

Lab 3: Data Frames and Apply

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)
```

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 <- data.frame(state.df, Abbr = state.abb)
state.df[1:3,]
```

```
##      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
state.df[1:3,]
```

```
##      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?

```
df3 <- state.df[state.df["center.x"]<-100 & state.df["Murder"]>9,]
df3
```

```
##           Population Income Illiteracy Life.Exp Murder HS.Grad Frost
## Alabama         3615   3624         2.1   69.05   15.1   41.3    20
## Alaska           365   6315         1.5   69.31   11.3   66.7   152
## Arkansas         2110   3378         1.9   70.66   10.1   39.9    65
## California       21198   5114         1.1   71.71   10.3   62.6    20
## Florida          8277   4815         1.3   70.66   10.7   52.6    11
## Georgia          4931   4091         2.0   68.54   13.9   40.6    60
## Illinois        11197   5107         0.9   70.14   10.3   52.6   127
## Kentucky         3387   3712         1.6   70.10   10.6   38.5    95
## Louisiana        3806   3545         2.8   68.76   13.2   42.2    12
## Michigan         9111   4751         0.9   70.63   11.1   52.8   125
## Mississippi      2341   3098         2.4   68.09   12.5   41.0    50
## Missouri         4767   4254         0.8   70.69    9.3   48.8   108
## Nevada           590   5149         0.5   69.03   11.5   65.2   188
## New Mexico       1144   3601         2.2   70.32    9.7   55.2   120
## New York        18076   4903         1.4   70.55   10.9   52.7    82
## North Carolina   5441   3875         1.8   69.21   11.1   38.5    80
## South Carolina   2816   3635         2.3   67.96   11.6   37.8    65
## Tennessee        4173   3821         1.7   70.11   11.0   41.8    70
## Texas           12237   4188         2.2   70.90   12.2   47.4    35
## Virginia         4981   4701         1.4   70.08    9.5   47.8    85
```

##	Area	Division	Abbr	Center.x	Center.y
## Alabama	50708	East South Central	AL	-86.7509	32.5901
## Alaska	566432	Pacific	AK	-127.2500	49.2500
## Arkansas	51945	West South Central	AR	-92.2992	34.7336
## California	156361	Pacific	CA	-119.7730	36.5341
## Florida	54090	South Atlantic	FL	-81.6850	27.8744
## Georgia	58073	South Atlantic	GA	-83.3736	32.3329
## Illinois	55748	East North Central	IL	-89.3776	40.0495
## Kentucky	39650	East South Central	KY	-84.7674	37.3915
## Louisiana	44930	West South Central	LA	-92.2724	30.6181
## Michigan	56817	East North Central	MI	-84.6870	43.1361
## Mississippi	47296	East South Central	MS	-89.8065	32.6758
## Missouri	68995	West North Central	MO	-92.5137	38.3347
## Nevada	109889	Mountain	NV	-116.8510	39.1063
## New Mexico	121412	Mountain	NM	-105.9420	34.4764
## New York	47831	Middle Atlantic	NY	-75.1449	43.1361
## North Carolina	48798	South Atlantic	NC	-78.4686	35.4195
## South Carolina	30225	South Atlantic	SC	-80.5056	33.6190
## Tennessee	41328	East South Central	TN	-86.4560	35.6767
## Texas	262134	West South Central	TX	-98.7857	31.3897
## Virginia	39780	South Atlantic	VA	-78.2005	37.5630

Among the state in the new data frame, California has the highest life expectancy.

Prostate cancer data set

Let's return to the prostate cancer data set that we looked in Week 2 (taken from the book *The Elements of Statistical Learning*). Below we read in a data frame of 97 men x 9 variables. You can remind yourself about what's been measured by looking back at the lab/homework (or by visiting the URL linked above in your web browser, clicking on "Data" on the left-hand menu, and clicking "Info" under "Prostate").

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

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, MARGIN=2, FUN=mean)
```

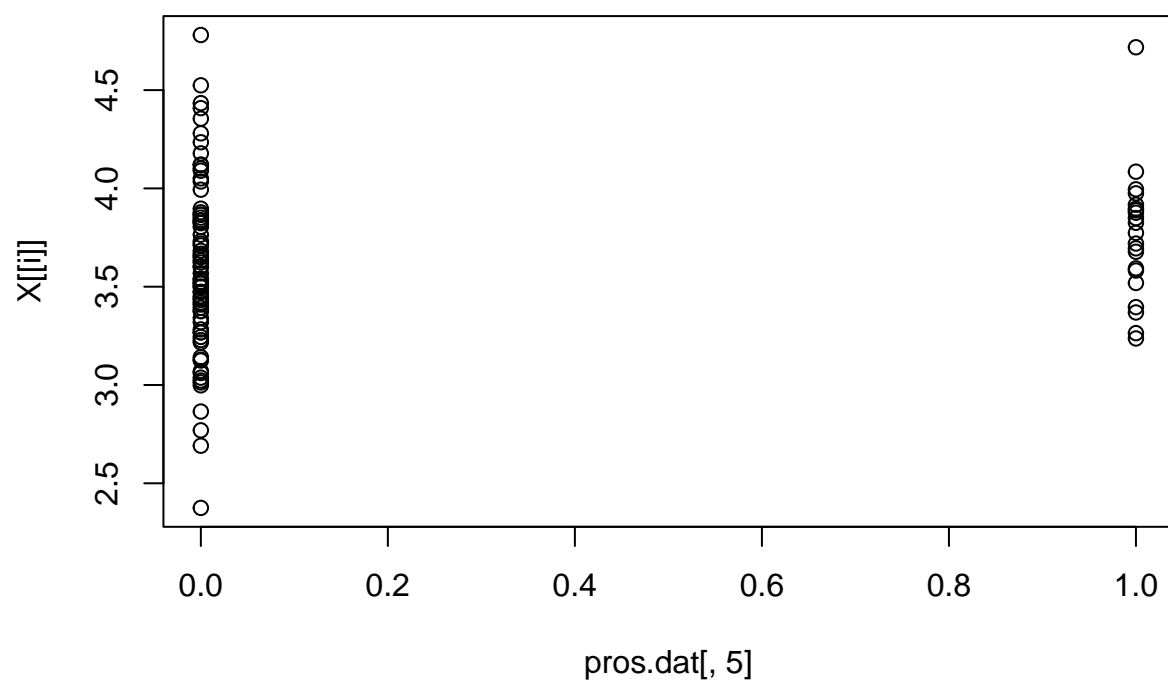
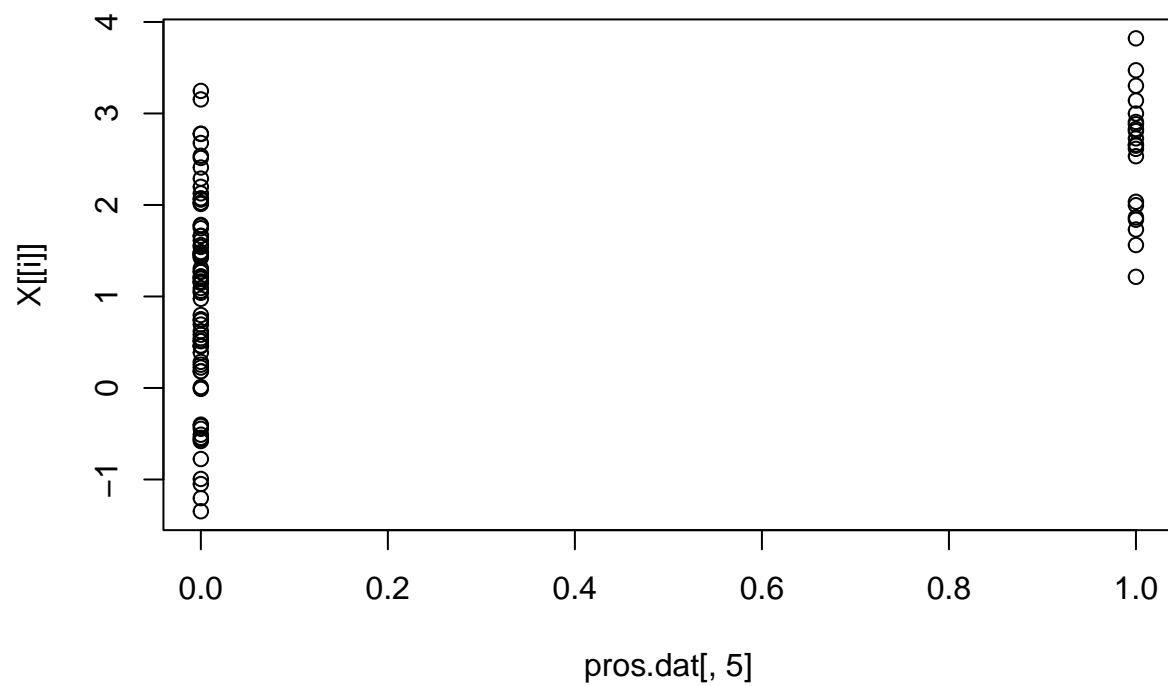
```
##      lcavol      lweight      age      lbph      svi      lcp      gleason
## 1.3500096 3.6289427 63.8659794 0.1003556 0.2164948 -0.1793656 6.7525773
##      pgg45      lpsa
## 24.3814433 2.4783869
```

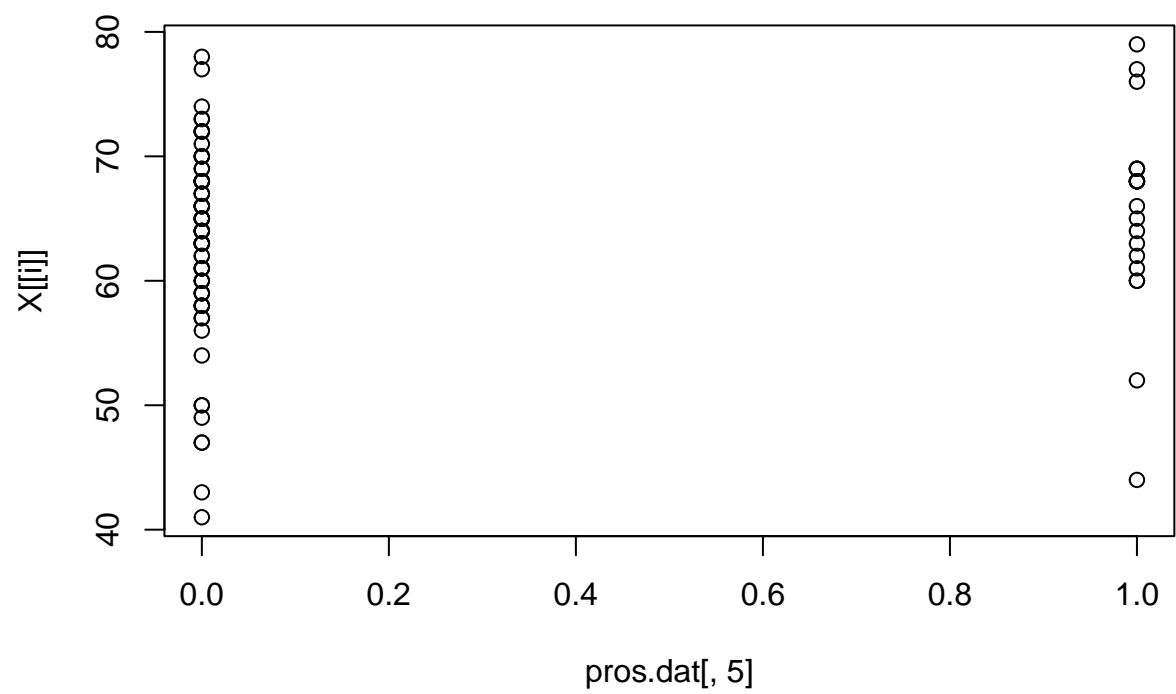
```
sapply(pros.dat, FUN=sd)
```

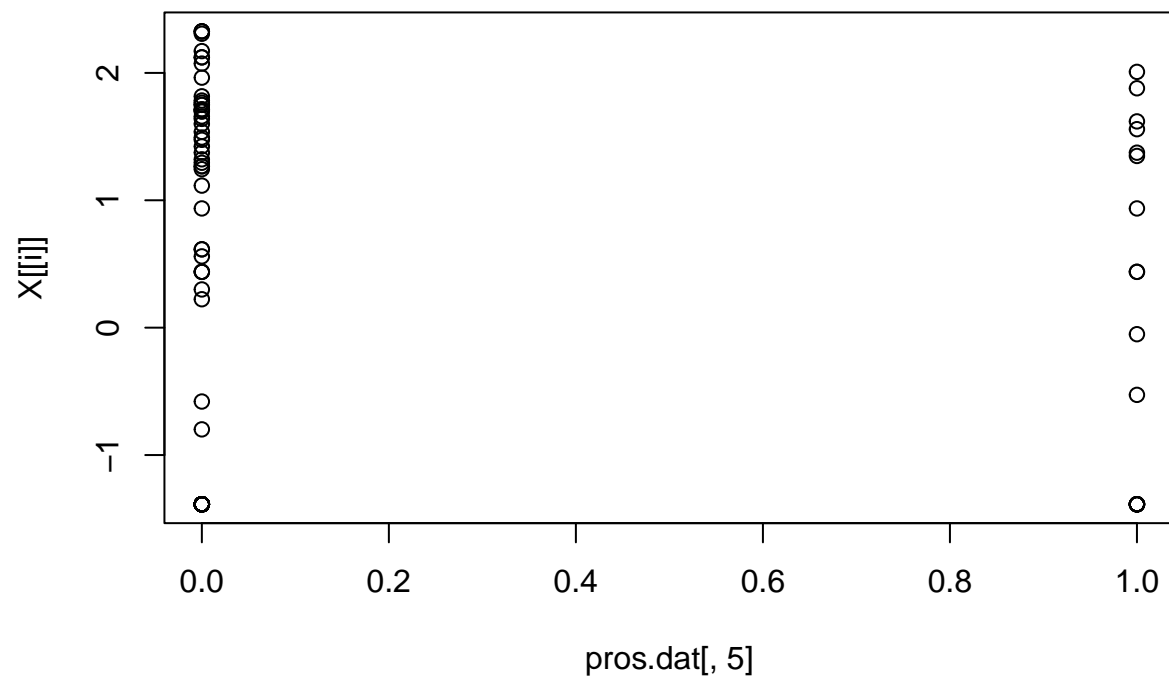
```
##      lcavol      lweight      age      lbph      svi      lcp      gleason
## 1.1786249 0.4284112 7.4451171 1.4508066 0.4139949 1.3982496 0.7221341
##      pgg45      lpsa
## 28.2040346 1.1543291
```

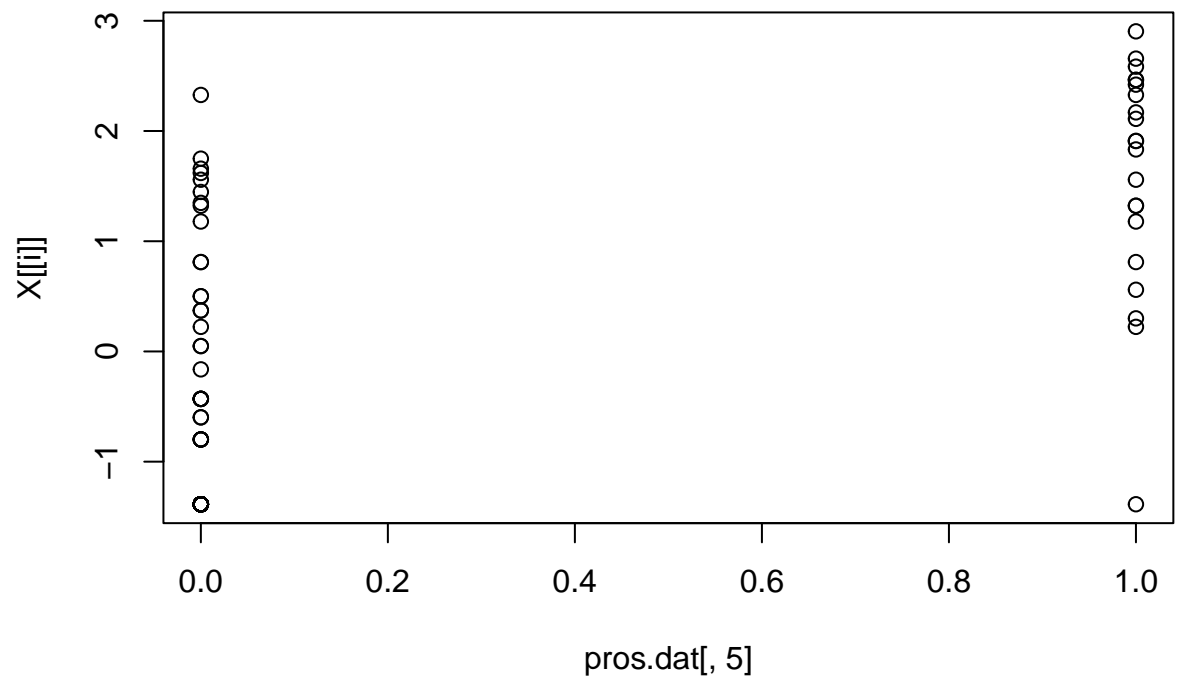
- **2b.** Let's plot each variable against SVI. Using `lapply()`, plot each column, excluding SVI, on the y-axis with SVI on the x-axis. This should require just one line of code.

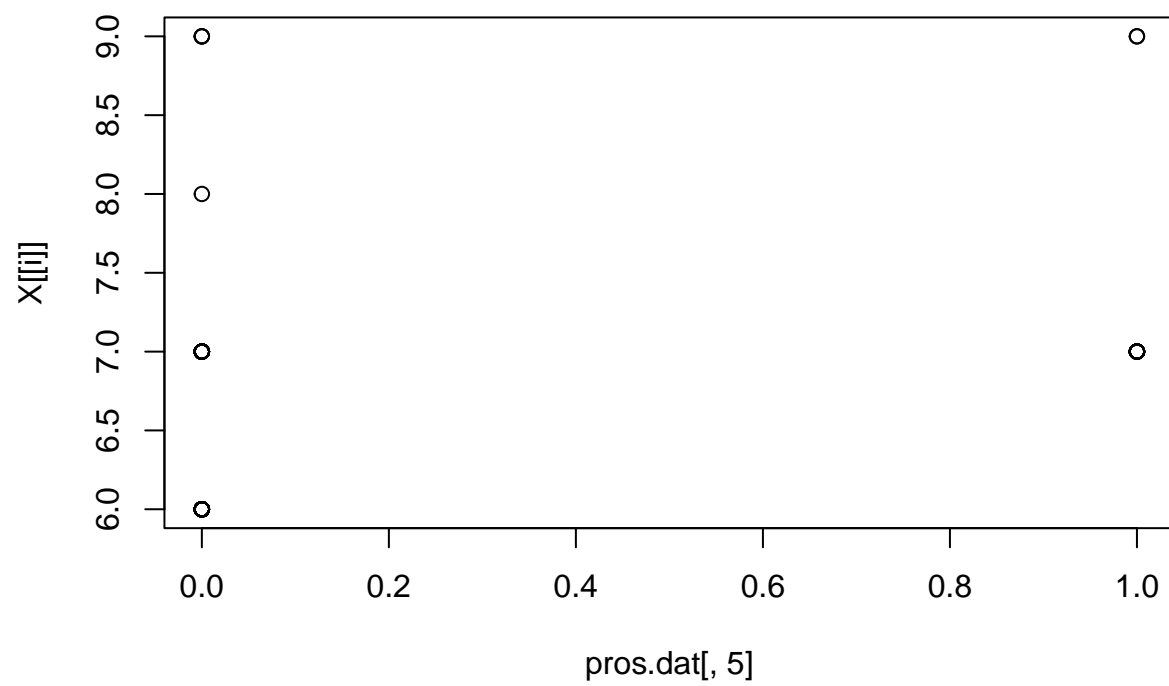
```
lapply(pros.dat[, -5], FUN=plot, x=pros.dat[, 5])
```

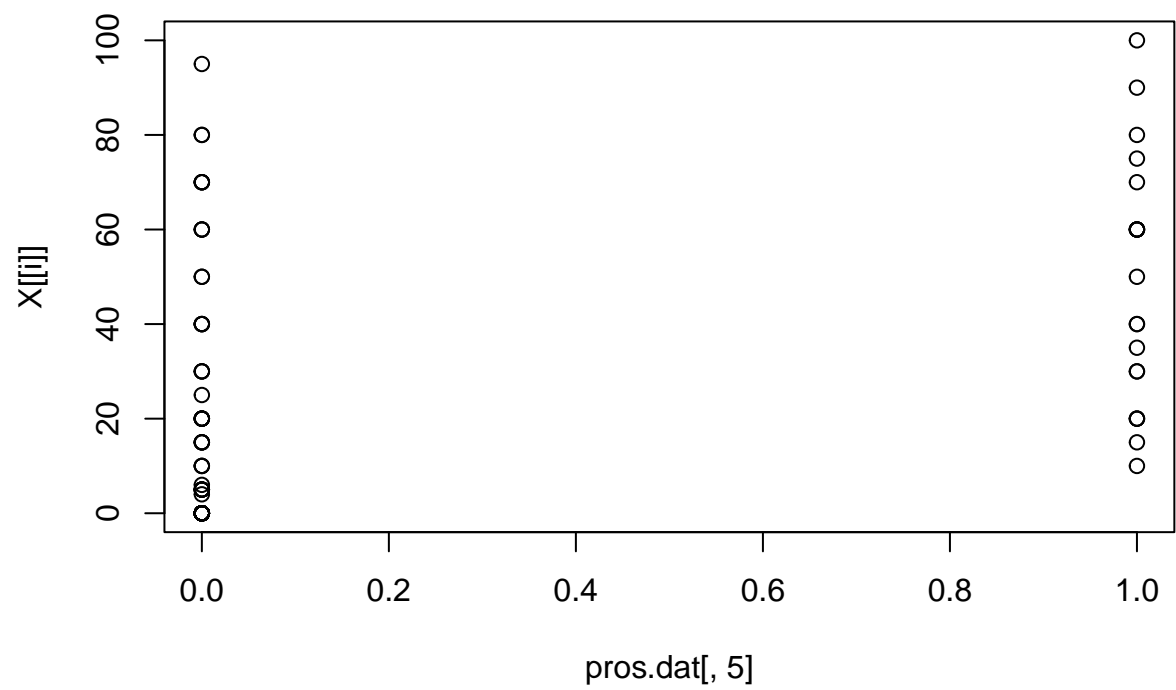


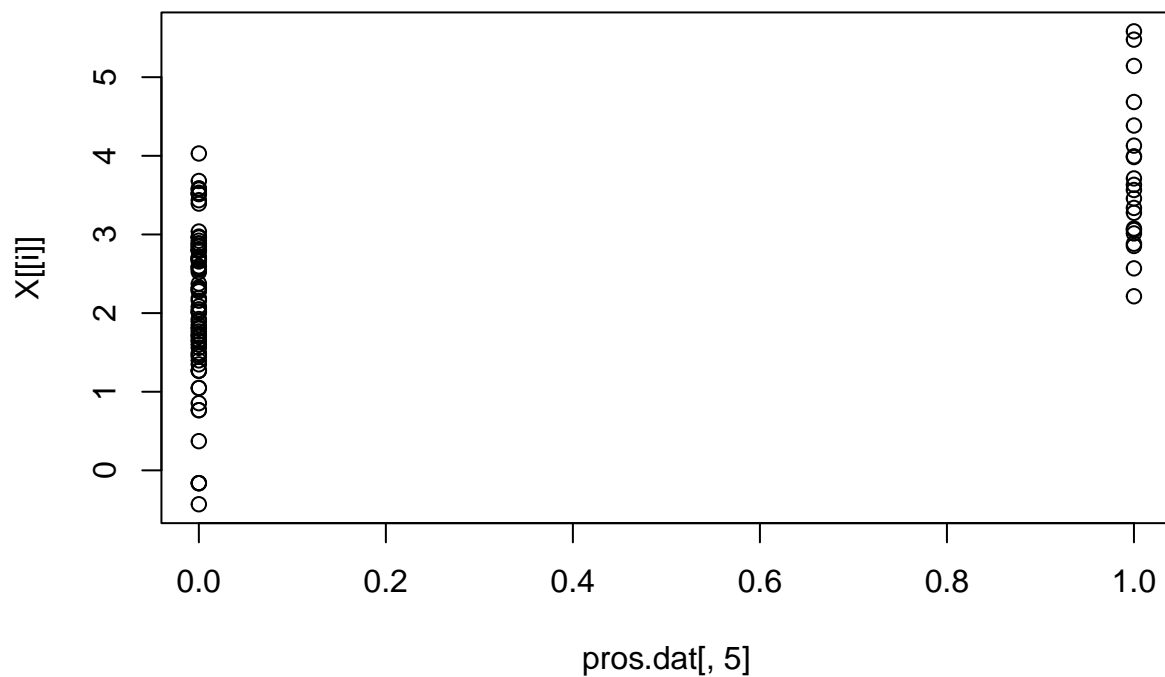












```
## $lcavol
## NULL
##
## $lweight
## NULL
##
## $age
## NULL
##
## $lbph
## NULL
##
## $lcp
## NULL
##
## $gleason
## NULL
##
## $pgg45
## NULL
##
## $lpsa
## NULL
```

- **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], 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.

```
pvals <- sapply(tests, '[', 'p.value')
```

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> (itself put together by scraping the official Summer Olympics website for information about the athletes). Below we read in the data and store it as `rio`.

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

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?

```
class(rio)
```

```
## [1] "data.frame"
```

```
dim(rio)
```

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

```
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 column appears to be missing its 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

max(table(rio$nationality))

## [1] 567

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

## USA
## 567
```

11538 athletes competed and represented 207 different nationalities. The USA had the most athletes, bring 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?

```
sapply(rio[,c("gold", "silver", "bronze")], MARGIN=2, FUN=sum)
```

```
##   gold silver bronze
##   668    657    706
```

The number of medals awarded is slightly unequal. I expect that this happens because of 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"] <- rowSums(rio[,c("gold", "silver", "bronze")])
maxm <- max(rio[, "total"])
maxg <- max(rio[, "gold"])
maxs <- max(rio[, "silver"])
rio[rio["total"]==maxm]
```

```
## [1] "491565031"
## [2] "Michael Phelps"
## [3] "USA"
## [4] "male"
## [5] "1985-06-30"
## [6] "1.94"
## [7] " 90"
## [8] "aquatics"
## [9] "5"
## [10] "1"
## [11] "0"
## [12] "The USA's Michael Phelps has claimed 22 Olympic medals from three editions, 18 of which were g
## [13] "6"
```

```
rio[rio["silver"]==maxs,]
```

```
##           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 Makhoulfi    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 200m freestyle					
## 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 had the most total and gold medals at 6. 21 athletes had the maximum amount 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"], rio[, "nationality"], sum)
total.by.nat
```

```
## AFG ALB ALG AND ANG ANT ARG ARM ARU ASA AUS AUT AZE BAH BAN BAR BDI BEL BEN BER
## 0 0 2 0 0 0 22 4 0 0 82 2 18 6 0 0 1 21 0 0
## BHU BIH BIZ BLR BOL BOT BRA BRN BRU BUL BUR CAF CAM CAN CAY CGO CHA CHI CHN CIV
## 0 0 0 12 0 0 51 2 0 7 0 0 0 69 0 0 0 0 113 2
## CMR COD COK COL COM CPV CRC CRO CUB CYP CZE DEN DJI DMA DOM ECU EGY ERI ESA ESP
## 0 0 0 8 0 0 0 24 11 0 15 41 0 0 1 0 3 0 0 45
## EST ETH FIJ FIN FRA FSM GAB GAM GBR GBS GEO GEQ GER GHA GRE GRN GUA GUI GUM GUY
## 4 8 13 1 95 0 0 0 145 0 7 0 160 0 7 1 0 0 0 0
## HAI HKG HON HUN INA IND IOA IRI IRL IRQ ISL ISR ISV ITA IVB JAM JOR JPN KAZ KEN
## 0 0 0 22 4 2 2 8 3 0 0 2 0 72 0 30 1 65 17 13
## KGZ KIR KOR KOS KSA LAO LAT LBA LBR LCA LES LIB LIE LTU LUX MAD MAR MAS MAW MDA
## 0 0 26 1 0 0 0 0 0 0 0 0 0 7 0 0 1 8 0 1
## MDV MEX MGL MHL MKD MLI MLT MNE MON MOZ MRI MTN MYA NAM NCA NED NEP NGR NIG NOR
## 0 5 2 0 0 0 0 0 0 0 0 0 0 0 0 47 0 18 1 19
## NRU NZL OMA PAK PAN PAR PER PHI PLE PLW PNG POL POR PRK PUR QAT ROT ROU RSA RUS
## 0 36 0 0 0 0 0 1 0 0 0 16 1 7 1 1 0 17 23 115
## RWA SAM SEN SEY SIN SKN SLE SLO SMR SOL SOM SRB SRI SSD STP SUD SUI SUR SVK SWE
## 0 0 0 0 1 0 0 4 0 0 0 53 0 0 0 0 11 0 8 28
## SWZ SYR TAN TGA THA TJK TKM TLS TOG TPE TTO TUN TUR TUV UAE UGA UKR URU USA UZB
## 0 0 0 0 6 1 0 0 0 5 1 3 8 0 1 0 15 0 264 13
## VAN VEN VIE VIN YEM ZAM ZIM
## 0 3 2 0 0 0 0
```

```
total.by.nat[total.by.nat == max(total.by.nat)]
```

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

```
sum(total.by.nat==0)
```

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

The USA had the most medals with 264. 120 countries had zero medals.

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


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

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

China had the most athletes among the countries who didn't earn a medal with 42.

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.

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

Add the `age` variable to the `rio` data frame. variable 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[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      GHG 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 two 62 year olds and eight 14 year olds.

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

```
rio[rio["age"] == max(rio[rio["total"]>0,]["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[rio["age"] == min(rio[rio["total"]>0,]["age"]),]
```

```
##           id          name nationality sex date_of_birth height
## 232  504635619 Ajna Kesely          HUN female   2001-09-10   1.65
## 716  711528333 Anastasiya Tyurina      TJK female   2001-09-27   1.65
## 4642 523195798 Janja Segel           SLO female   2001-06-17   1.77
## 7372 309196317 Mi Rae Kim             PRK female   2001-04-07   1.50
## 7949 354644354 Natsumi Sakai          JPN female   2001-06-19   1.72
## 8774 889734754 Qian Ren                CHN female   2001-02-20   1.62
## 9379 11169898 Sally Yee           FIJ female   2001-04-10    NA
##           weight      sport gold silver bronze info total age
## 232      54    aquatics    0      0      0         0    15
## 716      50    aquatics    0      0      0         0    15
## 4642     60    aquatics    0      0      0         0    15
## 7372     40    aquatics    0      0      0         0    15
## 7949     58    aquatics    0      0      0         0    15
## 8774     49    aquatics    1      0      0         1    15
## 9379     84 table tennis    0      0      0         0    15
```

One 59 year old medaled and seven 15 year olds 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"], rio[, "sport"], 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
aquatics	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).

```
mylist <- unlist(tapply(rio[, "age"], rio[, "sport"], minmax))
mymatrix <- matrix(mylist, ncol=2, byrow=TRUE)
head(mymatrix, 3)
```

```
##      [,1] [,2]
```

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

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 ratio of gold medalists with 9%, and golf had the lowest percent of medalists with about 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"])
sports["ave_weight_women"] <- tapply(rio[rio["sex"]=="female"],[, "weight"], rio[rio["sex"]=="female"],[, "sport"])
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 weights seem to decrease in the same way.