

Lecture 10 problem set

INSERT YOUR NAME HERE

December 14, 2018

Contents

Required reading and instructions	1
Part 1: Modify function from previous problem set that calculated for percent of people in each race/ethnicity group	1
Part 2: Loops	8
Grade: /20	

Required reading and instructions

Required reading

- Grolemund and Wickham 20.4 - 20.5 (chapter 20 is on “Vectors”)
- Grolemund and Wickham 21.1 - 21.3 (chapter 21 is on “iteration”)

General Problem Set instructions

In this homework, you will specify `pdf_document` as the output format. You must have LaTeX installed in order to create pdf documents.

If you have not yet installed MiKTeX/MacTeX, I recommend installing TinyTeX, which is much simpler to install!

- Instructions for installation of TinyTeX can be found [Here](#)
 - General Instructions for Problem Sets [Here](#)
-

Overview of problem set

This problem set will require you to do tasks that apply skills we learned about accessing elements of vectors to modify a user written function. It will also ask you to build some simple loops that give you practice utilizing the three different approaches to looping over an object. Step-by-step instructions are given below.

Part 1: Modify function from previous problem set that calculated for percent of people in each race/ethnicity group

First, you will run a couple chunks of code below, then you will answer specific questions described below.

Load libraries

```
library(tidyverse)
```

Run code below to load zip-code level data from the Census American Community Survey (ACS) and keep selected variables

/0.5

```
#options(tibble.print_min=90)
#options(tibble.print_min=10)

zip_data <- as.tibble(read.csv('https://github.com/ozanj/rclass/raw/master/data/acs/zip_to_state.csv',
  filter(!(state_code %in% c("PR"))) %>%
  arrange(zip_code) %>%
  select(state_code,zip_code,pop_total, pop_white, pop_black, pop_amerindian, pop_asian, pop_nativehawa
  rename(pop_nativeamer = pop_amerindian, pop_latinx = pop_hispanic)
#> Warning: `as.tibble()` is deprecated, use `as_tibble()` (but mind the new semantics).
#> This warning is displayed once per session.

names(zip_data)
#> [1] "state_code"      "zip_code"        "pop_total"
#> [4] "pop_white"       "pop_black"       "pop_nativeamer"
#> [7] "pop_asian"       "pop_nativehawaii" "pop_otherrace"
#> [10] "pop_tworaces"    "pop_latinx"
class(zip_data)
#> [1] "tbl_df"         "tbl"             "data.frame"
```

Run code below [answer to question from previous problem set] which creates percent race/ethnicity variables outside of a function for two race/ethnicity categories

/0.5

```
#white
zip_data$pop_white_pct <- (zip_data$pop_white/zip_data$pop_total)*100

zip_data %>% select(state_code,zip_code,pop_white,pop_total,pop_white_pct) %>% head(n=10)
#> # A tibble: 10 x 5
#>   state_code zip_code pop_white pop_total pop_white_pct
#>   <fct>      <chr>      <int>    <int>    <dbl>
#> 1 MA        01001        15079    17423     86.5
#> 2 MA        01002        22082    29970     73.7
#> 3 MA        01003         8295    11296     73.4
#> 4 MA        01005         5008     5228     95.8
#> 5 MA        01007        13601    14888     91.4
#> 6 MA        01008         1178     1194     98.7
#> 7 MA        01009          237       237     100
#> 8 MA        01010         3660     3718     98.4
#> 9 MA        01011         1425     1523     93.6
#> 10 MA       01012          509       528     96.4

zip_data$pop_white_pct <- NULL # remove variable

#latinx
zip_data$pop_latinx_pct <- (zip_data$pop_latinx/zip_data$pop_total)*100

zip_data %>% select(state_code,zip_code,pop_latinx,pop_total,pop_latinx_pct) %>% head(n=10)
#> # A tibble: 10 x 5
#>   state_code zip_code pop_latinx pop_total pop_latinx_pct
```

```
#>   <fct>      <chr>      <int>      <int>      <dbl>
#> 1 MA       01001      1314      17423      7.54
#> 2 MA       01002      1870      29970      6.24
#> 3 MA       01003       526      11296      4.66
#> 4 MA       01005        77      5228      1.47
#> 5 MA       01007       305      14888      2.05
#> 6 MA       01008         4      1194      0.335
#> 7 MA       01009         0       237         0
#> 8 MA       01010        43      3718      1.16
#> 9 MA       01011        51      1523      3.35
#> 10 MA      01012        11       528      2.08
```

```
zip_data$pop_latinx_pct <- NULL # remove variable
```

Question 1: Modify your approach to creating these percent race/ethnicity variables outside of a function

Specific task

- Instead of using this “Base R” approach to create variables
 - `data_frame_name$var_name_pct <- (data_frame_namevar_name/zip_datapop_total)*100`
- I want you to use this “Base R” approach to create variables
 - `data_frame_name[["var_name_pct"]] <- (data_frame_name[["var_name"]]/zip_data[["pop_total"]])*100`
- Perform task outside of a function for at least two race/ethnicity categories (can be white and latinx)

/1

```
names(zip_data)
#> [1] "state_code"      "zip_code"      "pop_total"
#> [4] "pop_white"      "pop_black"     "pop_nativeamer"
#> [7] "pop_asian"      "pop_nativehawaii" "pop_otherrace"
#> [10] "pop_tworaces"   "pop_latinx"
#white
zip_data[["pop_white_pct"]] <- (zip_data[["pop_white"]]/zip_data[["pop_total"]])*100

zip_data %>% select(state_code,zip_code,pop_white,pop_total,pop_white_pct) %>% head(n=10)
#> # A tibble: 10 x 5
#>   state_code zip_code pop_white pop_total pop_white_pct
#>   <fct>      <chr>      <int>      <int>      <dbl>
#> 1 MA       01001      15079      17423      86.5
#> 2 MA       01002      22082      29970      73.7
#> 3 MA       01003       8295      11296      73.4
#> 4 MA       01005       5008       5228      95.8
#> 5 MA       01007      13601      14888      91.4
#> 6 MA       01008       1178       1194      98.7
#> 7 MA       01009        237        237      100
#> 8 MA       01010       3660       3718      98.4
#> 9 MA       01011       1425       1523      93.6
#> 10 MA      01012        509        528      96.4

zip_data$pop_white_pct <- NULL # remove variable
```

```
#latinx
zip_data[["pop_latinx_pct"]] <- (zip_data[["pop_latinx"]]/zip_data[["pop_total"]])*100

zip_data %>% select(state_code,zip_code,pop_latinx,pop_total,pop_latinx_pct) %>% head(n=10)
#> # A tibble: 10 x 5
#>   state_code zip_code pop_latinx pop_total pop_latinx_pct
#>   <fct>      <chr>      <int>      <int>      <dbl>
#> 1 MA        01001        1314      17423        7.54
#> 2 MA        01002        1870      29970        6.24
#> 3 MA        01003         526      11296        4.66
#> 4 MA        01005          77       5228        1.47
#> 5 MA        01007         305      14888        2.05
#> 6 MA        01008          4       1194        0.335
#> 7 MA        01009          0       237         0
#> 8 MA        01010         43       3718        1.16
#> 9 MA        01011         51      1523        3.35
#> 10 MA       01012         11       528        2.08

zip_data$pop_latinx_pct <- NULL # remove variable
```

Question 2: Modify the function call you use to create these percent race/ethnicity variables within a function

First, you will run code below and then complete the specific task

Run code below [solution to previous problem set] which creates and calls the `pct_race` function to create percent race/ethnicity variables

- Note: the `pct_race` function below is slightly revised version of the solution from the previous problem set. Specifically, the below function only takes two arguments. In previous problem set I told you this function should take three arguments. I realized that one of these arguments was superfluous.
- Make sure to delete variable after checking that your function worked
 - `zip_data$pop_latinx_pct <- NULL`

/0.5

```
pct_race <- function(pop_var,total_var){
  (pop_var/total_var)*100 # this is what function returns; this code exists only inside function and wi
}

#show what this function returns fpr these inputs; but note that this doesn't create anything
str(pct_race(zip_data$pop_white,zip_data$pop_total))
#> num [1:32989] 86.5 73.7 73.4 95.8 91.4 ...

#call function to create pct white variable
zip_data$pop_white_pct <- pct_race(zip_data$pop_white,zip_data$pop_total)

zip_data %>% select(state_code,zip_code,pop_white,pop_total,pop_white_pct) %>% head(n=10)
#> # A tibble: 10 x 5
#>   state_code zip_code pop_white pop_total pop_white_pct
#>   <fct>      <chr>      <int>      <int>      <dbl>
#> 1 MA        01001      15079      17423        86.5
```

```

#> 2 MA      01002      22082      29970      73.7
#> 3 MA      01003      8295      11296      73.4
#> 4 MA      01005      5008      5228      95.8
#> 5 MA      01007     13601     14888      91.4
#> 6 MA      01008      1178      1194      98.7
#> 7 MA      01009       237       237      100
#> 8 MA      01010      3660      3718      98.4
#> 9 MA      01011      1425      1523      93.6
#> 10 MA     01012       509       528      96.4

zip_data$pop_white_pct <- NULL # remove variable

#call function to create pct latinx variable
zip_data$pop_latinx_pct <- pct_race(zip_data$pop_latinx,zip_data$pop_total)

zip_data %>% select(zip_code,pop_latinx,pop_total,pop_latinx_pct) %>% head(n=10)
#> # A tibble: 10 x 4
#>   zip_code pop_latinx pop_total pop_latinx_pct
#>   <chr>      <int>      <int>      <dbl>
#> 1 01001      1314      17423      7.54
#> 2 01002      1870      29970      6.24
#> 3 01003       526      11296      4.66
#> 4 01005        77       5228      1.47
#> 5 01007       305     14888      2.05
#> 6 01008         4       1194      0.335
#> 7 01009         0        237         0
#> 8 01010        43       3718      1.16
#> 9 01011        51      1523      3.35
#> 10 01012        11        528      2.08

zip_data$pop_latinx_pct <- NULL # remove variable

```

Specific task/question: Modify the function call you use to create these percent race/ethnicity variables within a function. Here are additional details

- Modify the function **call** of your `pct_race` function so that you replace instances of `df_name$var_name` with `'df_name[["var_name"]]`
- Note: the actual function `pct_race` will be exactly the same as before
- Call function for at least two race/ethnicity groups
- Make sure to delete variable after checking that your function worked
 - `zip_data$pop_latinx_pct <- NULL`

/1

```

pct_race <- function(pop_var,total_var){

  (pop_var/total_var)*100 # this is what function returns; this code exists only inside function and wi
}

#call function to create pct white variable
zip_data[["pop_white_pct"]] <- pct_race(zip_data[["pop_white"]],zip_data[["pop_total"]])

zip_data %>% select(state_code,zip_code,pop_white,pop_total,pop_white_pct) %>% head(n=10)

```

```

#> # A tibble: 10 x 5
#>   state_code zip_code pop_white pop_total pop_white_pct
#>   <fct>      <chr>    <int>    <int>    <dbl>
#> 1 MA        01001      15079    17423     86.5
#> 2 MA        01002      22082    29970     73.7
#> 3 MA        01003       8295    11296     73.4
#> 4 MA        01005       5008     5228     95.8
#> 5 MA        01007      13601    14888     91.4
#> 6 MA        01008       1178     1194     98.7
#> 7 MA        01009        237     237      100
#> 8 MA        01010       3660     3718     98.4
#> 9 MA        01011       1425     1523     93.6
#> 10 MA       01012        509      528     96.4

zip_data$pop_white_pct <- NULL # remove variable

#call function to create pct latinx variable
zip_data[["pop_latinx_pct"]] <- pct_race(zip_data[["pop_latinx"]],zip_data[["pop_total"]])

zip_data %>% select(zip_code,pop_latinx,pop_total,pop_latinx_pct) %>% head(n=10)
#> # A tibble: 10 x 4
#>   zip_code pop_latinx pop_total pop_latinx_pct
#>   <chr>      <int>    <int>    <dbl>
#> 1 01001      1314    17423     7.54
#> 2 01002      1870    29970     6.24
#> 3 01003       526    11296     4.66
#> 4 01005        77     5228     1.47
#> 5 01007       305    14888     2.05
#> 6 01008         4     1194     0.335
#> 7 01009         0     237       0
#> 8 01010        43     3718     1.16
#> 9 01011        51     1523     3.35
#> 10 01012        11      528     2.08

zip_data$pop_latinx_pct <- NULL # remove variable

```

Question 3: Modify function and function call used to create percent race/ethnicity variables within a function

Specific requirements

- Modify function so that the name of the data frame is a separate argument from the name of the variables and call function for at least two race/ethnicity groups
- Function will now take three arguments:
 1. `df`: name of the data frame (e.g., `zip_data`)
 2. `pop_var`: name of the variable that is the numerator for the percent race variable
 3. `total_var`: name of the variable that is the denominator for the percent race variable
- Hint for how to modify function body
 - Change code from this:


```
* (pop_var/total_var)*100
```
 - To this:


```
* (df[[pop_var]]/df[[total_var]])*100
```
- Hint for program call

- For the arguments `pop_var` and `total_var` you will now just refer to "variable name" rather than `df_name[["variable name"]]`
 - * Note: In function call, the values for `pop_var` and `total_var` should be in quotes
- However, in program call for `df` will be `df name` [i.e., without quotes]
 - * Hint: `df[["new variable"]] <- pct_race(df, "variable", "variable")`

/2

```
pct_race <- function(df,pop_var,total_var){
  (df[[pop_var]]/df[[total_var]])*100
}

#show what this function returns fpr these inputs; but note that this doesn't create anything
str(pct_race(zip_data,"pop_white","pop_total"))
#> num [1:32989] 86.5 73.7 73.4 95.8 91.4 ...

#call function to create pct white variable
zip_data[["pop_white_pct"]] <- pct_race(zip_data,"pop_white","pop_total")

zip_data %>% select(state_code,zip_code,pop_white,pop_total,pop_white_pct) %>% head(n=10)
#> # A tibble: 10 x 5
#>   state_code zip_code pop_white pop_total pop_white_pct
#>   <fct>      <chr>    <int>    <int>    <dbl>
#> 1 MA        01001      15079    17423     86.5
#> 2 MA        01002      22082    29970     73.7
#> 3 MA        01003       8295    11296     73.4
#> 4 MA        01005       5008     5228     95.8
#> 5 MA        01007      13601    14888     91.4
#> 6 MA        01008       1178     1194     98.7
#> 7 MA        01009        237       237     100
#> 8 MA        01010       3660     3718     98.4
#> 9 MA        01011       1425     1523     93.6
#> 10 MA       01012        509       528     96.4

zip_data$pop_white_pct <- NULL # remove variable

#call function to create pct latinx variable
zip_data[["pop_latinx_pct"]] <- pct_race(zip_data,"pop_latinx","pop_total")

zip_data %>% select(zip_code,pop_latinx,pop_total,pop_latinx_pct) %>% head(n=10)
#> # A tibble: 10 x 4
#>   zip_code pop_latinx pop_total pop_latinx_pct
#>   <chr>      <int>    <int>    <dbl>
#> 1 01001      1314    17423     7.54
#> 2 01002      1870    29970     6.24
#> 3 01003       526    11296     4.66
#> 4 01005        77     5228     1.47
#> 5 01007       305    14888     2.05
#> 6 01008         4     1194     0.335
#> 7 01009         0       237         0
#> 8 01010        43     3718     1.16
#> 9 01011        51     1523     3.35
```

```
#> 10 01012      11      528      2.08

zip_data$pop_latinx_pct <- NULL # remove variable
```

Part 2: Loops

There are three ways to loop over a data frame:

1. Loop over elements
 - e.g., sequence syntax is: `for (i in data_frame_name)`
2. Loop over element names
 - e.g., sequence syntax is: `for (i in names(data_frame_name))`
3. Loop over numeric indices of element position
 - e.g., sequence syntax is: `for (i in 1:length(data_frame_name))`

This part of the problem set will give you some practice looping over elements of a data frame using these three approaches. First, you will run the code below to create data frame called `zip_tiny` that consists of the first 10 observations of data frame `zip_data`. Then you will answer specific questions. All questions for this part of the problem set will utilize the data frame `zip_tiny`

Run the code below to create data frame called `zip_tiny` that consists of the first 10 observations of data frame `zip_data`

- Note: when we created `zip_data` above, we sorted by `zip_code` so no need to `arrange()` observations when creating `zip_tiny`

/0.5

```
#names(zip_data)

zip_tiny <- NULL # remove object if it exists
zip_tiny <- zip_data[1:10,] # base r approach
#zip_tiny <- zip_data %>% head(n=10) # tidyverse approach; yields same result as base r approach

#investigate object
typeof(zip_tiny) # list
#> [1] "list"
class(zip_tiny) # tibble, which is particular kind of data frame
#> [1] "tbl_df"      "tbl"          "data.frame"
str(zip_tiny)
#> Classes 'tbl_df', 'tbl' and 'data.frame':   10 obs. of  11 variables:
#> $ state_code      : Factor w/ 52 levels "AK","AL","AR",...: 20 20 20 20 20 20 20 20 20 20
#> $ zip_code        : chr  "01001" "01002" "01003" "01005" ...
#> $ pop_total       : int   17423 29970 11296 5228 14888 1194 237 3718 1523 528
#> $ pop_white       : int   15079 22082 8295 5008 13601 1178 237 3660 1425 509
#> $ pop_black       : int    209 1578 636 105 125 0 0 9 15 0
#> $ pop_nativeamer  : int     5 74 30 0 0 0 0 0 0 0
#> $ pop_asian       : int    603 3502 1538 32 443 7 0 6 11 0
#> $ pop_nativehawai: int     24 17 0 0 0 0 0 0 0 2
#> $ pop_otherrace   : int     88 72 45 0 34 0 0 0 4 0
#> $ pop_tworaces    : int    101 775 226 6 380 5 0 0 17 6
#> $ pop_latinx      : int   1314 1870 526 77 305 4 0 43 51 11
```


Question 1: Loop across elements of object

For this question, you get full credit just by running the code below. But try to understand how the sequence syntax works and what each line of the body is doing.

- Note that one line of the loop body calculates the mean value of the variable using the `mean()` function. The `mean()` function will not calculate mean values for variables that do not have numeric or logical classes (e.g., character vars, factor vars). But this won't stop code from running, so you can ignore these warnings.

/0.5

```
for (i in zip_tiny) {  
  
  cat("Value of object i=",i, fill=TRUE) # value of local variable i  
  cat("Object type=",typeof(i),"; length=",length(i),"; class=",class(i),sep=" ",fill=TRUE) # type, leng  
  print(attributes(i)) # note: we have to print attributes separately rather than in cat() because if v  
  cat("Mean value of object i=",mean(i, na.rm = TRUE),"\n", fill=TRUE) # calculate mean value of variab  
  #cat("\n",fill=TRUE)  
  
}  
  
#> Value of object i= 20 20 20 20 20 20 20 20 20 20  
#> Object type=integer; length=10; class=factor  
#> $levels  
#> [1] "AK" "AL" "AR" "AZ" "CA" "CO" "CT" "DC" "DE" "FL" "GA" "HI" "IA" "ID"  
#> [15] "IL" "IN" "KS" "KY" "LA" "MA" "MD" "ME" "MI" "MN" "MO" "MS" "MT" "NC"  
#> [29] "ND" "NE" "NH" "NJ" "NM" "NV" "NY" "OH" "OK" "OR" "PA" "PR" "RI" "SC"  
#> [43] "SD" "TN" "TX" "UT" "VA" "VT" "WA" "WI" "WV" "WY"  
#>  
#> $class  
#> [1] "factor"  
#> Warning in mean.default(i, na.rm = TRUE): argument is not numeric or  
#> logical: returning NA  
#> Mean value of object i= NA  
#>  
#> Value of object i= 01001 01002 01003 01005 01007 01008 01009 01010 01011  
#> 01012  
#> Object type=character; length=10; class=character  
#> NULL  
#> Warning in mean.default(i, na.rm = TRUE): argument is not numeric or  
#> logical: returning NA  
#> Mean value of object i= NA  
#>  
#> Value of object i= 17423 29970 11296 5228 14888 1194 237 3718 1523 528  
#> Object type=integer; length=10; class=integer  
#> NULL  
#> Mean value of object i= 8600.5  
#>  
#> Value of object i= 15079 22082 8295 5008 13601 1178 237 3660 1425 509  
#> Object type=integer; length=10; class=integer  
#> NULL  
#> Mean value of object i= 7107.4  
#>  
#> Value of object i= 209 1578 636 105 125 0 0 9 15 0  
#> Object type=integer; length=10; class=integer
```

```

#> NULL
#> Mean value of object i= 267.7
#>
#> Value of object i= 5 74 30 0 0 0 0 0 0 0
#> Object type=integer; length=10; class=integer
#> NULL
#> Mean value of object i= 10.9
#>
#> Value of object i= 603 3502 1538 32 443 7 0 6 11 0
#> Object type=integer; length=10; class=integer
#> NULL
#> Mean value of object i= 614.2
#>
#> Value of object i= 24 17 0 0 0 0 0 0 0 2
#> Object type=integer; length=10; class=integer
#> NULL
#> Mean value of object i= 4.3
#>
#> Value of object i= 88 72 45 0 34 0 0 0 4 0
#> Object type=integer; length=10; class=integer
#> NULL
#> Mean value of object i= 24.3
#>
#> Value of object i= 101 775 226 6 380 5 0 0 17 6
#> Object type=integer; length=10; class=integer
#> NULL
#> Mean value of object i= 151.6
#>
#> Value of object i= 1314 1870 526 77 305 4 0 43 51 11
#> Object type=integer; length=10; class=integer
#> NULL
#> Mean value of object i= 420.1

```

Question 2: Loop across names of object elements

Question: Write a loop that loops across **names** of object elements of data frame `zip_tiny` [as opposed to looping across element contents as above]

- The body of the loop only needs to contain this line of code:
 - `cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)`

SOLUTION

```

/1
for (i in names(zip_tiny)) {
  cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)
}
#>
#> value of object i=state_code; type=character
#>
#> value of object i=zip_code; type=character
#>
#> value of object i=pop_total; type=character
#>

```

```
#> value of object i=pop_white; type=character
#>
#> value of object i=pop_black; type=character
#>
#> value of object i=pop_nativeamer; type=character
#>
#> value of object i=pop_asian; type=character
#>
#> value of object i=pop_nativehawaii; type=character
#>
#> value of object i=pop_otherrace; type=character
#>
#> value of object i=pop_tworaces; type=character
#>
#> value of object i=pop_latinx; type=character
```

Question 3: Loop across names of object elements continued

Question: Modify the previous loop to also print the element contents associated with each element name, using `[]` rather than `[[i]]` to access the element contents

- I want you to print the structure (i.e., `str()` function) of the element contents rather than directly printing the element contents
 - Hint for syntax: `print(str(data_frame_name[i]))`
- First line of loop body should be the same as previous loop:
 - `cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)`
- You should have two lines of code in your loop body

SOLUTION

```
/1
for (i in names(zip_tiny)) {
  cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)
  print(str(zip_tiny[i])) # "Access element contents using []"
}
#>
#> value of object i=state_code; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ state_code: Factor w/ 52 levels "AK","AL","AR",...: 20 20 20 20 20 20 20 20 20 20
#> NULL
#>
#> value of object i=zip_code; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ zip_code: chr "01001" "01002" "01003" "01005" ...
#> NULL
#>
#> value of object i=pop_total; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ pop_total: int 17423 29970 11296 5228 14888 1194 237 3718 1523 528
#> NULL
#>
#> value of object i=pop_white; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
```

```

#> $ pop_white: int 15079 22082 8295 5008 13601 1178 237 3660 1425 509
#> NULL
#>
#> value of object i=pop_black; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ pop_black: int 209 1578 636 105 125 0 0 9 15 0
#> NULL
#>
#> value of object i=pop_nativeamer; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ pop_nativeamer: int 5 74 30 0 0 0 0 0 0 0
#> NULL
#>
#> value of object i=pop_asian; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ pop_asian: int 603 3502 1538 32 443 7 0 6 11 0
#> NULL
#>
#> value of object i=pop_nativehawaii; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ pop_nativehawaii: int 24 17 0 0 0 0 0 0 0 2
#> NULL
#>
#> value of object i=pop_otherrace; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ pop_otherrace: int 88 72 45 0 34 0 0 0 4 0
#> NULL
#>
#> value of object i=pop_tworaces; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ pop_tworaces: int 101 775 226 6 380 5 0 0 17 6
#> NULL
#>
#> value of object i=pop_latinx; type=character
#> Classes 'tbl_df', 'tbl' and 'data.frame': 10 obs. of 1 variable:
#> $ pop_latinx: int 1314 1870 526 77 305 4 0 43 51 11
#> NULL

```

Question 4: Loop across names of object elements continued

Question: Modify the previous loop to revise the way the loop prints the element contents associated with each element name, this time using `[[]]` rather than `[]` to access the element contents

- I want you to print the structure (i.e., `str()` function) of the element contents rather than directly printing the element contents
 - Hint for syntax: `print(str(data_frame_name[[i]]))`
- First line of loop body should be the same as previous loop:
 - `cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)`
- You should have two lines of code in your loop body (same as above)

SOLUTION

/1

```

for (i in names(zip_tiny)) {
  cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)
  print(str(zip_tiny[[i]])) # "Access element contents using [[]]"
}
#>
#> value of object i=state_code; type=character
#> Factor w/ 52 levels "AK","AL","AR",...: 20 20 20 20 20 20 20 20 20 20
#> NULL
#>
#> value of object i=zip_code; type=character
#> chr [1:10] "01001" "01002" "01003" "01005" "01007" "01008" "01009" ...
#> NULL
#>
#> value of object i=pop_total; type=character
#> int [1:10] 17423 29970 11296 5228 14888 1194 237 3718 1523 528
#> NULL
#>
#> value of object i=pop_white; type=character
#> int [1:10] 15079 22082 8295 5008 13601 1178 237 3660 1425 509
#> NULL
#>
#> value of object i=pop_black; type=character
#> int [1:10] 209 1578 636 105 125 0 0 9 15 0
#> NULL
#>
#> value of object i=pop_nativeamer; type=character
#> int [1:10] 5 74 30 0 0 0 0 0 0 0
#> NULL
#>
#> value of object i=pop_asian; type=character
#> int [1:10] 603 3502 1538 32 443 7 0 6 11 0
#> NULL
#>
#> value of object i=pop_nativehawaii; type=character
#> int [1:10] 24 17 0 0 0 0 0 0 0 2
#> NULL
#>
#> value of object i=pop_otherrace; type=character
#> int [1:10] 88 72 45 0 34 0 0 0 4 0
#> NULL
#>
#> value of object i=pop_tworaces; type=character
#> int [1:10] 101 775 226 6 380 5 0 0 17 6
#> NULL
#>
#> value of object i=pop_latinx; type=character
#> int [1:10] 1314 1870 526 77 305 4 0 43 51 11
#> NULL

```

Question 5: Loop across names of object elements continued

Question: When using the `for (i in names(data_frame_name))` approach to loop over elements in a data frame, what is the difference between objects created by the syntax `data_frame_name[i]` and objects created by the syntax `data_frame_name[[i]]`?

YOUR ANSWER HERE:

/1

SOLUTION: Objects created by the syntax `data_frame_name[i]` have the same `type` and `class` as the data frame. By contrast objects created by the syntax `data_frame_name[[i]]` remove a level of hierarchy, so these objects are always vectors.

Question 6: Loop across names of object elements continued

Question: Modify the previous loop to add a line that prints the mean value for each element of the data frame `zip_tiny`

- First line of loop body should be:
 - `cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)`
- Second line of loop body should print the structure (i.e., `str()` function) of the element contents [this line will be the same as second line in previous loops]
- Third line of loop body will print the mean value for each element
 - Hint: when calculating means, use the `data_frame_name[[i]]` approach to access element contents rather than the `data_frame_name[i]` approach
- Third line of code should start with:
 - * `cat("Mean of element named", i, "is", ...)`
- Note: the `mean()` function will not calculate mean values for variables that do not have numeric or logical classes (e.g., character vars, factor vars). But this won't stop code from running, so you can ignore these warnings.

/2

```
for (i in names(zip_tiny)) {  
  cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)  
  print(str(zip_tiny[[i]])) # "Access element contents using [[]"  
  cat("Mean of element named", i, "is", mean(zip_tiny[[i]], na.rm = TRUE), fill=TRUE)  
}  
#>  
#> value of object i=state_code; type=character  
#> Factor w/ 52 levels "AK","AL","AR",...: 20 20 20 20 20 20 20 20 20 20  
#> NULL  
#> Warning in mean.default(zip_tiny[[i]], na.rm = TRUE): argument is not  
#> numeric or logical: returning NA  
#> Mean of element named state_code is NA  
#>  
#> value of object i=zip_code; type=character  
#> chr [1:10] "01001" "01002" "01003" "01005" "01007" "01008" "01009" ...  
#> NULL  
#> Warning in mean.default(zip_tiny[[i]], na.rm = TRUE): argument is not  
#> numeric or logical: returning NA  
#> Mean of element named zip_code is NA  
#>  
#> value of object i=pop_total; type=character
```

```

#> int [1:10] 17423 29970 11296 5228 14888 1194 237 3718 1523 528
#> NULL
#> Mean of element named pop_total is 8600.5
#>
#> value of object i=pop_white; type=character
#> int [1:10] 15079 22082 8295 5008 13601 1178 237 3660 1425 509
#> NULL
#> Mean of element named pop_white is 7107.4
#>
#> value of object i=pop_black; type=character
#> int [1:10] 209 1578 636 105 125 0 0 9 15 0
#> NULL
#> Mean of element named pop_black is 267.7
#>
#> value of object i=pop_nativeamer; type=character
#> int [1:10] 5 74 30 0 0 0 0 0 0 0
#> NULL
#> Mean of element named pop_nativeamer is 10.9
#>
#> value of object i=pop_asian; type=character
#> int [1:10] 603 3502 1538 32 443 7 0 6 11 0
#> NULL
#> Mean of element named pop_asian is 614.2
#>
#> value of object i=pop_nativehawaii; type=character
#> int [1:10] 24 17 0 0 0 0 0 0 0 2
#> NULL
#> Mean of element named pop_nativehawaii is 4.3
#>
#> value of object i=pop_otherrace; type=character
#> int [1:10] 88 72 45 0 34 0 0 0 4 0
#> NULL
#> Mean of element named pop_otherrace is 24.3
#>
#> value of object i=pop_tworaces; type=character
#> int [1:10] 101 775 226 6 380 5 0 0 17 6
#> NULL
#> Mean of element named pop_tworaces is 151.6
#>
#> value of object i=pop_latinx; type=character
#> int [1:10] 1314 1870 526 77 305 4 0 43 51 11
#> NULL
#> Mean of element named pop_latinx is 420.1

```

Question 7: Loop across names of object elements continued

Question: Modify the previous loop (which calculates mean values in the last line of the loop body) so that the loop is only run for variables that are logical or numeric

- The body of the loop will be exactly the same as the body of the previous loop
- Change the sequence syntax as follows:
 - from this approach: `for (i in names(zip_tiny))`
 - to this approach: `for (i in c("var_name1", "var_name2", "var_name3", "etc..."))`

- * Essentially, you will manually insert the name of all variables from `zip_data` that have a numeric class
- * Note that variable names must be enclosed by quotes
- * Note that a more advanced approach to this is on page 80 of lecture 10

SOLUTION

/1.5

```
names(zip_tiny)
#> [1] "state_code"      "zip_code"      "pop_total"
#> [4] "pop_white"      "pop_black"     "pop_nativeamer"
#> [7] "pop_asian"      "pop_nativehawaii" "pop_otherrace"
#> [10] "pop_tworaces"   "pop_latinx"

for (i in c("pop_total", "pop_white", "pop_black", "pop_nativeamer", "pop_asian", "pop_nativehawaii", "pop_otherrace")) {
  cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)
  print(str(zip_tiny[[i]])) # "Access element contents using [[]]"
  cat("mean of element named", i, "is", mean(zip_tiny[[i]], na.rm = TRUE), fill=TRUE)
}
#>
#> value of object i=pop_total; type=character
#> int [1:10] 17423 29970 11296 5228 14888 1194 237 3718 1523 528
#> NULL
#> mean of element named pop_total is 8600.5
#>
#> value of object i=pop_white; type=character
#> int [1:10] 15079 22082 8295 5008 13601 1178 237 3660 1425 509
#> NULL
#> mean of element named pop_white is 7107.4
#>
#> value of object i=pop_black; type=character
#> int [1:10] 209 1578 636 105 125 0 0 9 15 0
#> NULL
#> mean of element named pop_black is 267.7
#>
#> value of object i=pop_nativeamer; type=character
#> int [1:10] 5 74 30 0 0 0 0 0 0 0
#> NULL
#> mean of element named pop_nativeamer is 10.9
#>
#> value of object i=pop_asian; type=character
#> int [1:10] 603 3502 1538 32 443 7 0 6 11 0
#> NULL
#> mean of element named pop_asian is 614.2
#>
#> value of object i=pop_nativehawaii; type=character
#> int [1:10] 24 17 0 0 0 0 0 0 0 2
#> NULL
#> mean of element named pop_nativehawaii is 4.3
#>
#> value of object i=pop_otherrace; type=character
#> int [1:10] 88 72 45 0 34 0 0 0 4 0
#> NULL
#> mean of element named pop_otherrace is 24.3
```



```
#>
#> value of object i=pop_tworaces; type=character
#> int [1:10] 101 775 226 6 380 5 0 0 17 6
#> NULL
#> mean of element named pop_tworaces is 151.6
#>
#> value of object i=pop_latinx; type=character
#> int [1:10] 1314 1870 526 77 305 4 0 43 51 11
#> NULL
#> mean of element named pop_latinx is 420.1
```

Question 8: Loop over elements based on numeric element position

First, run this code to become acquainted with the components involved for writing the `sequence` syntax for this approach to looping

/0.5

```
zip_tiny
#> # A tibble: 10 x 11
#>   state_code zip_code pop_total pop_white pop_black pop_nativeamer
#>   <fct>      <chr>      <int>      <int>      <int>      <int>
#> 1 MA        01001        17423       15079        209         5
#> 2 MA        01002       29970       22082       1578        74
#> 3 MA        01003       11296        8295        636        30
#> 4 MA        01005        5228        5008        105         0
#> 5 MA        01007       14888       13601        125         0
#> 6 MA        01008        1194        1178         0         0
#> 7 MA        01009         237         237         0         0
#> 8 MA        01010        3718        3660         9         0
#> 9 MA        01011        1523        1425         15         0
#> 10 MA       01012         528         509         0         0
#> # ... with 5 more variables: pop_asian <int>, pop_nativehawaii <int>,
#> #   pop_otherrace <int>, pop_tworaces <int>, pop_latinx <int>
length(zip_tiny) # length = number of elements = number of variables (when object is data frame)
#> [1] 11
1:length(zip_tiny)
#> [1] 1 2 3 4 5 6 7 8 9 10 11
```

Question: Use `for (i in 1:length(data_frame_name))` approach to loop over elements of the data frame `zip_tiny` based on element position.

- Your loop body should be:
 - `cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)`

/1

```
for (i in 1:length(zip_tiny)) {
  cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)
}
#>
#> value of object i=1; type=integer
#>
#> value of object i=2; type=integer
#>
#> value of object i=3; type=integer
```

```

#>
#> value of object i=4; type=integer
#>
#> value of object i=5; type=integer
#>
#> value of object i=6; type=integer
#>
#> value of object i=7; type=integer
#>
#> value of object i=8; type=integer
#>
#> value of object i=9; type=integer
#>
#> value of object i=10; type=integer
#>
#> value of object i=11; type=integer

```

Question 9: Loop over elements based on numeric element position, continued

Question: Modify the loop above to add a second line that prints out the name of the variable associated with that element position

- Hint for syntax: `names(data_frame_name)[[i]]`
- First line of loop body should be:
 - `cat("\n", "Value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)`
- Second line of loop body should start with:
 - `cat("Variable name associated with object i =", ...)`

/1

```

for (i in 1:length(zip_tiny)) {
  cat("\n", "Value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)
  cat("Variable name associated with object i = ", names(zip_tiny)[[i]], sep="", fill=TRUE)
}
#>
#> Value of object i=1; type=integer
#> Variable name associated with object i = state_code
#>
#> Value of object i=2; type=integer
#> Variable name associated with object i = zip_code
#>
#> Value of object i=3; type=integer
#> Variable name associated with object i = pop_total
#>
#> Value of object i=4; type=integer
#> Variable name associated with object i = pop_white
#>
#> Value of object i=5; type=integer
#> Variable name associated with object i = pop_black
#>
#> Value of object i=6; type=integer
#> Variable name associated with object i = pop_nativeamer
#>

```

```

#> Value of object i=7; type=integer
#> Variable name associated with object i = pop_asian
#>
#> Value of object i=8; type=integer
#> Variable name associated with object i = pop_nativehawaii
#>
#> Value of object i=9; type=integer
#> Variable name associated with object i = pop_otherrace
#>
#> Value of object i=10; type=integer
#> Variable name associated with object i = pop_tworaces
#>
#> Value of object i=11; type=integer
#> Variable name associated with object i = pop_latinx

```

Question 10: Loop over elements based on numeric element position, continued

Question: Keeping all the code from the loop above, add a third line to the loop body that prints the structure of the element contents associated with that variable, using `[[]]` rather than `[]` to access element contents

- syntax hint: `print(str(data_frame_name[[i]]))`

/1.5

```

for (i in 1:length(zip_tiny)) {
  cat("\n", "value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)
  cat("variable name associated with object i = ", names(zip_tiny)[[i]], sep="", fill=TRUE)
  print(str(zip_tiny[[i]]))
}
#>
#> value of object i=1; type=integer
#> variable name associated with object i = state_code
#> Factor w/ 52 levels "AK","AL","AR",...: 20 20 20 20 20 20 20 20 20 20
#> NULL
#>
#> value of object i=2; type=integer
#> variable name associated with object i = zip_code
#> chr [1:10] "01001" "01002" "01003" "01005" "01007" "01008" "01009" ...
#> NULL
#>
#> value of object i=3; type=integer
#> variable name associated with object i = pop_total
#> int [1:10] 17423 29970 11296 5228 14888 1194 237 3718 1523 528
#> NULL
#>
#> value of object i=4; type=integer
#> variable name associated with object i = pop_white
#> int [1:10] 15079 22082 8295 5008 13601 1178 237 3660 1425 509
#> NULL
#>
#> value of object i=5; type=integer
#> variable name associated with object i = pop_black
#> int [1:10] 209 1578 636 105 125 0 0 9 15 0

```

```

#> NULL
#>
#> value of object i=6; type=integer
#> variable name associated with object i = pop_nativeamer
#> int [1:10] 5 74 30 0 0 0 0 0 0 0
#> NULL
#>
#> value of object i=7; type=integer
#> variable name associated with object i = pop_asian
#> int [1:10] 603 3502 1538 32 443 7 0 6 11 0
#> NULL
#>
#> value of object i=8; type=integer
#> variable name associated with object i = pop_nativehawaii
#> int [1:10] 24 17 0 0 0 0 0 0 0 2
#> NULL
#>
#> value of object i=9; type=integer
#> variable name associated with object i = pop_otherrace
#> int [1:10] 88 72 45 0 34 0 0 0 4 0
#> NULL
#>
#> value of object i=10; type=integer
#> variable name associated with object i = pop_tworaces
#> int [1:10] 101 775 226 6 380 5 0 0 17 6
#> NULL
#>
#> value of object i=11; type=integer
#> variable name associated with object i = pop_latinx
#> int [1:10] 1314 1870 526 77 305 4 0 43 51 11
#> NULL

```

Question 11: Loop over elements based on numeric element position, continued

Question: Keeping all the code from the loop above, add a fourth line to the loop body that prints the mean value for each element of the data frame `zip_tiny`

- Hint: when calculating means, use the `data_frame_name[[i]]` approach to access element contents rather than the `data_frame_name[i]` approach
- Note: the `mean()` function will not calculate mean values for variables that do not have numeric or logical classes (e.g., character vars, factor vars). But this won't stop code from running, so you can ignore these warnings.
- Fourth line of code should start with:
 - `cat("Mean of element named",names(df[[i]]),"is", ...)`

/2

```

for (i in 1:length(zip_tiny)) {
  cat("\n", "Value of object i=", i, "; type=", typeof(i), sep="", fill=TRUE)
  cat("Variable name associated with object i=", names(zip_tiny)[[i]], sep="", fill=TRUE)
  print(str(zip_tiny[[i]]))
  cat("Mean of element named", names(zip_tiny)[[i]], "is", mean(zip_tiny[[i]], na.rm = TRUE), fill=TRUE)
}

```

```

#>
#> Value of object i=1; type=integer
#> Variable name associated with object i=state_code
#> Factor w/ 52 levels "AK","AL","AR",...: 20 20 20 20 20 20 20 20 20 20
#> NULL
#> Warning in mean.default(zip_tiny[[i]], na.rm = TRUE): argument is not
#> numeric or logical: returning NA
#> Mean of element named state_code is NA
#>
#> Value of object i=2; type=integer
#> Variable name associated with object i=zip_code
#> chr [1:10] "01001" "01002" "01003" "01005" "01007" "01008" "01009" ...
#> NULL
#> Warning in mean.default(zip_tiny[[i]], na.rm = TRUE): argument is not
#> numeric or logical: returning NA
#> Mean of element named zip_code is NA
#>
#> Value of object i=3; type=integer
#> Variable name associated with object i=pop_total
#> int [1:10] 17423 29970 11296 5228 14888 1194 237 3718 1523 528
#> NULL
#> Mean of element named pop_total is 8600.5
#>
#> Value of object i=4; type=integer
#> Variable name associated with object i=pop_white
#> int [1:10] 15079 22082 8295 5008 13601 1178 237 3660 1425 509
#> NULL
#> Mean of element named pop_white is 7107.4
#>
#> Value of object i=5; type=integer
#> Variable name associated with object i=pop_black
#> int [1:10] 209 1578 636 105 125 0 0 9 15 0
#> NULL
#> Mean of element named pop_black is 267.7
#>
#> Value of object i=6; type=integer
#> Variable name associated with object i=pop_nativeamer
#> int [1:10] 5 74 30 0 0 0 0 0 0 0
#> NULL
#> Mean of element named pop_nativeamer is 10.9
#>
#> Value of object i=7; type=integer
#> Variable name associated with object i=pop_asian
#> int [1:10] 603 3502 1538 32 443 7 0 6 11 0
#> NULL
#> Mean of element named pop_asian is 614.2
#>
#> Value of object i=8; type=integer
#> Variable name associated with object i=pop_nativehawaii
#> int [1:10] 24 17 0 0 0 0 0 0 0 2
#> NULL
#> Mean of element named pop_nativehawaii is 4.3
#>

```

```

#> Value of object i=9; type=integer
#> Variable name associated with object i=pop_otherrace
#> int [1:10] 88 72 45 0 34 0 0 0 4 0
#> NULL
#> Mean of element named pop_otherrace is 24.3
#>
#> Value of object i=10; type=integer
#> Variable name associated with object i=pop_tworaces
#> int [1:10] 101 775 226 6 380 5 0 0 17 6
#> NULL
#> Mean of element named pop_tworaces is 151.6
#>
#> Value of object i=11; type=integer
#> Variable name associated with object i=pop_latinx
#> int [1:10] 1314 1870 526 77 305 4 0 43 51 11
#> NULL
#> Mean of element named pop_latinx is 420.1

```

Once finished, knit to (pdf) and upload both .Rmd and pdf files to class website under the week 10 tab
Remember to use this naming convention “lastname_firstname_ps10”