   Thick-tailed opposum Didelphimorphia        19.4  0.370
## 62        Giant armadillo       Cingulata        18.1 60.000
## 20 North American Opossum Didelphimorphia        18.0  1.700
## 18   Long-nosed armadillo       Cingulata        17.4  3.500
```

c) (10 pts) 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 <- transform(msleep, wt_ratio = brainwt/bodywt)
msleep <- transform(msleep, rem_ratio = sleep_total/sleep_rem)

#I also decided to add the number of sleep cycles undertaken
msleep <- transform(msleep, cycle_count = sleep_total/sleep_cycle)

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   wt_ratio rem_ratio
## 1        12.1        NA          NA  11.9      NA  50.000         NA        NA
## 2        17.0       1.8          NA   7.0 0.01550   0.480 0.03229167  9.444444
## 3        14.4       2.4          NA   9.6      NA   1.350         NA  6.000000
## 4        14.9       2.3   0.1333333   9.1 0.00029   0.019 0.01526316  6.478261
## 5         4.0       0.7   0.6666667  20.0 0.42300 600.000 0.00070500  5.714286
## 6        14.4       2.2   0.7666667   9.6      NA   3.850         NA  6.545455
##   cycle_count
## 1          NA
## 2          NA
## 3          NA
## 4   111.75000
## 5     6.00000
## 6    18.78261
```

d) (14 pts) Display the average, min and max sleep times for each order. (group_by(), summarise(), groupby(), agg())

```
by_order <- msleep %>% group_by(msleep$order)
by_order_summary <- by_order %>% summarise(mean_sleep = mean(sleep_total,
                                                             na.rm = TRUE),
                                           min_Sleep = min(sleep_total),
                                           max_sleep = max(sleep_total))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
print(by_order_summary)
```

```
## # A tibble: 19 x 4
##    `msleep$order`  mean_sleep min_Sleep max_sleep
##    <chr>                <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) (16 pts) 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())

```
by_orderE_summary<-by_order %>% summarise(mean_wt_ratio = mean(wt_ratio,na.rm = TRUE),
                                          mean_brainwt = mean(brainwt,na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
#brain wt = avg weight ratio for order * bodywt

#here we need to remove rows without bodywt since we would be unable to calculate brainwt without them.
msleepE_byavg_wt_ratio <-msleep[!is.na(msleep$bodywt),]

for(i in 1:length(msleepE_byavg_wt_ratio$brainwt)) {
```

```
  if(is.na(msleepE_byavg_wt_ratio$brainwt[i])) {
    msleepE_byavg_wt_ratio$brainwt[i] <- by_orderE_summary$mean_wt_ratio[which(by_orderE_summary$`msleep
  }
}
head(msleepE_byavg_wt_ratio)
```

```
##                         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   wt_ratio
## 1        12.1        NA          NA  11.9 0.37129771  50.000         NA
## 2        17.0       1.8          NA   7.0 0.01550000   0.480 0.03229167
## 3        14.4       2.4          NA   9.6 0.01892814   1.350         NA
## 4        14.9       2.3   0.1333333   9.1 0.00029000   0.019 0.01526316
## 5         4.0       0.7   0.6666667  20.0 0.42300000 600.000 0.00070500
## 6        14.4       2.2   0.7666667   9.6        NaN   3.850         NA
##   rem_ratio cycle_count
## 1        NA          NA
## 2  9.444444          NA
## 3  6.000000          NA
## 4  6.478261   111.75000
## 5  5.714286     6.00000
## 6  6.545455    18.78261
```

```
#brain wt = avg weight ratio for order
msleepE_byavg_brainwt <- msleep

for(i in 1:length(msleepE_byavg_brainwt$brainwt)) {
  if(is.na(msleepE_byavg_brainwt$brainwt[i])) {
    msleepE_byavg_brainwt$brainwt[i] <- by_orderE_summary$mean_brainwt[
      which(by_orderE_summary$`msleep$order` == msleepE_byavg_brainwt$order[i])]
  }
}

head(msleepE_byavg_brainwt)
```

```
##                         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   wt_ratio
## 1        12.1        NA          NA  11.9 0.09857143  50.000         NA
## 2        17.0       1.8          NA   7.0 0.01550000   0.480 0.03229167
## 3        14.4       2.4          NA   9.6 0.00356800   1.350         NA
## 4        14.9       2.3   0.1333333   9.1 0.00029000   0.019 0.01526316
```

```
## 5         4.0       0.7   0.6666667  20.0 0.42300000 600.000 0.00070500
## 6        14.4       2.2   0.7666667   9.6        NaN   3.850         NA
##   rem_ratio cycle_count
## 1        NA          NA
## 2  9.444444          NA
## 3  6.000000          NA
## 4  6.478261   111.75000
## 5  5.714286     6.00000
## 6  6.545455    18.78261
```

```
#the first makes the assumption that there is a relationship between body mass and brain mass that
#is consistent across all creatures under a particular order.
#it uses it to calculate brain weights based on this presumption.

#the second makes the assumption that every animal under an order has on average, the same sized
#brain and then applies it to all animals in the order.

#of these two,  the better is likely the first method as using the relationship is more likely
#to be closer to the truth than applying a broad average. However, it
#isn't perfect either. What would be better would be to determine the greatest predictors of
#brain weight and construct a pattern from them as only using two variables (brain weight and body weig
#might not show us the entire picture.
```

Question 2. (40 pts total) 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. Note: if you are using pandas you can perform these same operations, just replace the pivot_longer() function with melt() and the pivot_wider() function with pivot(). However, you may prefer to use R for this question, as the dataset is from an R package. a) (5 pts) 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?

```
who <- tidyr::who
 #we need to correct the inconsistency in the col names where we expect new_rel but
#instead have newrel. not adding this line would break things later on.
```

b) (5 pts) How many entries are removed from the dataset when you set values_drop_na to true in the pivot_longer command (in this dataset)?

```
#with drop_na = FALSE
whoTEST <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = FALSE
  )


who1 <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
```

```
    values_drop_na = TRUE
  )
nrow(whoTEST) - nrow(who1)
```

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

```
#329394 rows are removed from the data set


who2 <- who1 %>%
  mutate(names_from = stringr::str_replace(key, "newrel", "new_rel"))

who3 <- who2 %>%
  separate(key, c("new", "type", "sexage"), sep = "_")
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 2580 rows [243,
## 244, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 903,
## 904, 905, 906, ...].
```

```
who4 <- who3 %>%
  select(-new, -iso2, -iso3)

who5 <- who4 %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

c) (5 pts) 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?

```
#implicit missing values are ones whose presence isn't directly expressed but
#rather implied, this would include an absence of recorded results from a
#particular region.

#explicit missing values however, are entries where it is made clear
#that something is missing, such as NA's in a col on a row.


#we expect there to be 56 entries for each country for each year given
#that that is the number of possible combinations in the names col.
#thus we can check to see if any of the countries are missing data for any years.
#we know our year range is from 1980 to 2013 and we know we include 219 countries,
#we can calculate the total expected results and determine if
#we have less than expected.

who5 %>%
  distinct(who5$country) %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   219
```

```r
#we expect this many results
expected <- 219 * 34 * 56

#we have this many results
nrow(who5)
```

```
## [1] 76046
```

```r
#thus we can conclude we are missing
expected - nrow(who5)
```

```
## [1] 340930
```

```r
#rows from our dataset.
```

d) (5 pts) 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?

```r
# we could use some integer key representations of some of our variables,
#like var and sex to save memory. separating var, sex ,
#and age could lead to different evaluations too.
```

e) (10 pts) 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

```r
#I decided to compare the cases change between post soviet states that joined
#the EU and those who didn't.
# 3 Nations joined the EU, fortunately for this comparison,
#they all joined in 2004.
#Thus, I combined the case counts for Lithuania, Estonia and Latvia and
#compared them to the combined case counts of the other former blocks,
#to determine if their rates declined faster than their standalone counterparts.

#as the plots show, both groups experienced a large outbreak in 2005,
#however the effectiveness of treating it is markedly different.

eu_block_nations <- c("Latvia","Estonia","Lithuania")
former_block_nations <-c("Republic of Moldova","Ukraine","Belarus",
                         "Uzbekistan","Kazakhstan","Georgia",
                         "Azerbaijan","Kyrgyzstan","Tajikistan",
                         "Armenia","Turkmenistan")



eu_data <- subset(who5,who5$country %in% eu_block_nations & who5$year >= 2004)
block_data <-subset(who5,who5$country %in% former_block_nations & who5$year >= 2004)


eu_by_year <- eu_data %>% group_by(eu_data$year)
block_by_year <- block_data %>% group_by(block_data$year)

eu_by_year_summary <- eu_by_year %>% summarise(sum_cases = sum(cases))
```

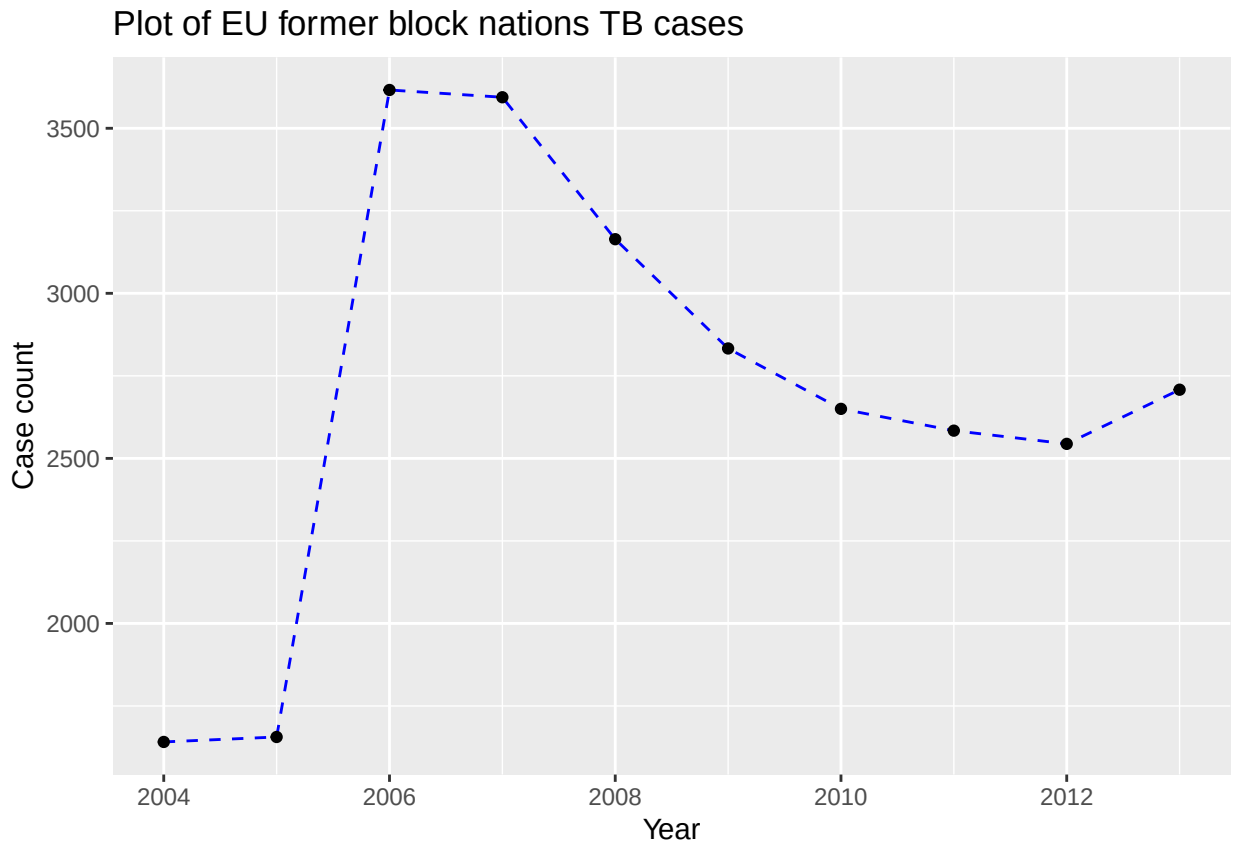
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
block_by_year_summary <- block_by_year %>% summarise(sum_cases = sum(cases))
```

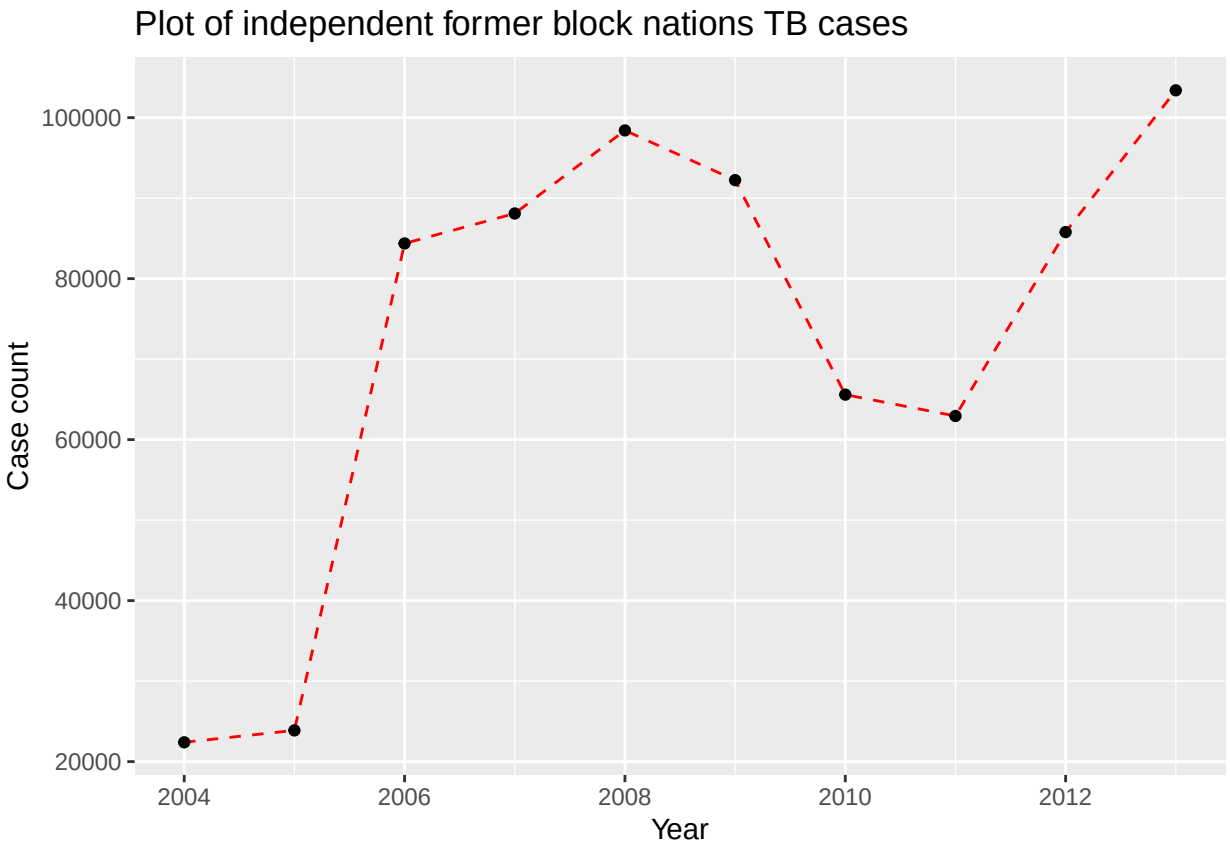
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
x <- 2004:2013


# Basic line plot with points
print(plt2 <- ggplot(data=eu_by_year_summary, aes(x=`eu_data$year`,
                                                  y=sum_cases, group=1)) +
        geom_line(linetype = "dashed",color="blue")+
        labs(title="Plot of EU former block nations TB cases",x="Year",
             y = "Case count")+
        geom_point())
```




```
print(plt3 <- ggplot(data=block_by_year_summary, aes(x=`block_data$year`,
                                                     y=sum_cases, group=1)) +
        geom_line(linetype = "dashed",color="red")+
        labs(title="Plot of independent former block nations TB cases",x="Year",
             y = "Case count")+
        geom_point())
```


Plot of independent former block nations TB cases
Case count
100000
80000
60000
40000
20000
2004
2006
2008
2010
2012
Year