HW 2

Jeff Gould

9/8/2020

1 - Read Chapter 3

2 - You are given a set of m objects that is divided into K groups, where the i^{th} group is of size m_i. If the goal is to obtain a sample of size $n \leq m$, what is the difference between the following sampling schemes?

- We randomly select $n \times m_i/m$ elements from each group
- We randomly select n elements from the data set, without regards for the group to which an object belongs

Consider this simple example: $K = 10$, $m = 100$, $m_i = 10 \ \forall i \in [1, 2, \ldots, 10]$, $n = 30$

If we randomly select $n \times m_i/m$ elements from each group, then we will look at each group and sample exactly 3 objects from each group ($n \times m_i/m = 30 * 10/100 = 3$). In a more general sense, our sample will have a representation of each group proportional to that groups representation of the entire data set. This is **stratified random sampling**.

If we randomly select $n = 30$ elements out of the entire data set, then there will likely not be representation for each group directly proportional to their represnetation of the whole data set. Using the example above, we could select 6 elements from group 1, but not even select a single element from group 3, so long as we sampled 30 total elements. This is **simple random sampling**.

3 - Experiment with different types of binning on the glass data (installed with WEKA or available in R via the mlbench package). Try equal width and equal depth binning. Try various numbers of bins (at least three). Use OneR on the original data and on your binned data. Compare the results and write a brief description of what you observe.

First - test `OneR` on the raw data:

```
#install.packages("mlbench")
library(mlbench)
library(OneR)
data(Glass)

OneR(Glass)
```

```
## Warning in OneR.data.frame(Glass): data contains unused factor levels
```

```
##
## Call:
```

```
## OneR.data.frame(x = Glass)
##
## Rules:
## If Al = (0.287,0.932] then Type = 1
## If Al = (0.932,1.57]  then Type = 1
## If Al = (1.57,2.22]   then Type = 2
## If Al = (2.22,2.86]   then Type = 7
## If Al = (2.86,3.5]    then Type = 5
##
## Accuracy:
## 100 of 214 instances classified correctly (46.73%)
```

Using `OneR` on the raw data, we see that by dividing the data based on Aluminum content, we are able to properly classify 46.73% of the data correctly. This rule divides the glass into 5 bins, two of which are classified as type 1.

Now bin the data into bins of equal widths, calculate `OneR` for bins of 3, 5, 7, 9, 11, and 13, and show an example plot for 5 bins:

```
equalWidthOneR <- function(bins){

  glassBinEqualWidth <- data.frame(apply(Glass[,1:9], 2, cut, include.lowest = T, breaks = bins))
  glassBinEqualWidth$Type = Glass$Type
  print(glue::glue("OneR Results using {bins} bins of equal width: "))
  return(OneR(glassBinEqualWidth))
}
for (i in c(3,5,7,9, 11, 13)) {

  output <- equalWidthOneR(i)
  print(output)
}
```

```
## OneR Results using 3 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualWidth)
##
## Rules:
## If Al = (1.36,2.43]  then Type = 2
## If Al = (2.43,3.5]   then Type = 7
## If Al = [0.287,1.36] then Type = 1
##
## Accuracy:
## 113 of 214 instances classified correctly (52.8%)
##
## OneR Results using 5 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualWidth)
##
## Rules:
## If Al = (0.932,1.57]  then Type = 1
## If Al = (1.57,2.22]   then Type = 2
## If Al = (2.22,2.86]   then Type = 7
```

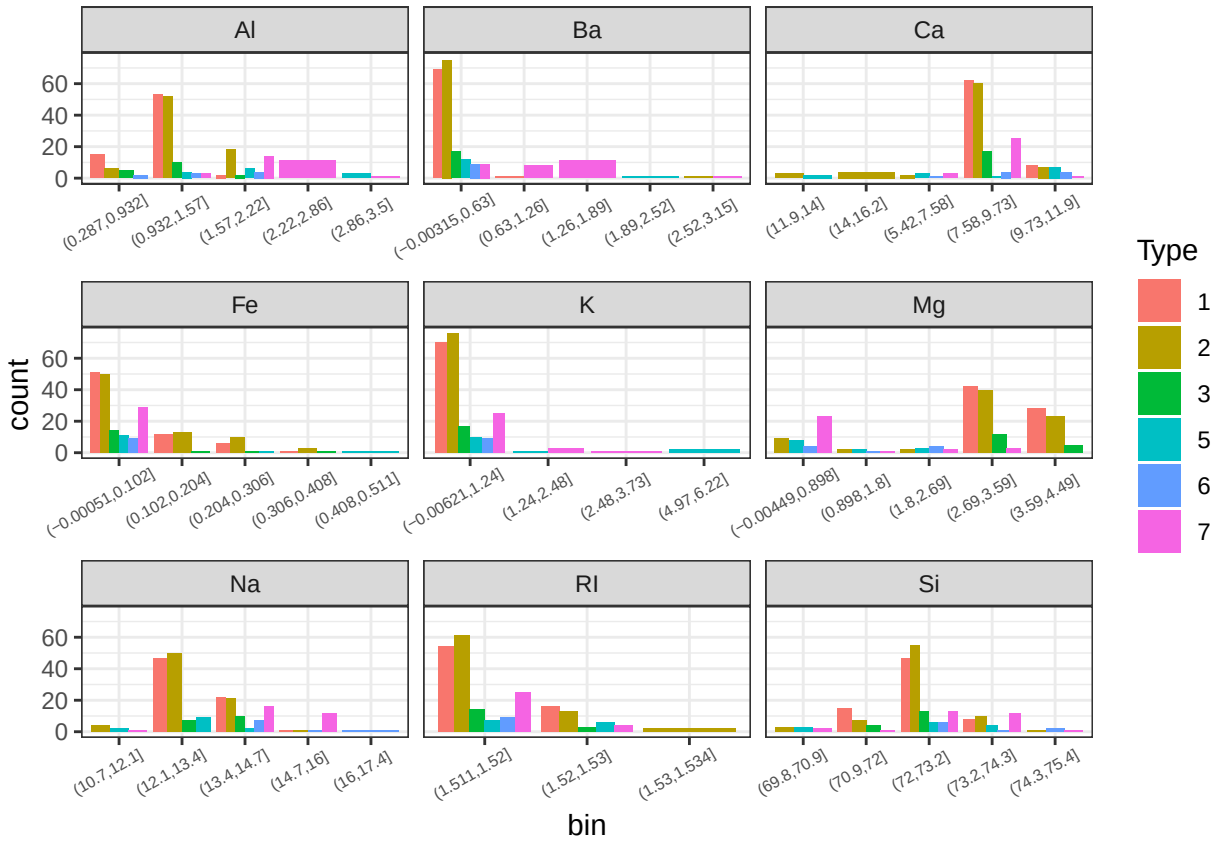
```
## If Al = (2.86,3.5]    then Type = 5
## If Al = [0.287,0.932] then Type = 1
##
## Accuracy:
## 101 of 214 instances classified correctly (47.2%)
##
## OneR Results using 7 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualWidth)
##
## Rules:
## If Mg = (0.641,1.28]     then Type = 2
## If Mg = (1.28,1.92]      then Type = 5
## If Mg = (1.92,2.57]      then Type = 6
## If Mg = (2.57,3.21]      then Type = 2
## If Mg = (3.21,3.85]      then Type = 1
## If Mg = (3.85,4.49]      then Type = 2
## If Mg = [-0.00449,0.641] then Type = 7
##
## Accuracy:
## 112 of 214 instances classified correctly (52.34%)
##
## OneR Results using 9 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualWidth)
##
## Rules:
## If Al = (0.647,1]     then Type = 1
## If Al = (1,1.36]      then Type = 1
## If Al = (1.36,1.72]   then Type = 2
## If Al = (1.72,2.07]   then Type = 7
## If Al = (2.07,2.43]   then Type = 7
## If Al = (2.43,2.79]   then Type = 7
## If Al = (2.79,3.14]   then Type = 5
## If Al = (3.14,3.5]    then Type = 5
## If Al = [0.287,0.647] then Type = 1
##
## Accuracy:
## 124 of 214 instances classified correctly (57.94%)
##
## OneR Results using 11 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualWidth)
##
## Rules:
## If Al = (0.582,0.874] then Type = 1
## If Al = (0.874,1.17]  then Type = 1
## If Al = (1.17,1.46]   then Type = 1
## If Al = (1.46,1.75]   then Type = 2
## If Al = (1.75,2.04]   then Type = 7
## If Al = (2.04,2.33]   then Type = 7
```

```
## If Al = (2.33,2.62]   then Type = 7
## If Al = (2.62,2.92]   then Type = 7
## If Al = (2.92,3.21]   then Type = 5
## If Al = (3.21,3.5]    then Type = 5
## If Al = [0.287,0.582] then Type = 1
##
## Accuracy:
## 124 of 214 instances classified correctly (57.94%)
##
## OneR Results using 13 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualWidth)
##
## Rules:
## If Al = (0.537,0.784] then Type = 2
## If Al = (0.784,1.03]  then Type = 1
## If Al = (1.03,1.28]   then Type = 1
## If Al = (1.28,1.52]   then Type = 1
## If Al = (1.52,1.77]   then Type = 2
## If Al = (1.77,2.02]   then Type = 7
## If Al = (2.02,2.27]   then Type = 7
## If Al = (2.27,2.51]   then Type = 7
## If Al = (2.51,2.76]   then Type = 7
## If Al = (2.76,3.01]   then Type = 7
## If Al = (3.01,3.25]   then Type = 5
## If Al = (3.25,3.5]    then Type = 5
## If Al = [0.287,0.537] then Type = 1
##
## Accuracy:
## 117 of 214 instances classified correctly (54.67%)
```

```
# glassBins5EqualWidth <- data.frame(apply(Glass[,1:9], 2, cut, include.lowest = T, breaks = 5))
# glassBins5EqualWidth$Type = Glass$Type
# glassBins5EqualWidth <- glassBins5EqualWidth %>%
#   pivot_longer(cols = RI:Fe, names_to = "element", values_to = "bin")
#
# ggplot(glassBins5EqualWidth, aes(x = bin, fill = Type)) +
#   geom_bar() +
#   facet_wrap(vars(element), scales = "free_x") +
#   theme_bw()

glassBins5EqualWidth <- data.frame(apply(Glass[,1:9], 2, function(x){
    bin(x, method = "length")
  }))
glassBins5EqualWidth$Type = Glass$Type
glassBins5EqualWidth <- glassBins5EqualWidth %>%
  pivot_longer(cols = RI:Fe, names_to = "element", values_to = "bin")

ggplot(glassBins5EqualWidth, aes(x = bin, fill = Type)) +
  geom_bar(position = "dodge") +
  facet_wrap(vars(element), scales = "free_x") +
  theme_bw() +
  theme(axis.text.x = element_text(size = 6, angle = 30, vjust = 0.8, hjust = 0.7))
```



Interestingly, we initially find higher accuracy when dividing the data into 3 bins than 5, achieving 52.8% accuracy. At 7 bins, optimal accuracy is achieved ruling based on Magnesium content, but no improvement relative to 3 bins. When we jump up to 9 bins however, we now reach 57.94% accuracy, classifying on Aluminum again. We see no further improvement when we jump to 11 bins. Accuracy decreases when we jump to 13 bins, suggesting we might be overfitting.

Now bin the data into bins of equal depth, calculate the `OneR` rule for bins of 3, 5, 7, 9, 11, 13. Additionally, show an example plot for 5 bins.

```
equalDepthOneR <- function(bins){

  glassBinEqualDepth <- data.frame(apply(Glass[,1:9], 2, function(x){
    cut(x,include.lowest = T,breaks = unique(quantile(x, probs = 0:bins/bins)))
  }
    )
    )
  glassBinEqualDepth$Type = Glass$Type
  print(glue::glue("OneR Results using {bins} bins of equal width: "))
  return(OneR(glassBinEqualDepth))
}
for (i in c(3,5,7,9,11, 13)) {
  output <- equalDepthOneR(i)
  print(output)
}
```

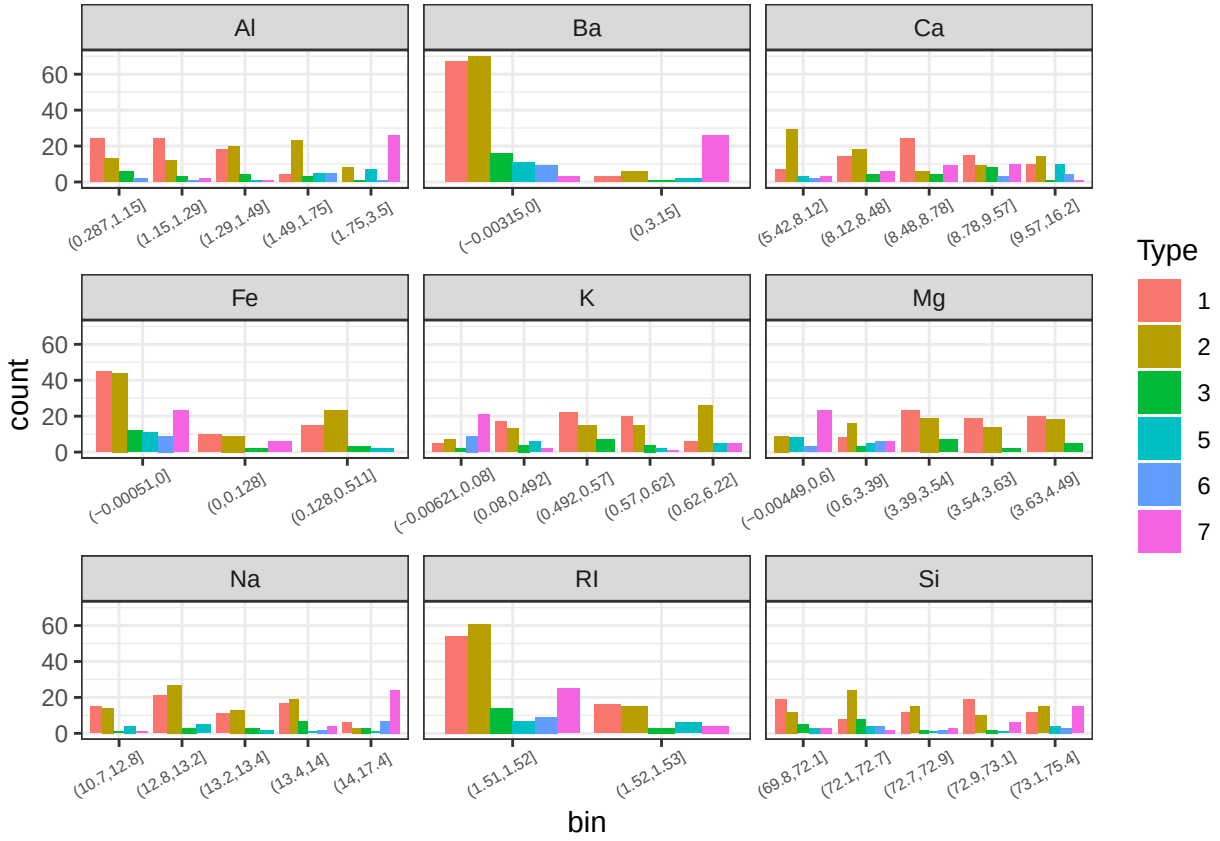
```
## OneR Results using 3 bins of equal width:
```

```
##
## Call:
## OneR.data.frame(x = glassBinEqualDepth)
##
## Rules:
## If RI = (1.517,1.518] then Type = 1
## If RI = (1.518,1.534] then Type = 1
## If RI = [1.511,1.517] then Type = 2
##
## Accuracy:
## 98 of 214 instances classified correctly (45.79%)
##
## OneR Results using 5 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualDepth)
##
## Rules:
## If Al = (1.15,1.29] then Type = 1
## If Al = (1.29,1.49] then Type = 2
## If Al = (1.49,1.75] then Type = 2
## If Al = (1.75,3.5]  then Type = 7
## If Al = [0.29,1.15] then Type = 1
##
## Accuracy:
## 117 of 214 instances classified correctly (54.67%)
##
## OneR Results using 7 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualDepth)
##
## Rules:
## If Al = (1.05,1.22] then Type = 1
## If Al = (1.22,1.31] then Type = 1
## If Al = (1.31,1.45] then Type = 1
## If Al = (1.45,1.57] then Type = 2
## If Al = (1.57,1.88] then Type = 2
## If Al = (1.88,3.5]  then Type = 7
## If Al = [0.29,1.05] then Type = 1
##
## Accuracy:
## 120 of 214 instances classified correctly (56.07%)
##
## OneR Results using 9 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualDepth)
##
## Rules:
## If Al = (0.887,1.17] then Type = 1
## If Al = (1.17,1.25]  then Type = 1
## If Al = (1.25,1.33]  then Type = 1
## If Al = (1.33,1.43]  then Type = 1
```

```
## If Al = (1.43,1.54]  then Type = 2
## If Al = (1.54,1.68]  then Type = 2
## If Al = (1.68,2.01]  then Type = 7
## If Al = (2.01,3.5]   then Type = 7
## If Al = [0.29,0.887] then Type = 1
##
## Accuracy:
## 121 of 214 instances classified correctly (56.54%)
##
## OneR Results using 11 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualDepth)
##
## Rules:
## If Al = (0.83,1.12] then Type = 1
## If Al = (1.12,1.2]  then Type = 1
## If Al = (1.2,1.27]  then Type = 1
## If Al = (1.27,1.33] then Type = 1
## If Al = (1.33,1.43] then Type = 1
## If Al = (1.43,1.53] then Type = 2
## If Al = (1.53,1.61] then Type = 2
## If Al = (1.61,1.79] then Type = 2
## If Al = (1.79,2.09] then Type = 7
## If Al = (2.09,3.5]  then Type = 7
## If Al = [0.29,0.83] then Type = 1
##
## Accuracy:
## 124 of 214 instances classified correctly (57.94%)
##
## OneR Results using 13 bins of equal width:
##
## Call:
## OneR.data.frame(x = glassBinEqualDepth)
##
## Rules:
## If Al = (0.784,1.08] then Type = 1
## If Al = (1.08,1.18]  then Type = 1
## If Al = (1.18,1.23]  then Type = 1
## If Al = (1.23,1.29]  then Type = 2
## If Al = (1.29,1.34]  then Type = 1
## If Al = (1.34,1.42]  then Type = 1
## If Al = (1.42,1.51]  then Type = 2
## If Al = (1.51,1.56]  then Type = 2
## If Al = (1.56,1.66]  then Type = 2
## If Al = (1.66,1.86]  then Type = 2
## If Al = (1.86,2.15]  then Type = 7
## If Al = (2.15,3.5]   then Type = 7
## If Al = [0.29,0.784] then Type = 1
##
## Accuracy:
## 126 of 214 instances classified correctly (58.88%)
```

```r
glassBins5EqualDepth <- data.frame(apply(Glass[,1:9], 2, function(x){
    bin(x, method = "content")
  }))
glassBins5EqualDepth$Type = Glass$Type
glassBins5EqualDepth <- glassBins5EqualDepth %>%
  pivot_longer(cols = RI:Fe, names_to = "element", values_to = "bin")

ggplot(glassBins5EqualDepth, aes(x = bin, fill = Type)) +
  geom_bar(position = "dodge") +
  facet_wrap(vars(element), scales = "free_x") +
  theme_bw() +
  theme(axis.text.x = element_text(size = 6, angle = 30, vjust = 0.8, hjust = 0.7))
```



Binning on eqaul depth instead of equal width tends to have higher accuracy. At five bins, classifying on Aluminum, 54.67% accuracy is obtained, compared to 46.73% of five bins. Accuracy gradually increases as the number of bins increases, achieving highest accuracy at 13 bins, where 58.88% of the glass was correctly classified