Managing Data with R

This tutorial is an attempt to demonstrate how to manage data with R. In this lab, we will be using **customer** dataset which can be downloaded from eLearn@USM.

Load the dataset into R, and name the data frame as custData.

```
> library(tidyverse)
```

```
> custData <- read_delim('cust.data.manage.csv', delim=',')
```

Correct column housing.type data type. Convert it from char to factor.

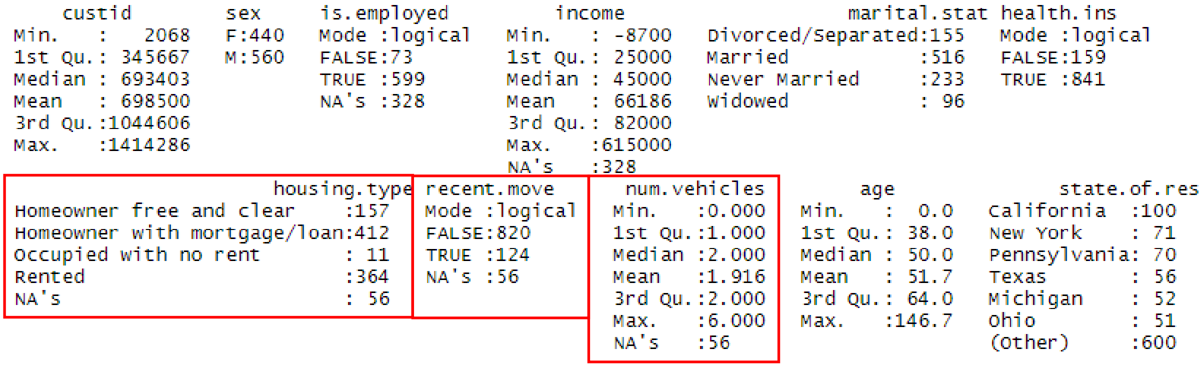
```
> custData$housing.type <- as.factor(custData$housing.type)
```

Handling Missing Values and Outliers

Use summary command to examine the distribution of the dataset.

```
> summary(custData)
```

A summary such as follows will be shown in the console.



```
     custid          sex       is.employed        income                   marital.stat  health.ins    
 Min.   :   2068   F:440   Mode :logical   Min.   : -8700   Divorced/Separated:155   Mode :logical  
 1st Qu.: 345667   M:560   FALSE:73        1st Qu.: 25000   Married           :516   FALSE:159      
 Median : 693403           TRUE :599       Median : 45000   Never Married     :233   TRUE :841      
 Mean   : 698500           NA's :328       Mean   : 66186   Widowed           : 96                  
 3rd Qu.:1044606                           3rd Qu.: 82000                                           
 Max.   :1414286                           Max.   :615000                                           
                                           NA's   :328                                              
                       housing.type recent.move     num.vehicles        age           state.of.res
 Homeowner free and clear    :157   Mode :logical   Min.   :0.000   Min.   :  0.0   California  :100  
 Homeowner with mortgage/loan:412   FALSE:820       1st Qu.:1.000   1st Qu.: 38.0   New York    : 71  
 Occupied with no rent       : 11   TRUE :124       Median :2.000   Median : 50.0   Pennsylvania: 70  
 Rented                      :364   NA's :56        Mean   :1.916   Mean   : 51.7   Texas       : 56  
 NA's                        : 56                   3rd Qu.:2.000   3rd Qu.: 64.0   Michigan    : 52  
                                                    Max.   :6.000   Max.   :146.7   Ohio        : 51  
                                                    NA's   :56                      (Other)     :600  
```

Dropping Missing Values

There are 1000 customers, 56 rows represent 6% of the data. It's not trivial but it's not a huge number. Let's analyse the three attributes.

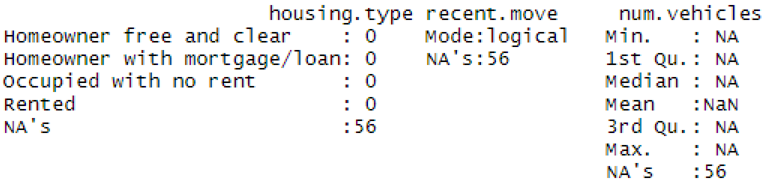
```
> custData_NAs = select(filter(custData, is.na(housing.type)),
  housing.type, recent.move, num.vehicles)
```

```
> summary(custData_NAs)
```

Similar output can be achieved using pipe operator

```
> custData %>% filter(is.na(housing.type)) %>% select(housing.type,
  recent.move, num.vehicles) %>% summary()
```

As we can see the three attributes missing exactly 56 values, means that it's the same customers in each case. So, it's probably safe to drop the rows with missing values.



```
                       housing.type recent.move    num.vehicles
 Homeowner free and clear    : 0    Mode:logical   Min.   : NA  
 Homeowner with mortgage/loan: 0    NA's:56        1st Qu.: NA  
 Occupied with no rent       : 0                   Median : NA  
 Rented                      : 0                   Mean   :NaN  
 NA's                        :56                   3rd Qu.: NA  
                                                   Max.   : NA  
                                                   NA's   :56   
```

We can use **drop_na** to drop all rows with missing values.

```
> custData %>% drop_na()
```

But we want to drop only the 56 rows. The remaining missing values will be imputed with some values. To drop the 56 rows, we can use one of the columns as parameter of drop_na. Notice that we are creating a subset since we do not want to replace the original tibble.

```
> custData_subset <- custData %>% drop_na("housing.type")
```

Filling Missing Values in Numerical Data

What should we do with the missing values in attribute income? There are 328 rows with missing values. We believe income is an important attribute and the rows should not be dropped. We can fill the missing values with the expected or mean income. Calculate the mean income as follows.

```
> meanIncome <- mean(custData_subset$income, na.rm=T)
```

We fill the missing value with mean income using **replace_na**.

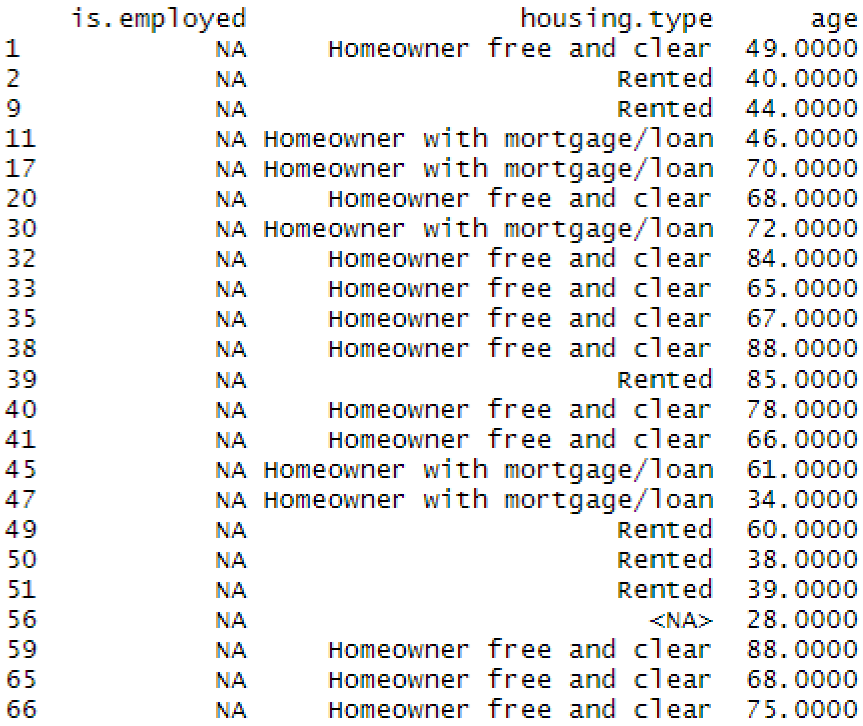
```
> custData_subset$income.fix <- custData_subset$income %>%
  replace_na(meanIncome)
```

The summary shows there is no missing value.

```
> summary(custData_subset$income.fix)
```

Filling Missing Values in Categorical Data

What about attribute is.employed? Examining the dataset, we can conclude that the customers might not in the active workforce and are not seeking paid employment.



```
    is.employed                housing.type      age
1            NA     Homeowner free and clear  49.0000
2            NA                       Rented  40.0000
9            NA                       Rented  44.0000
11           NA Homeowner with mortgage/loan  46.0000
17           NA Homeowner with mortgage/loan  70.0000
20           NA     Homeowner free and clear  68.0000
30           NA Homeowner with mortgage/loan  72.0000
32           NA     Homeowner free and clear  84.0000
33           NA     Homeowner free and clear  65.0000
35           NA     Homeowner free and clear  67.0000
38           NA     Homeowner free and clear  88.0000
39           NA                       Rented  85.0000
40           NA     Homeowner free and clear  78.0000
41           NA     Homeowner free and clear  66.0000
45           NA Homeowner with mortgage/loan  61.0000
47           NA Homeowner with mortgage/loan  34.0000
49           NA                       Rented  60.0000
50           NA                       Rented  38.0000
51           NA                       Rented  39.0000
56           NA                         <NA>  28.0000
59           NA     Homeowner free and clear  88.0000
65           NA     Homeowner free and clear  68.0000
66           NA     Homeowner free and clear  75.0000
```

Let’s group them into a single category. Here, we create a new category ("not it active workforce") and rename TRUE to "employed" and FALSE to "not employed".

```
> custData_subset$is.employed.fix <-
  ifelse(is.na(custData_subset$is.employed), "not in active workforce",
  ifelse(custData_subset$is.employed==T, "employed", "not employed"))
```

Replacing Outliers with Max/Min Values

What should we do with the negative value in attribute income.fix? We believe income is not supposed to have negative values. We can trim the values with 0 (minimum income is zero).

We replace the negative value(s) with 0.

```
> custData_subset$income.fix<-ifelse(custData_subset$income.fix<0, 0,
  custData_subset$income.fix)
```

The summary shows there is no negative value(s).

```
> summary(custData_subset$income.fix)
```

Converting Numerical Data to Categorical Data

We can also deal with the missing values by converting the attribute to categorical data. Then, we assign the missing values (NA) to "no income". To define income groups or range of interest, type the following statement.

```
> breaks <- c(0, 10000, 50000, 100000, 250000, 1000000)
```

Then, cut the data into groups using the defined groups.

```
> custData_subset$income.groups <- cut(custData_subset$income.fix,
  breaks=breaks, include.lowest=T)
```

Argument include.lowest=T is to make sure zero income data is included in the lowest group.

Data Transformation

Let's normalize the income by median income. Assuming we have median income for each state. Download the information from eLearn@USM and read the it into R.

```
> medianincome <- read.table("median.income.csv", sep=',', header=T)
```

Merge median income into customer data frame by matching the attribute custData$state.of.res to the attribute medianincome$State

```
> custData_subset <- merge(custData_subset, medianincome,
  by.x="state.of.res", by.y="State")
```

We can achieve similar output using pipe operator.

```
> custData_subset <- custData_subset %>% left_join(medianincome,
  by=c("state.of.res" = "State"))

> summary(custData_subset[,c("state.of.res", "income.fix",
  "Median.Income")])
```

Normalize the income by median income

```
> custData_subset <- mutate(custData_subset, income.fix.norm=income.fix
  /Median.Income)

> summary(custData_subset$income.fix.norm)
```

Exercises

Load Credit Risk dataset.

Replace negative values in Age column with median age.

Using IQR rule and empirical rule with -2.5σ and 2.5σ, determine the valid range of Credit.amount column. Use only positive values when determining the valid range.

Explain what to be done with the outliers in Credit.amount column.

Replace negative values in Credit.amount column with median value.

Derive a new attribute called Credit amount per duration attribute.