Lab 3: Data Frames and Apply

Statistical Computing, 36-350

Name: Rufus Petrie

This week's agenda: getting familiar with data frames; practicing how to use the apply family of functions.

States data set

Below we construct a data frame, of 50 states x 10 variables. The first 8 variables are numeric and the last 2 are factors. The numeric variables here come from the built-in `state.x77` matrix, which records various demographic factors on 50 US states, measured in the 1970s. You can learn more about this state data set by typing `?state.x77` into your R console.

```
state.df = data.frame(state.x77, Region=state.region, Division=state.division)
```

Q1. Basic data frame manipulations

- **1a.** Add a column to `state.df`, containing the state abbreviations that are stored in the built-in vector `state.abb`. Name this column `Abbr`. You can do this in (at least) two ways: by using a call to `data.frame()`, or by directly defining `state.df$Abbr`. Display the first 3 rows and all 11 columns of the new `state.df`.

```
state.df$Abbr <- state.abb
head(state.df, 3)
```

```
##         Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## Alabama       3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska         365   6315        1.5    69.31   11.3    66.7   152 566432
## Arizona       2212   4530        1.8    70.55    7.8    58.1    15 113417
##         Region           Division Abbr
## Alabama  South East South Central   AL
## Alaska    West            Pacific   AK
## Arizona   West           Mountain   AZ
```

- **1b.** Remove the `Region` column from `state.df`. You can do this in (at least) two ways: by using negative indexing, or by directly setting `state.df$Region` to be `NULL`. Display the first 3 rows and all 10 columns of `state.df`.

```
state.df$Region <- NULL
head(state.df, 3)
```

```
##         Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## Alabama       3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska         365   6315        1.5    69.31   11.3    66.7   152 566432
## Arizona       2212   4530        1.8    70.55    7.8    58.1    15 113417
##                    Division Abbr
## Alabama East South Central   AL
## Alaska             Pacific   AK
```

```
## Arizona           Mountain   AZ
```

- **1c.** Add two columns to `state.df`, containing the x and y coordinates (longitude and latitude, respectively) of the center of the states, that are stored in the (existing) list `state.center`. Hint: take a look at this list in the console, to see what its elements are named. Name these two columns `Center.x` and `Center.y`. Display the first 3 rows and all 12 columns of `state.df`.

```
state.df$Center.x <- state.center$x
state.df$Center.y <- state.center$y
head(state.df, 3)
```

```
##         Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## Alabama       3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska         365   6315        1.5    69.31   11.3    66.7   152 566432
## Arizona       2212   4530        1.8    70.55    7.8    58.1    15 113417
##                   Division Abbr  Center.x Center.y
## Alabama East South Central   AL  -86.7509  32.5901
## Alaska              Pacific   AK -127.2500  49.2500
## Arizona           Mountain   AZ -111.6250  34.2192
```

- **1d.** Make a new data frame which contains only those states whose longitude is less than -100. Do this in two different ways: using manual indexing, and `subset`(). Check that they are equal to each other, using an appropriate function call.

```
df1 <- state.df[state.df$Center.x < 100,]
df2 <- subset(state.df, Center.x < 100)
all.equal(df1, df2)
```

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

- **1e.** Make a new data frame which contains only the states whose longitude is less than -100, and whose murder rate is above 9%. Print this new data frame to the console. Among the states in this new data frame, which has the highest average life expectancy?

```
df1 <- state.df[state.df$Center.x < -100 & state.df$Murder > 9,]
df1
```

```
##            Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
## California      21198   5114        1.1    71.71   10.3    62.6    20 156361
## Nevada            590   5149        0.5    69.03   11.5    65.2   188 109889
## New Mexico       1144   3601        2.2    70.32    9.7    55.2   120 121412
##            Division Abbr Center.x Center.y
## Alaska      Pacific   AK -127.250  49.2500
## California  Pacific   CA -119.773  36.5341
## Nevada     Mountain   NV -116.851  39.1063
## New Mexico Mountain   NM -105.942  34.4764
```

Among these states, California has the highest life expectancy.

Prostate cancer data set

Below we read in the prostate cancer data set that we looked in the last lab. You can remind yourself about what's been measured by looking back at the lab.

```
pros.dat =
  read.table("http://www.stat.cmu.edu/~ryantibs/statcomp/data/pros.dat")
```

Q2. Practice with the apply family

- **2a.** Using `sapply()`, calculate the mean of each variable. Also, calculate the standard deviation of each variable. Each should require just one line of code. Display your results.

```
sapply(pros.dat, FUN = mean)
```

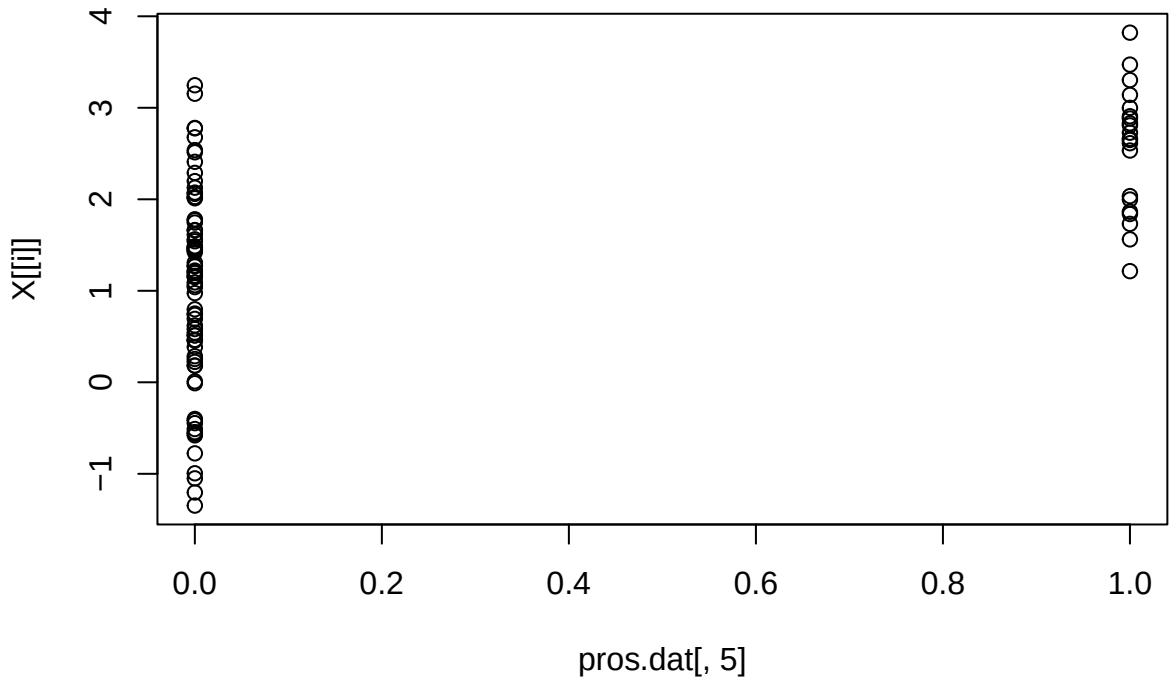
```
##     lcavol    lweight        age       lbph        svi        lcp    gleason
##  1.3500096  3.6289427 63.8659794  0.1003556  0.2164948 -0.1793656  6.7525773
##      pgg45       lpsa
## 24.3814433  2.4783869
```

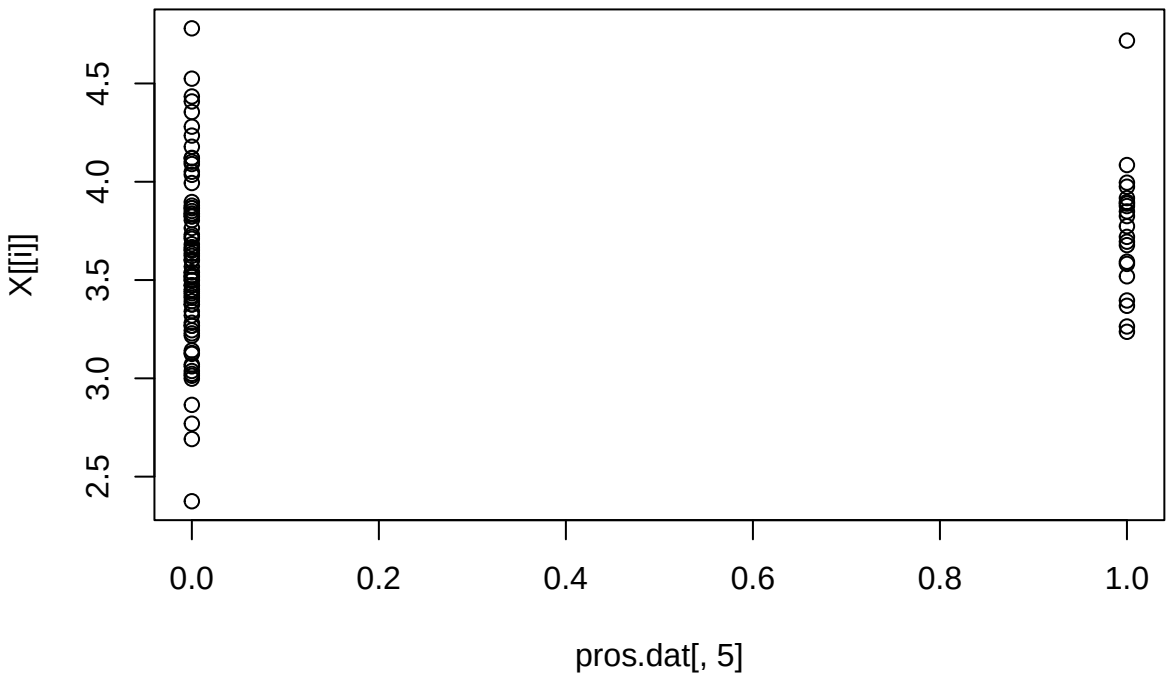
```
sapply(pros.dat, FUN = sd)
```

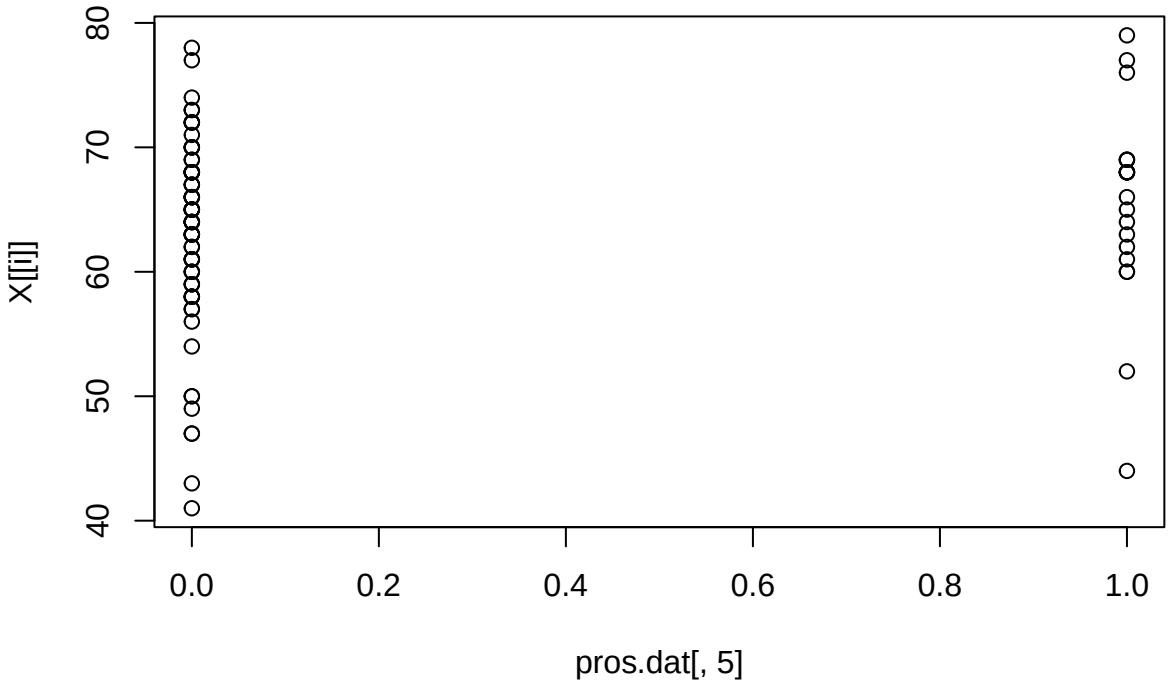
```
##     lcavol    lweight        age       lbph        svi        lcp    gleason
##  1.1786249  0.4284112  7.4451171  1.4508066  0.4139949  1.3982496  0.7221341
##      pgg45       lpsa
## 28.2040346  1.1543291
```

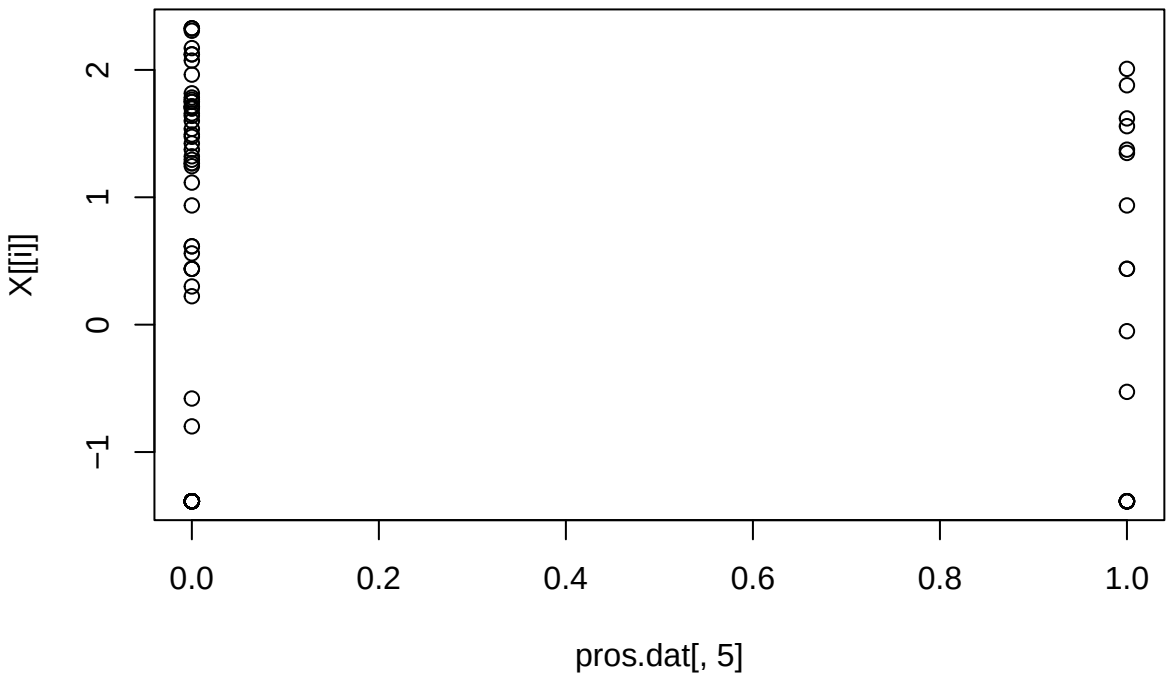
- **2b.** Let's plot each variable against SVI. Using `lapply()`, plot each column, excluding SVI, on the y-axis with SVI on the x-axis. This should require just one line of code. **Challenge**: label the y-axes in your plots appropriately. Your solution should still consist of just one line of code and use an apply function. Hint: for this part, consider using `mapply()`.

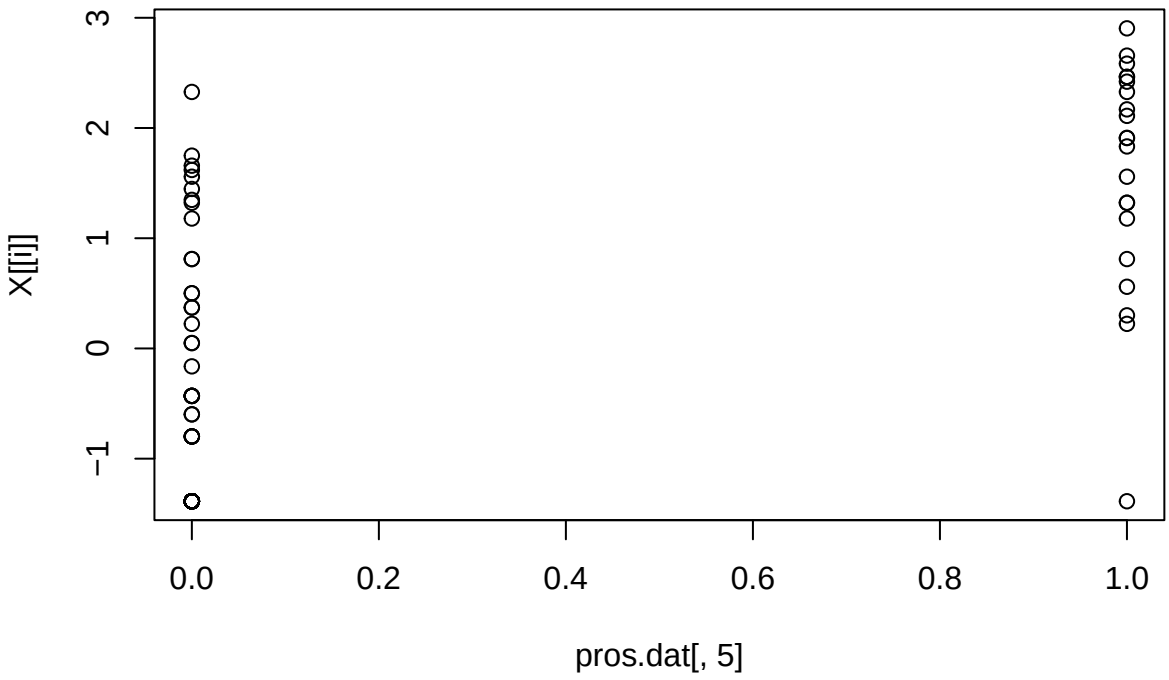
```
lapply(pros.dat[,-5], FUN = plot, x = pros.dat[,5])
```

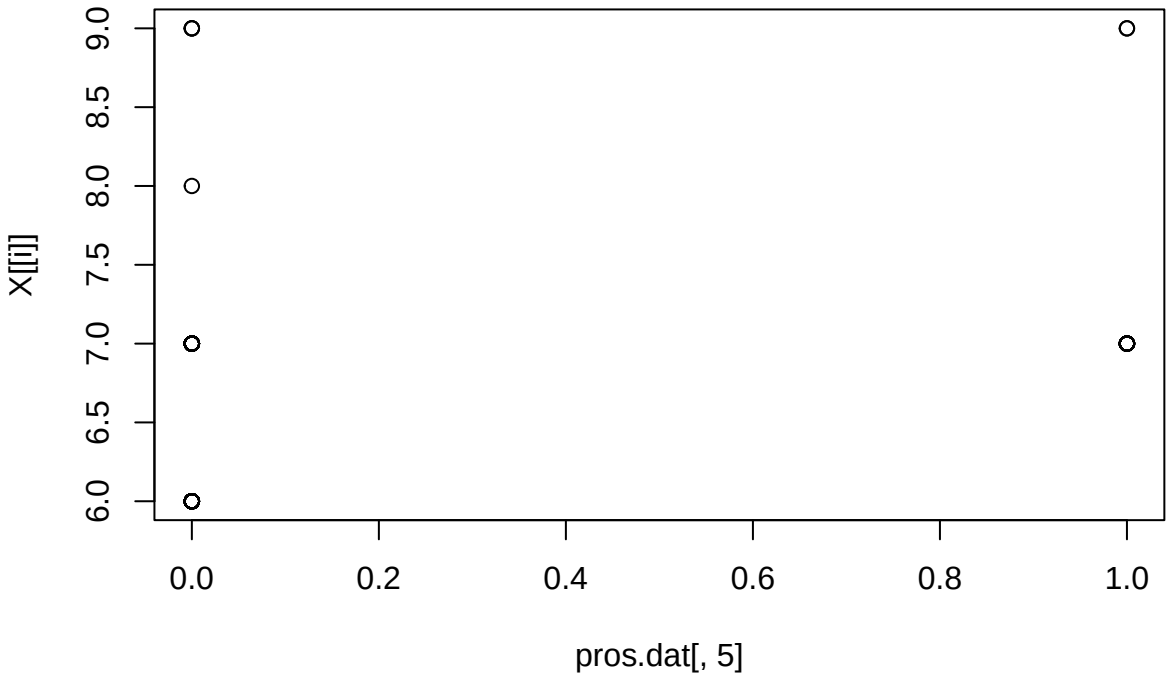


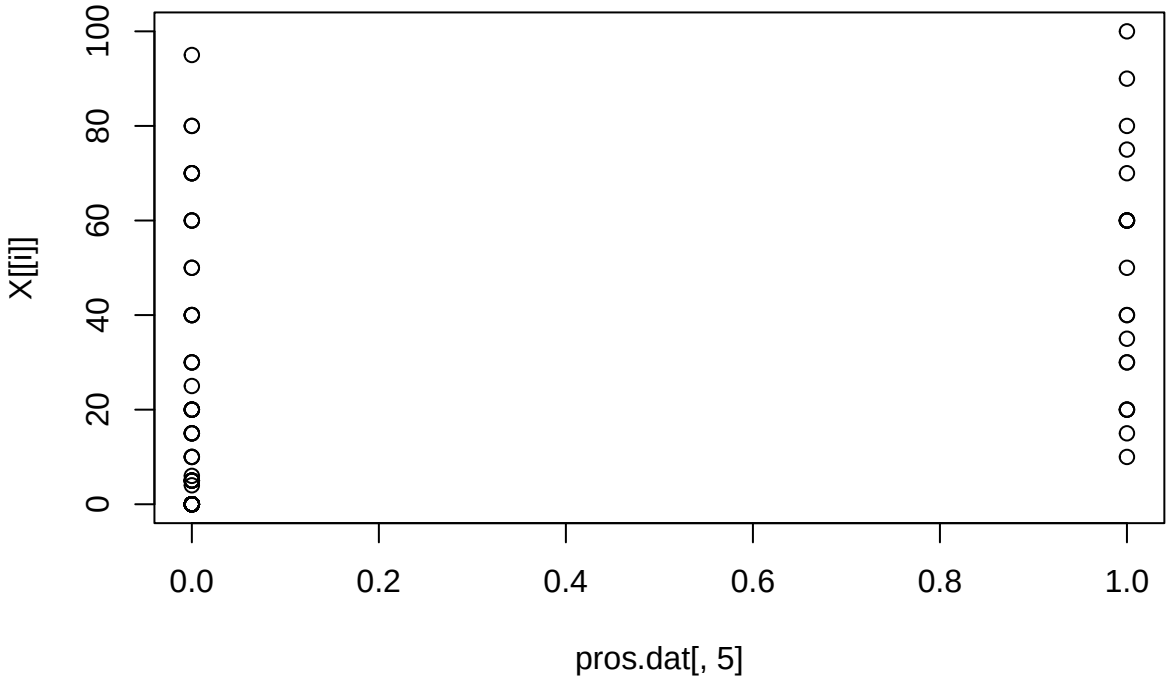

X[[i]]
2.5
3.0
3.5
4.0
4.5
0.0
0.2
0.4
0.6
0.8
1.0
pros.dat[, 5]

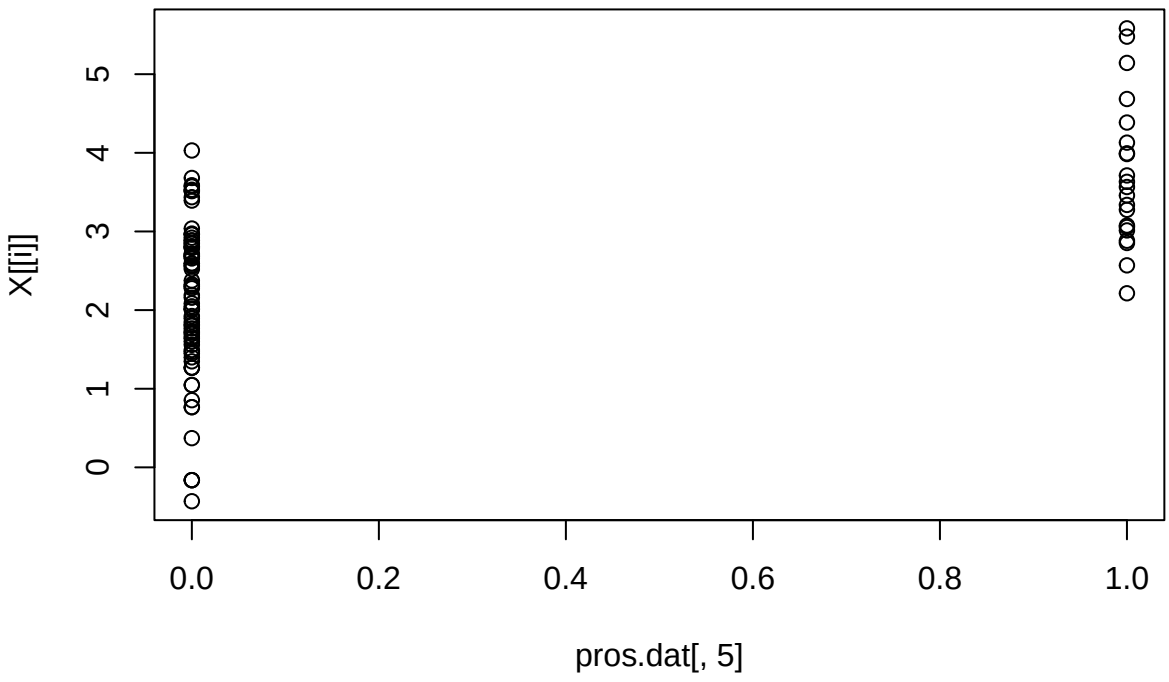

X[[i]]
40
50
60
70
80
0.0
0.2
0.4
0.6
0.8
1.0
pros.dat[, 5]


X[[i]]
2
1
0
−1
0.0
0.2
0.4
0.6
0.8
1.0
pros.dat[, 5]


X[[i]]
3
2
1
0
−1
0.0
0.2
0.4
0.6
0.8
1.0
pros.dat[, 5]


X[[i]]
6.0
6.5
7.0
7.5
8.0
8.5
9.0
0.0
0.2
0.4
0.6
0.8
1.0
pros.dat[, 5]


X[[i]]
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
pros.dat[, 5]




```
## $lcavol
## NULL
##
## $lweight
## NULL
##
## $age
## NULL
##
## $lbph
## NULL
##
## $lcp
## NULL
##
## $gleason
## NULL
##
## $pgg45
## NULL
##
## $lpsa
## NULL
```

- **2c.** Now, use `lapply()` to perform t-tests for each variable in the data set, between SVI and non-SVI groups. To be precise, you will perform a t-test for each variable excluding the SVI variable itself. For convenience, we've defined a function `t.test.by.ind()` below, which takes a numeric variable `x`,

and then an indicator variable `ind` (of 0s and 1s) that defines the groups. Run this function on the columns of `pros.dat`, excluding the SVI column itself, and save the result as `tests`. What kind of data structure is `tests`? Print it to the console.

```
t.test.by.ind = function(x, ind) {
  stopifnot(all(ind %in% c(0, 1)))
  return(t.test(x[ind == 0], x[ind == 1]))
}
```

```
tests <- lapply(pros.dat[,-5], FUN = t.test.by.ind, ind = pros.dat[,5])
typeof(tests)
```

```
## [1] "list"
```

- **2d.** Using `lapply()` again, extract the p-values from the `tests` object you created in the last question, with just a single line of code. Hint: first, take a look at the first element of `tests`, what kind of object is it, and how is the p-value stored? Second, run the command `` `[[`(pros.dat, "lcavol") `` in your console—what does this do? Now use what you've learned to extract p-values from the `tests` object.

```
lapply(tests, FUN = '[[', 'p.value')
```

```
## $lcavol
## [1] 1.25104e-10
##
## $lweight
## [1] 0.07472088
##
## $age
## [1] 0.2770533
##
## $lbph
## [1] 0.3834772
##
## $lcp
## [1] 4.579752e-10
##
## $gleason
## [1] 0.0008816293
##
## $pgg45
## [1] 2.482255e-05
##
## $lpsa
## [1] 7.879066e-08
```

Rio Olympics data set

Now we're going to examine data from the 2016 Summer Olympics in Rio de Janeiro, taken from https://github.com/flother/rio2016 (complete data on the 2020 Summer Olympics in Tokyo doesn't appear to be available yet). Below we read in the data and store it as `rio`.

```
rio = read.csv("http://www.stat.cmu.edu/~ryantibs/statcomp/data/rio.csv")
```

Q3. More practice with data frames and apply

- **3a.** What kind of object is rio? What are its dimensions and columns names of `rio`? What does each row represent? Is there any missing data?

```
typeof(rio)
```

```
## [1] "list"
```

```
dim(rio)
```

```
## [1] 11538    12
```

```
colnames(rio)
```

```
##  [1] "id"            "name"          "nationality"   "sex"
##  [5] "date_of_birth" "height"        "weight"        "sport"
##  [9] "gold"          "silver"        "bronze"        "info"
```

```
head(rio)
```

```
##          id           name nationality    sex date_of_birth height weight
## 1 736041664 A Jesus Garcia         ESP   male    1969-10-17   1.72     64
## 2 532037425     A Lam Shin         KOR female    1986-09-23   1.68     56
## 3 435962603    Aaron Brown         CAN   male    1992-05-27   1.98     79
## 4 521041435     Aaron Cook         MDA   male    1991-01-02   1.83     80
## 5  33922579     Aaron Gate         NZL   male    1990-11-26   1.81     71
## 6 173071782    Aaron Royle         AUS   male    1990-01-26   1.80     67
##       sport gold silver bronze info
## 1 athletics    0      0      0
## 2   fencing    0      0      0
## 3 athletics    0      0      1
## 4 taekwondo    0      0      0
## 5   cycling    0      0      0
## 6 triathlon    0      0      0
```

The info variable appears to be missing values.

- **3b.** Use `rio` to answer the following questions. How many athletes competed in the 2016 Summer Olympics? How many countries were represented? What were these countries, and how many athletes competed for each one? Which country brought the most athletes, and how many was this? Hint: for a factor variable `f`, you can use `table(f)` see how many elements in `f` are in each level of the factor.

```
nrow(rio)
```

```
## [1] 11538
```

```
length(table(rio$nationality))
```

```
## [1] 207
```

```
table(rio$nationality)
```

```
##
## AFG ALB ALG AND ANG ANT ARG ARM ARU ASA AUS AUT AZE BAH BAN BAR BDI BEL BEN BER
##   3   6  68   5  26   9 223  32   7   4 431  71  56  30   7  11   9 108   6   8
## BHU BIH BIZ BLR BOL BOT BRA BRN BRU BUL BUR CAF CAM CAN CAY CGO CHA CHI CHN CIV
```

```
##   2  11   3 124  12  12 485  34   3  50   5   6   6 321   5  10   2  42 404  12
## CMR COD COK COL COM CPV CRC CRO CUB CYP CZE DEN DJI DMA DOM ECU EGY ERI ESA ESP
##  24   4   9 154   4   5  11  88 123  16 104 128   7   2  29  38 122  12   8 313
## EST ETH FIJ FIN FRA FSM GAB GAM GBR GBS GEO GEQ GER GHA GRE GRN GUA GUI GUM GUY
##  46  38  54  54 410   5   6   4 374   5  40   2 441  16  93   7  21   5   5   6
## HAI HKG HON HUN INA IND IOA IRI IRL IRQ ISL ISR ISV ITA IVB JAM JOR JPN KAZ KEN
##  10  38  30 154  28 123   9  64  80  26   8  47   7 312   4  57   8 346 103  80
## KGZ KIR KOR KOS KSA LAO LAT LBA LBR LCA LES LIB LIE LTU LUX MAD MAR MAS MAW MDA
##  19   3 213   8  11   6  32   7   2   5   8   9   3  67  10   6  49  32   5  23
## MDV MEX MGL MHL MKD MLI MLT MNE MON MOZ MRI MTN MYA NAM NCA NED NEP NGR NIG NOR
##   4 126  43   5   6   6   7  35   3   6  11   2   7  10   5 249   7  78   6  62
## NRU NZL OMA PAK PAN PAR PER PHI PLE PLW PNG POL POR PRK PUR QAT ROT ROU RSA RUS
##   2 208   4   7  10  11  29  13   6   5   8 242  95  31  40  39  10  98 146 286
## RWA SAM SEN SEY SIN SKN SLE SLO SMR SOL SOM SRB SRI SSD STP SUD SUI SUR SVK SWE
##   7   8  22  10  25   7   4  63   5   3   2 103   9   3   3   6 104   6  51 164
## SWZ SYR TAN TGA THA TJK TKM TLS TOG TPE TTO TUN TUR TUV UAE UGA UKR URU USA UZB
##   2   7   7   7  54   7   9   3   5  56  32  61 103   1  13  21 205  17 567  70
## VAN VEN VIE VIN YEM ZAM ZIM
##   4  88  23   4   3   7  35
```

```
table(rio$nationality)[table(rio$nationality) == max(table(rio$nationality))]
```

```
## USA
## 567
```

11538 athletes competed in the 2016 Summer Olympics. 207 countries were represented. The USA brought the most athletes with 567.

- **3c.** How many medals of each type—gold, silver, bronze—were awarded at this Olympics? Are they equal? Is this result surprising, and can you explain what you are seeing?

```
colSums(rio[,c("gold", "silver", "bronze")])
```

```
##   gold silver bronze
##    666    655    704
```

The amount of each medal awarded is slightly uneven. This could be the case if multiple medals are awarded for ties.

- **3d.** Create a column called `total` which adds the number of gold, silver, and bronze medals for each athlete, and add this column to `rio`. Which athlete had the most number of medals and how many was this? Gold medals? Silver medals? In the case of ties, here, display all the relevant athletes.

```
rio$total <- rio$gold + rio$silver + rio$bronze
rio[rio$total == max(rio$total),]
```

```
##             id           name nationality  sex date_of_birth height weight
## 7402 491565031 Michael Phelps         USA male    1985-06-30   1.94     90
##         sport gold silver bronze
## 7402 aquatics    5      1      0
##
## 7402 The USA's Michael Phelps has claimed 22 Olympic medals from three editions, 18 of which were gol
##      total
## 7402     6
```

```
rio[rio$gold == max(rio$gold),]
```

```
##             id           name nationality  sex date_of_birth height weight
## 7402 491565031 Michael Phelps         USA male    1985-06-30   1.94     90
```

```
##         sport gold silver bronze
## 7402 aquatics    5      1      0
##
## 7402 The USA's Michael Phelps has claimed 22 Olympic medals from three editions, 18 of which were gol
##      total
## 7402     6
```

```
rio[rio$silver == max(rio$silver),]
```

```
##              id                         name nationality    sex date_of_birth
## 419    71010173          Alexandra Raisman         USA female    1994-05-25
## 1880   51787706   Chad Guy Bertrand le Clos         RSA   male    1992-04-12
## 2301  527822094               Danell Leyva         USA   male    1991-10-30
## 2544  634903913             Denis Abliazin         RUS   male    1992-08-03
## 2749  327966166               Duncan Scott         GBR   male    1997-05-06
## 3049  661638106                Emma McKeon         AUS female    1994-05-24
## 3441   28413973           Florent Manaudou         FRA   male    1990-11-12
## 3512  688191947            Franziska Weber         GER female    1989-05-24
## 4419  121190622 Isaquias Queiroz dos Santos         BRA   male    1994-01-03
## 4578  924475457                  James Guy         GBR   male    1995-11-26
## 4713  217440009                Jazz Carlin         GBR female    1990-09-17
## 6524  341947091            Madeline Groves         AUS female    1995-05-25
## 6828   80802864               Maria Paseka         RUS female    1995-07-19
## 8922  360632507              Rebecca James         GBR female    1991-11-29
## 9534  773163998               Sarah Hammer         USA female    1983-08-18
## 9903  973414226              Simone Manuel         USA female    1996-08-02
## 10297 701625147          Taoufik Makhloufi         ALG   male    1988-04-29
## 10523 128638379                Tina Dietze         GER female    1988-01-25
## 10986 526167499                 Wenyan Sun         CHN female    1989-12-27
## 11102 773136288              Xuechen Huang         CHN female    1990-02-25
## 11371  86099624              Yulia Efimova         RUS female    1992-04-03
##       height weight      sport gold silver bronze
## 419     1.58     52 gymnastics    1      2      0
## 1880    1.90     83   aquatics    0      2      0
## 2301    1.73     72 gymnastics    0      2      0
## 2544    1.60     62 gymnastics    0      2      1
## 2749    1.91     74   aquatics    0      2      0
## 3049    1.80     60   aquatics    1      2      1
## 3441    1.99     99   aquatics    0      2      0
## 3512    1.76     70      canoe    0      2      0
## 4419    1.75     85      canoe    0      2      1
## 4578    1.88     84   aquatics    0      2      0
## 4713    1.76     62   aquatics    0      2      0
## 6524    1.79     66   aquatics    0      2      0
## 6828    1.61     48 gymnastics    0      2      0
## 8922    1.71     66    cycling    0      2      0
## 9534    1.71     65    cycling    0      2      0
## 9903    1.78     72   aquatics    2      2      0
## 10297   1.70     67  athletics    0      2      0
## 10523   1.72     68      canoe    0      2      0
## 10986   1.70     58   aquatics    0      2      0
## 11102   1.75     62   aquatics    0      2      0
## 11371     NA     NA   aquatics    0      2      0
##
## 419
```

```
## 1880  Chad le Clos dream came true at London 2012, when he beat Michael Phelps to win gold in the 200
## 2301
## 2544
## 2749
## 3049
## 3441
## 3512
## 4419
## 4578
## 4713
## 6524
## 6828
## 8922
## 9534
## 9903
## 10297
## 10523
## 10986
## 11102
## 11371
##       total
## 419       3
## 1880      2
## 2301      2
## 2544      3
## 2749      2
## 3049      4
## 3441      2
## 3512      2
## 4419      3
## 4578      2
## 4713      2
## 6524      2
## 6828      2
## 8922      2
## 9534      2
## 9903      4
## 10297     2
## 10523     2
## 10986     2
## 11102     2
## 11371     2
```

Michael Phelps earned the most medals with 6. Phelps also had the most gold medals with 5. 21 different athletes earned the maximum of 2 silver medals.

- **3e.** Using `tapply()`, calculate the total medal count for each country. Save the result as `total.by.nat`, and print it to the console. Which country had the most number of medals, and how many was this? How many countries had zero medals?

```
total.by.nat <- tapply(rio$total, INDEX = rio$nationality, FUN = sum)
total.by.nat[total.by.nat == max(total.by.nat)]
```

```
## USA
## 264
```

```
length(total.by.nat[total.by.nat == 0])
```

```
## [1] 120
```

The USA had the most medals with 264. 120 different countries failed to earn a medal.

- **3f.** Among the countries that had zero medals, which had the most athletes, and how many athletes was this? (Ouch!)

```
zeros <- rownames(total.by.nat[total.by.nat == 0])
rio.zero <- rio[rio$nationality %in% zeros,]
rio.zero$dummy <- 1
rio.zero <- tapply(rio.zero$dummy, rio.zero$nationality, FUN = sum)
rio.zero[rio.zero == max(rio.zero)]
```

```
## CHI
##  42
```

Of the countries with zero medals, Chile had the most athletes with 42.

Q4. Young and old folks

- **4a.** The variable `date_of_birth` contains strings of the date of birth of each athlete. Use the `substr()` function to extract the year of birth for each athlete, and then create a new numeric variable called `age`, equal to 2016 - (the year of birth). (Here we're ignoring days and months for simplicity.) Hint: to extract the first 4 characters of a string `str`, you can use `substr(str, 1, 4)`. As always, you can also look at the help file for `substr()` for more details.

  Add the `age` variable to the `rio` data frame. Who is the oldest athlete, and how old is he/she? Youngest athlete, and how old is he/she? In the case of ties, here, display all the relevant athletes.

```
rio$age <- 2016 - as.numeric(substr(rio$date_of_birth, 1, 4))
rio[rio$age == max(rio$age),]
```

```
##             id           name nationality    sex date_of_birth height weight
## 5300 271404469 Julie Brougham         NZL female    1954-05-20   1.57     48
## 7093 590552399     Mary Hanna         AUS female    1954-12-01   1.73     63
##           sport gold silver bronze info total age
## 5300 equestrian    0      0      0          0  62
## 7093 equestrian    0      0      0          0  62
```

```
rio[rio$age == min(rio$age),]
```

```
##             id                 name nationality    sex date_of_birth height
## 655   209671126     Ana Iulia Dascal         ROU female    2002-09-12   1.83
## 2432  380938305      Darya Semyonova         TKM female    2002-05-28   1.70
## 3306   91359398    Fatima Alkaramova         AZE female    2002-06-26   1.75
## 3599   32924852        Gaurika Singh         NEP female    2002-11-26   1.55
## 5577   55365531    Kaya Adwoa Forson         GHA female    2002-03-19     NA
## 9919  112175885 Siri Arun Budcharern         LAO female    2002-01-12   1.66
## 10434 326914230          Thint Myaat         MYA   male    2002-04-14   1.60
## 11149 188592965            Yanhan Ai         CHN female    2002-02-07   1.68
##       weight    sport gold silver bronze info total age
## 655       60 aquatics    0      0      0          0  14
## 2432      50 aquatics    0      0      0          0  14
## 3306      60 aquatics    0      0      0          0  14
## 3599      45 aquatics    0      0      0          0  14
```

```
## 5577      NA aquatics    0      0      0          0  14
## 9919      63 aquatics    0      0      0          0  14
## 10434     52 aquatics    0      0      0          0  14
## 11149     54 aquatics    0      0      0          0  14
```

There were 2 62 year olds and 8 14 year olds.

- **4b.** Answer the same questions as in the last part, but now only among athletes who won a medal.

```
rio.medaled <- rio[rio$total > 0,]
rio.medaled[rio.medaled$age == max(rio.medaled$age),]
```

```
##             id         name nationality  sex date_of_birth height weight
## 8019 764396400 Nick Skelton         GBR male    1957-12-30   1.75     76
##           sport gold silver bronze info total age
## 8019 equestrian    1      0      0          1  59
```

```
rio.medaled[rio.medaled$age == min(rio.medaled$age),]
```

```
##             id     name nationality    sex date_of_birth height weight    sport
## 8774 889734754 Qian Ren         CHN female    2001-02-20   1.62     49 aquatics
##      gold silver bronze info total age
## 8774    1      0      0          1  15
```

One 59 year old medaled and one 15 year old medaled.

- **4c.** Using a single call to `tapply()`, answer: how old are the youngest and oldest athletes, for each sport?

```
minmax <- function(x){c(min = min(x), max = max(x))}
tapply(rio[,"age"], INDEX = rio[,"sport"], FUN = minmax)
```

```
## $aquatics
## min max
##  14  41
##
## $archery
## min max
##  17  44
##
## $athletics
## min max
##  16  47
##
## $badminton
## min max
##  19  40
##
## $basketball
## min max
##  20  39
##
## $boxing
## min max
##  18  37
##
## $canoe
## min max
```

```
##  18  49
##
## $cycling
## min max
##  18  45
##
## $equestrian
## min max
##  18  62
##
## $fencing
## min max
##  16  42
##
## $football
## min max
##  16  41
##
## $golf
## min max
##  18  47
##
## $gymnastics
## min max
##  16  41
##
## $handball
## min max
##  19  44
##
## $hockey
## min max
##  18  37
##
## $judo
## min max
##  19  39
##
## $`modern pentathlon`
## min max
##  17  37
##
## $rowing
## min max
##  18  57
##
## $`rugby sevens`
## min max
##  18  36
##
## $sailing
## min max
##  17  55
##
```

```
## $shooting
## min max
##  16  56
##
## $`table tennis`
## min max
##  15  54
##
## $taekwondo
## min max
##  17  38
##
## $tennis
## min max
##  19  44
##
## $triathlon
## min max
##  19  38
##
## $volleyball
## min max
##  18  41
##
## $weightlifting
## min max
##  16  41
##
## $wrestling
## min max
##  18  38
```

- **4d.** You should see that your output from `tapply()` in the last part is a list, which is not particularly convenient. Convert this list into a matrix that has one row for each sport, and two columns that display the ages of the youngest and oldest athletes in that sport. The first 3 rows should look like this:

```
                Youngest Oldest
athletics             14     41
archery               17     44
athletics             16     47
```

You'll notice that we set the row names according to the sports, and we also set appropriate column names. Hint: `unlist()` will unravel all the values in a list; and `matrix()`, as you've seen before, can be used to create a matrix from a vector of values. After you've converted the results to a matrix, print it to the console (and make sure its first 3 rows match those displayed above).

```
my_list <- unlist(tapply(rio[,"age"], INDEX = rio[,"sport"], FUN = minmax))
my_matrix <- matrix(my_list, ncol = 2, byrow = TRUE)
head(my_matrix,3)
```

```
##      [,1] [,2]
## [1,]   14   41
## [2,]   17   44
## [3,]   16   47
```

Q5. Sport by sport

- **5a.** Create a new data frame called `sports`, which we'll populate with information about each sporting event at the Summer Olympics. Initially, define `sports` to contain a single variable called `sport` which contains the names of the sporting events in alphabetical order. Then, add a column called `n_participants` which contains the number of participants in each sport. Use one of the apply functions to determine the number of gold medals given out for each sport, and add this as a column called `n_gold`. Using your newly created `sports` data frame, calculate the ratio of the number of gold medals to participants for each sport. Which sport has the highest ratio? Which has the lowest?

```
rio$dummy <- 1
sports <- data.frame(sport = rownames(table(rio$sport)))
sports$n_participants <- tapply(rio[,"dummy"], rio[,"sport"], FUN = sum)
sports$n_gold <- tapply(rio[,"gold"], rio[,"sport"], FUN = sum)
sports$gold_ratio <- sports$n_gold / sports$n_participants
sports[sports$gold_ratio == max(sports$gold_ratio),]
```

```
##         sport n_participants n_gold gold_ratio
## 13 gymnastics            324     30 0.09259259
```

```
sports[sports$gold_ratio == min(sports$gold_ratio),]
```

```
##    sport n_participants n_gold gold_ratio
## 12  golf            120      2 0.01666667
```

Gymnastics had the highest gold ratio with 9.2%, and golf had the lowest gold ratio with 1.6%.

- **5b.** Use one of the apply functions to compute the average weight of the participants in each sport, and add this as a column to `sports` called `ave_weight`. Important: there are missing weights in the data set coded as `NA`, but your column `ave_weight` should ignore these, i.e., it should be itself free of `NA` values. You will have to pass an additional argument to your apply call in order to achieve this. Hint: look at the help file for the `mean()` function; what argument can you set to ignore `NA` values? Once computed, display the average weights along with corresponding sport names, in decreasing order of average weight.

```
sports$ave_weight <- tapply(rio[,"weight"], rio[,"sport"], FUN = mean, na.rm = TRUE)
sports[with(sports, order(ave_weight, decreasing = TRUE)),]
```

```
##                sport n_participants n_gold gold_ratio ave_weight
## 5         basketball            288     24 0.08333333   87.75000
## 14          handball            363     29 0.07988981   83.71060
## 26        volleyball            384     28 0.07291667   80.10209
## 27     weightlifting            258     15 0.05813953   79.98062
## 18            rowing            547     48 0.08775137   79.93832
## 19      rugby sevens            300     25 0.08333333   78.72391
## 28         wrestling            353     18 0.05099150   77.74212
## 7              canoe            331     27 0.08157100   77.01529
## 16              judo            392     14 0.03571429   76.87632
## 21          shooting            390     15 0.03846154   73.90526
## 24            tennis            196      8 0.04081633   73.16230
## 1            aquatics           1445    120 0.08304498   72.30164
## 2             archery            128      8 0.06250000   72.19048
## 12               golf            120      2 0.01666667   71.44348
## 20            sailing            380     15 0.03947368   71.16935
## 10            fencing            246     21 0.08536585   70.66122
## 15             hockey            432     34 0.07870370   68.90046
## 4           badminton            172      8 0.04651163   68.77439
```

```
## 11          football            611     36 0.05891980   68.43396
## 23         taekwondo            128      8 0.06250000   68.08800
## 8            cycling            525     27 0.05142857   67.82072
## 3          athletics           2363     66 0.02793060   67.71773
## 9          equestrian            222     15 0.06756757   67.49302
## 17 modern pentathlon             72      2 0.02777778   65.95833
## 22      table tennis            172      8 0.04651163   65.17751
## 25         triathlon            110      2 0.01818182   60.63303
## 13        gymnastics            324     30 0.09259259   54.27900
## 6             boxing            286     13 0.04545455         NaN
```

- **5c.** As in the last part, compute the average weight of athletes in each sport, but now separately for men and women. You should therefore add two new columns, called `ave_weight_men` and `ave_weight_women`, to `sports`. Once computed, display the average weights along with corresponding sports, for men and women, each list sorted in decreasing order of average weight. Are the orderings roughly similar?

```
sports$ave_weight_men <- tapply(rio[rio$sex == "male",][,"weight"],
                                rio[rio$sex == "male",][,"sport"],
                                FUN = mean, na.rm = TRUE)
sports$ave_weight_women <- tapply(rio[rio$sex == "female",][,"weight"],
                                  rio[rio$sex == "female",][,"sport"],
                                  FUN = mean, na.rm = TRUE)
sports[with(sports, order(ave_weight_men, ave_weight_women, decreasing=TRUE)),]
```

```
##                sport n_participants n_gold gold_ratio ave_weight ave_weight_men
## 5         basketball            288     24 0.08333333   87.75000      100.29787
## 14          handball            363     29 0.07988981   83.71060       95.43169
## 19      rugby sevens            300     25 0.08333333   78.72391       90.45033
## 26        volleyball            384     28 0.07291667   80.10209       89.42188
## 27     weightlifting            258     15 0.05813953   79.98062       87.53896
## 18            rowing            547     48 0.08775137   79.93832       86.50462
## 28         wrestling            353     18 0.05099150   77.74212       85.37288
## 16              judo            392     14 0.03571429   76.87632       84.61674
## 1           aquatics           1445    120 0.08304498   72.30164       82.21906
## 7              canoe            331     27 0.08157100   77.01529       82.15000
## 21          shooting            390     15 0.03846154   73.90526       81.06897
## 24            tennis            196      8 0.04081633   73.16230       80.41748
## 2            archery            128      8 0.06250000   72.19048       80.07937
## 12              golf            120      2 0.01666667   71.44348       79.00000
## 10           fencing            246     21 0.08536585   70.66122       78.78512
## 15            hockey            432     34 0.07870370   68.90046       77.37500
## 20           sailing            380     15 0.03947368   71.16935       77.12207
## 4           badminton            172      8 0.04651163   68.77439       76.15663
## 23         taekwondo            128      8 0.06250000   68.08800       74.80952
## 3          athletics           2363     66 0.02793060   67.71773       74.77768
## 11          football            611     36 0.05891980   68.43396       74.45171
## 17 modern pentathlon             72      2 0.02777778   65.95833       73.91667
## 9          equestrian            222     15 0.06756757   67.49302       72.95489
## 8            cycling            525     27 0.05142857   67.82072       72.57605
## 22      table tennis            172      8 0.04651163   65.17751       72.55814
## 25         triathlon            110      2 0.01818182   60.63303       66.81481
## 13        gymnastics            324     30 0.09259259   54.27900       63.25455
## 6             boxing            286     13 0.04545455         NaN            NaN
##    ave_weight_women
## 5          75.37762
```

```
## 14         70.78916
## 19         66.59589
## 26         70.68421
## 27         68.78846
## 18         69.77619
## 28         61.80531
## 16         65.39216
## 1          62.28448
## 7          66.45794
## 21         62.67568
## 24         64.67045
## 2          64.30159
## 12         63.20000
## 10         62.73387
## 15         60.42593
## 20         63.19497
## 4          61.20988
## 23         61.25806
## 3          60.15254
## 11         61.06107
## 17         58.00000
## 9          58.63415
## 8          60.20725
## 22         57.53012
## 25         54.56364
## 13         49.55502
## 6               NaN
```

The decrease in male and female weight is somewhat similar.