DMIT Vignette: OCCAM Data-Prep with R/Looking at Census-Income Data

Dale Frakes

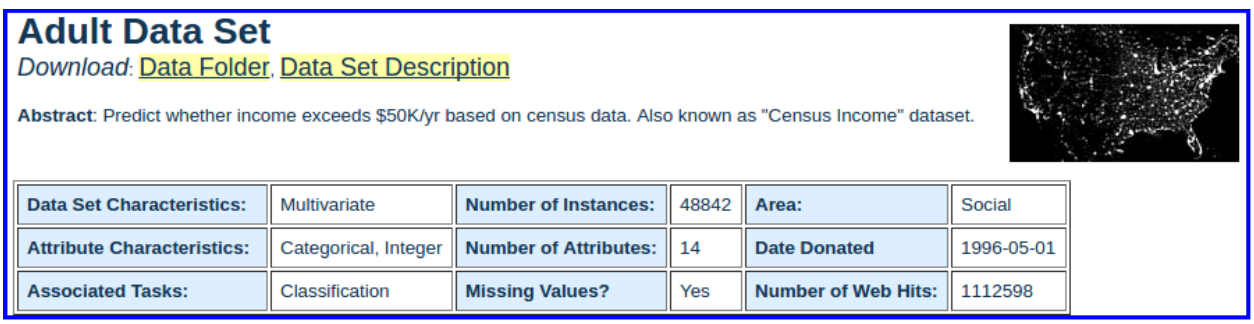
March 20, 2018

Introduction

The goal of this vignette is to demonstrate taking a data set and preparing it for use with OCCAM using R. It demonstrates loading data, then using a variety of tools in R for working with categorical data (such as adjusting the levels, and binning continuous variables) then uses a function that will render an OCCAM data input file from a dataframe of factors (categorical variables).

Datasets

We'll be selecting a data set from the UCIML Repository. Specifically we'll be using the Adult dataset (http://archive.ics.uci.edu/ml/datasets/Adult). It's a collection of some census data that's been used in many publications to show the capability of various algorithms to be able to predict if the income of a case will be above $50,000. It's appealing for this project because it has several categorical variables and a few continuous ones.



Adult Data Set

Download: Data Folder, Data Set Description

Abstract: Predict whether income exceeds $50K/yr based on census data. Also known as "Census Income" dataset.

Data Set Characteristics:	Multivariate	**Number of Instances:**	48842	**Area:**	Social
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	14	**Date Donated**	1996-05-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	1112598

Preparing the Data

The first step, of course, is loading the data into R so we can analyze it. The data from UCIML usually comes without headings/variable-names and has a separate file containing those. The following loads the data into a data frame called `ad` and names the columns appropriately.

```r
# read the main datafile, keeping default of strings as factors, remove any whitespace around strings
ad <- read.csv("./data/adult.data.txt", header=FALSE, strip.white = TRUE)

# read in the file with column names
an <- read.csv("./data/adult.names.txt", header=FALSE, sep=";", stringsAsFactors=FALSE)

# hard-coding that we are only interested in the rows 97-107 (the rest is just explanation)
col_names <- sapply(strsplit(as.vector(an[94:107,]),":"), "[", 1)

# remove hyphens from column names
col_names <- gsub("-", "", col_names)
```

```r
col_names <- c(col_names, "income")

colnames(ad) <- col_names
```

Note that this bit of code is somewhat hard-coded for this data set and is not generally applicable.

And now that the data is loaded, let's take a couple looks at it, just to see what's in there. One of the things I really like about R is that it makes it pretty simple to inspect a dataset. The first thing I usually like to use is the `str` (structure) command. It gives some great details about a variable, or particularly a data frame, including data types and typical values.

```r
str(ad)
```

```
'data.frame':   32561 obs. of  15 variables:
 $ age          : int  39 50 38 53 28 37 49 52 31 42 ...
 $ workclass    : Factor w/ 9 levels "?","Federal-gov",..: 8 7 5 5 5 5 5 7 5 5 ...
 $ fnlwgt       : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
 $ education    : Factor w/ 16 levels "10th","11th",..: 10 10 12 2 10 13 7 12 13 10 ...
 $ educationnum : int  13 13 9 7 13 14 5 9 14 13 ...
 $ maritalstatus: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
 $ occupation   : Factor w/ 15 levels "?","Adm-clerical",..: 2 5 7 7 11 5 9 5 11 5 ...
 $ relationship : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
 $ race         : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
 $ sex          : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
 $ capitalgain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
 $ capitalloss  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ hoursperweek : int  40 13 40 40 40 40 16 45 50 40 ...
 $ nativecountry: Factor w/ 42 levels "?","Cambodia",..: 40 40 40 40 6 40 24 40 40 40 ...
 $ income       : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

At this point it might also be helpful to look at the variable descriptions from the `names` file:

```r
cat(an[94:107,], sep ="\n")
```

```
age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Neve
fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12t
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent,
occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-
relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex: Female, Male.
capital-gain: continuous.
capital-loss: continuous.
hours-per-week: continuous.
native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-et
```

Along with the structure of the data, it's helpful to get a `summary` of the data values within the data.

```r
summary(ad)
```

```
      age                   workclass         fnlwgt
 Min.   :17.00   Private         :22696   Min.   :  12285
 1st Qu.:28.00   Self-emp-not-inc: 2541   1st Qu.: 117827
 Median :37.00   Local-gov       : 2093   Median : 178356
```

```
 Mean   :38.58   ?                : 1836   Mean   : 189778  
 3rd Qu.:48.00   State-gov        : 1298   3rd Qu.: 237051  
 Max.   :90.00   Self-emp-inc     : 1116   Max.   :1484705  
                 (Other)          :  981                    
        education      educationnum                 maritalstatus  
 HS-grad     :10501   Min.   : 1.00   Divorced             : 4443  
 Some-college: 7291   1st Qu.: 9.00   Married-AF-spouse    :   23  
 Bachelors   : 5355   Median :10.00   Married-civ-spouse   :14976  
 Masters     : 1723   Mean   :10.08   Married-spouse-absent:  418  
 Assoc-voc   : 1382   3rd Qu.:12.00   Never-married        :10683  
 11th        : 1175   Max.   :16.00   Separated            : 1025  
 (Other)     : 5134                   Widowed              :  993  
           occupation           relationship                    race      
 Prof-specialty :4140   Husband       :13193   Amer-Indian-Eskimo:  311  
 Craft-repair   :4099   Not-in-family : 8305   Asian-Pac-Islander: 1039  
 Exec-managerial:4066   Other-relative:  981   Black             : 3124  
 Adm-clerical   :3770   Own-child     : 5068   Other             :  271  
 Sales          :3650   Unmarried     : 3446   White             :27816  
 Other-service  :3295   Wife          : 1568                             
 (Other)        :9541                                                    
     sex         capitalgain     capitalloss      hoursperweek  
 Female:10771   Min.   :    0   Min.   :   0.0   Min.   : 1.00  
 Male  :21790   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00  
                Median :    0   Median :   0.0   Median :40.00  
                Mean   : 1078   Mean   :  87.3   Mean   :40.44  
                3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00  
                Max.   :99999   Max.   :4356.0   Max.   :99.00  

       nativecountry     income     
 United-States:29170   <=50K:24720  
 Mexico       :  643   >50K : 7841  
 ?            :  583                
 Philippines  :  198                
 Germany      :  137                
 Canada       :  121                
 (Other)      : 1709                
```

Fixing "Education"

"Education" appears in two different variables, one continuous, and one categorical (factor). I'm not sure we need both, so let's take a look at the values they take together.

```r
ed <- unique(ad[c("educationnum","education")])

ed[order(ed$educationnum),]
```

```
    educationnum    education
225            1    Preschool
161            2      1st-4th
57             3      5th-6th
16             4      7th-8th
7              5          9th
78             6         10th
4              7         11th
```

```
416             8        12th
3               9     HS-grad
11             10 Some-college
15             11    Assoc-voc
14             12   Assoc-acdm
1              13    Bachelors
6              14      Masters
53             15  Prof-school
21             16    Doctorate
```

As we can see from that, it appears `educationnum` is merely a coding of `education`. Since R already encodes `education` as a factor, we can probably remove it. But before do that, we'll want to see how `education` has been encoded as a factor. One advantage we get from `educationnum` is that the amount of education is ordered, which may make it easier to re-bin into fewer bins/factors later in our analysis. For example, we could combine the values of `educationum` from 1-12 as a single factor, "k-12". The default factorization of `education` as it was read from the data file may end up with a less sensible order.

We can investigate how `education` was set up as a factor this way. `str` tell us the following:

```
str(ad$education)
```

```
 Factor w/ 16 levels "10th","11th",..: 10 10 12 2 10 13 7 12 13 10 ...
```

Which makes it look like the order isn't sensible, but is probably alphabetical, which may not be as helpful. We can confirm using `levels`, and then asking for the levels 1-5.

```
levels(ad$education)
```

```
 [1] "10th"         "11th"         "12th"         "1st-4th"
 [5] "5th-6th"      "7th-8th"      "9th"          "Assoc-acdm"
 [9] "Assoc-voc"    "Bachelors"    "Doctorate"    "HS-grad"
[13] "Masters"      "Preschool"    "Prof-school"  "Some-college"
```

```
levels(ad$education)[1:5]
```

```
[1] "10th"    "11th"    "12th"    "1st-4th" "5th-6th"
```

Having confirmed that the factor coding for `education` makes less sense than `educationnum`, what we'd like to do is re-factor `education` with the order provided by `educationnum`, then delete `educationnum`. Before we do that, let's take a look at the breakdown of `education` to make sure we end up with a correct re-coding.

```
summary(ad$education)
```

```
        10th         11th         12th      1st-4th      5th-6th
         933         1175          433          168          333
     7th-8th          9th   Assoc-acdm    Assoc-voc    Bachelors
         646          514         1067         1382         5355
   Doctorate      HS-grad      Masters    Preschool  Prof-school
         413        10501         1723           51          576
Some-college
        7291
```

Let's use `ed` from above to set up the new factorization or `levels` for `education`. Then use that to re-factor the `education` variable.

```
ed <- ed[order(ed$educationnum),]
levels(factor(ed$educationnum, levels=ed$education))
```

```
 [1] "Preschool"    "1st-4th"      "5th-6th"      "7th-8th"
 [5] "9th"          "10th"         "11th"         "12th"
```

```
 [9] "HS-grad"      "Some-college" "Assoc-voc"    "Assoc-acdm"  
[13] "Bachelors"    "Masters"      "Prof-school"  "Doctorate"   
```

```r
ad$education <- factor(ad$education, levels = as.vector(ed$education))

summary(ad$education)
```

```
   Preschool      1st-4th      5th-6th      7th-8th          9th 
          51          168          333          646          514 
        10th         11th         12th      HS-grad Some-college 
         933         1175          433        10501         7291 
   Assoc-voc   Assoc-acdm    Bachelors      Masters  Prof-school 
        1382         1067         5355         1723          576 
   Doctorate 
         413 
```

And with that summary, we can see that we've recoded `education` to have the correct order, and a quick comparison of the summary numbers shows that the values match after that transformation.

We can now remove `educationnum` from the dataset. We can also get rid of the `ed` data frame since we won't use it any more.

```r
ad <- ad[, !(colnames(ad) %in% c("educationnum"))]
rm(ed)

str(ad)
```

```
'data.frame':   32561 obs. of  14 variables:
 $ age          : int  39 50 38 53 28 37 49 52 31 42 ...
 $ workclass    : Factor w/ 9 levels "?","Federal-gov",..: 8 7 5 5 5 5 5 7 5 5 ...
 $ fnlwgt       : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
 $ education    : Factor w/ 16 levels "Preschool","1st-4th",..: 13 13 9 7 13 14 5 9 14 13 ...
 $ maritalstatus: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
 $ occupation   : Factor w/ 15 levels "?","Adm-clerical",..: 2 5 7 7 11 5 9 5 11 5 ...
 $ relationship : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
 $ race         : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
 $ sex          : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
 $ capitalgain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
 $ capitalloss  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ hoursperweek : int  40 13 40 40 40 40 16 45 50 40 ...
 $ nativecountry: Factor w/ 42 levels "?","Cambodia",..: 40 40 40 40 6 40 24 40 40 40 ...
 $ income       : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

Removing `fnlwgt`

`fnlwgt` is some kind of weighting in the dataset that has some detailed explanation about it, but I still can't quite make sense of how to use it. So for purposes of this analysis, I'm removing the variable.

```r
ad <- ad[, !(colnames(ad) %in% c("fnlwgt"))]
```

Now for the Binning

In this section we'll deal with the variables and try to feature different ways variables can be binned in R using various libraries.

Most of the variables are already categorical because they are factors. Let's take a look at the variables that are not factors and decide how to handle them.

```
str(ad[,!sapply(ad, is.factor)])
```

```
'data.frame':   32561 obs. of  4 variables:
 $ age         : int  39 50 38 53 28 37 49 52 31 42 ...
 $ capitalgain : int  2174 0 0 0 0 0 0 0 14084 5178 ...
 $ capitalloss : int  0 0 0 0 0 0 0 0 0 0 ...
 $ hoursperweek: int  40 13 40 40 40 40 16 45 50 40 ...
```

```
summary(ad[,!sapply(ad, is.factor)])
```

```
      age         capitalgain     capitalloss      hoursperweek
 Min.   :17.00   Min.   :    0   Min.   :   0.0   Min.   : 1.00
 1st Qu.:28.00   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00
 Median :37.00   Median :    0   Median :   0.0   Median :40.00
 Mean   :38.58   Mean   : 1078   Mean   :  87.3   Mean   :40.44
 3rd Qu.:48.00   3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00
 Max.   :90.00   Max.   :99999   Max.   :4356.0   Max.   :99.00
```

Age

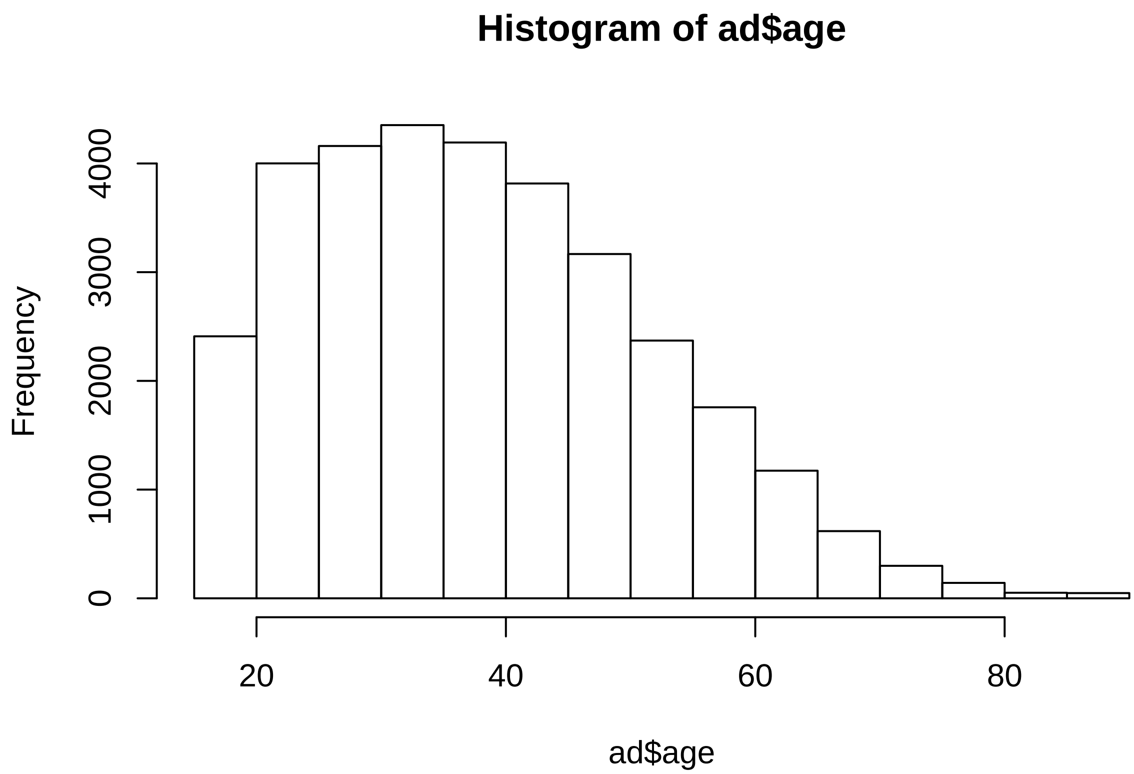
The `summary` data above help see what values `age` takes. But it might be even more helpful to see more detail. It's an integer from 17 to 90, but it might be helpful to see the distribution.

One way is to use the `table` command. Another is to do an actual histogram as a chart.

```
table(ad$age)
```

```

 17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
395 550 712 753 720 765 877 798 841 785 835 867 813 861 888 828 875 886
 35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52
876 898 858 827 816 794 808 780 770 724 734 737 708 543 577 602 595 478
 53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70
464 415 419 366 358 366 355 312 300 258 230 208 178 150 151 120 108  89
 71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88
 72  67  64  51  45  46  29  23  22  22  20  12   6  10   3   1   1   3
 90
 43
```

```
hist(x = ad$age)
```



Age, of course, is one of those variables that's often binned in a specific way. For the Census, this is often broken down in a hierarchy. The top level is "Under 65 Years" and "64 and Over".

Under that are: 15-24, 25-34, 35-44, 45-54, 55-64, 65-74, 75+

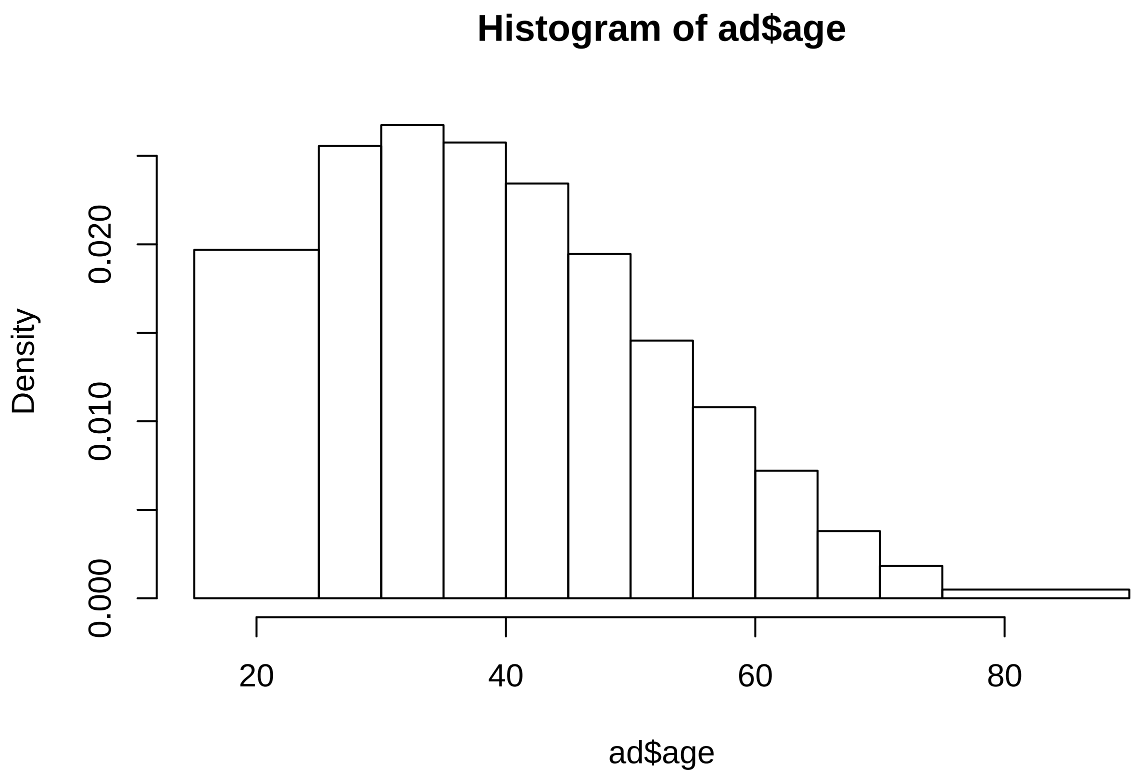
And lowest is: 15-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59, 60-64, 65-69, 70-74, 75+

Let's start by using the lowest level of age hierarchy, since it's easy to regroup them later into fewer bins.

```
age_breaks <- seq(from = 25, to = 75, by = 5)    # all but first break are 5 years apart

age_breaks <- c(15, age_breaks)                  # insert the first value at the beginning

age_breaks <- c(age_breaks, max(ad$age))         # make sure the highest value is contained in the break

# confirm that our age-breaks look like what we've described above
age_breaks
```

```
 [1] 15 25 30 35 40 45 50 55 60 65 70 75 90
```

```
# and now plot a histogram using these bins
hist(x = ad$age, breaks = age_breaks)
```



`cut` can be used with the same breaks as `hist` to bin data. `hist` can be called with a `plot = FALSE` option to then get the counts or density. A summary of `cut` with the same breaks reveals the same counts as shown in the histogram.

```
hist(x = ad$age, breaks = age_breaks, plot = FALSE)$counts
```

```
 [1] 6411 4161 4353 4193 3816 3167 2371 1757 1174  618  299  241
```

```
summary(cut(x=ad$age, breaks = age_breaks))
```

```
(15,25] (25,30] (30,35] (35,40] (40,45] (45,50] (50,55] (55,60] (60,65]
   6411    4161    4353    4193    3816    3167    2371    1757    1174
(65,70] (70,75] (75,90]
    618     299     241
```

And now we're ready to replace the `age` variable with its discretized (factor) version. Note that this permanently changes the variable.

```
ad$age <- cut(x=ad$age, breaks = age_breaks)
str(ad$age)
```

```
 Factor w/ 12 levels "(15,25]","(25,30]",..: 4 6 4 7 2 4 6 7 3 5 ...
```

```
summary(ad$age)
```

```
(15,25] (25,30] (30,35] (35,40] (40,45] (45,50] (50,55] (55,60] (60,65]
   6411    4161    4353    4193    3816    3167    2371    1757    1174
(65,70] (70,75] (75,90]
    618     299     241
```

Before moving on, let's look at how we would aggregate/group the levels of the age factor we just created. One easy way is to simply assign new `levels` to the variable, duplicating the new level in the position of the lower levels it encompasses. In this case, it should go from 12 levels to 7.

```r
# Old: 15-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59, 60-64, 65-69, 70-74, 75+
# New: 15-24, 25-34,        35-44,        45-54,        55-64,        65-74,        75+

# make a new vector with the new factor labels.  In the data above the non-end factors are grouped by 2
# so the new vector states the new factor twice for each of those
new_age_breaks <- c("15-24",
                    "25-34", "25-34",   # encompases 25-29 and 30-34
                    "35-44", "35-44",
                    "45-54", "45-54",
                    "55-64", "55-64",
                    "65-74", "65-74",
                    "75+")

# make a copy of the age variable because we don't want to make this change to our data (just yet)
age_tmp <- ad$age

levels(age_tmp) <- new_age_breaks

str(age_tmp)
```

```
 Factor w/ 7 levels "15-24","25-34",..: 3 4 3 4 2 3 4 4 2 3 ...
```

```r
summary(age_tmp)
```

```
15-24 25-34 35-44 45-54 55-64 65-74   75+
 6411  8514  8009  5538  2931   917   241
```

A similar procedure would be followed to apply the highest level of hierarchy with just 2 levels.

Note that OCCAM offers the ability to re-bin a variable by combining existing levels (states). Also note the the above re-binning only works if new levels are a strict combination of the old levels. If you want new break-points (e.g. age 15-32), it's necessary to apply the new bins to the original numeric data.

Capital-Gain

`capitalgain` is a variable that will be more challenging/interesting than `age`. It is not evenly distributed and has a lot of sparseness. This will offer an opportunity to explore other binning options in R.

```r
summary(ad$capitalgain)
```

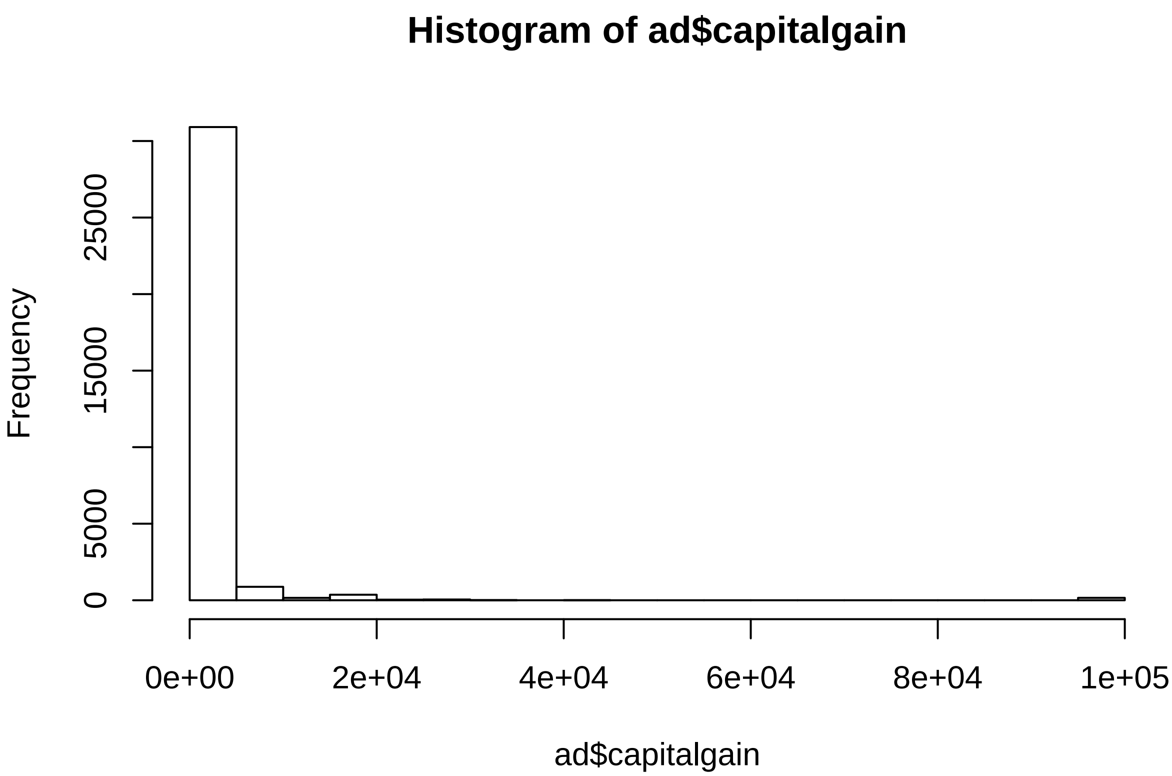
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0    1078       0   99999
```

```r
table(ad$capitalgain)
```

```

    0   114   401   594   914   991  1055  1086  1111  1151  1173  1409
29849     6     2    34     8     5    25     4     1     8     3     7
 1424  1455  1471  1506  1639  1797  1831  1848  2009  2036  2050  2062
    3     1     7    15     1     7     7     6     3     4     5     2
 2105  2174  2176  2202  2228  2290  2329  2346  2354  2387  2407  2414
    9    48    23    16     5     5     6     6    11     1    19     8
 2463  2538  2580  2597  2635  2653  2829  2885  2907  2936  2961  2964
   11     1    12    20    11     5    31    24    11     3     3     9
 2977  2993  3103  3137  3273  3325  3411  3418  3432  3456  3464  3471
    8     2    97    37     6    53    24     5     4     2    23     8
 3674  3781  3818  3887  3908  3942  4064  4101  4386  4416  4508  4650
```

```
   14    12     7     6    32    14    42    20    70    12    12    41
 4687  4787  4865  4931  4934  5013  5060  5178  5455  5556  5721  6097
    3    23    17     1     7    69     1    97    11     5     3     1
 6360  6418  6497  6514  6723  6767  6849  7298  7430  7443  7688  7896
    3     9    11     5     2     5    27   246     9     5   284     3
 7978  8614  9386  9562 10520 10566 10605 11678 13550 14084 14344 15020
    1    55    22     4    43     6    12     2    27    41    26     5
15024 15831 18481 20051 22040 25124 25236 27828 34095 41310 99999
  347     6     2    37     1     4    11    34     5     2   159
```

```r
hist(ad$capitalgain)
```

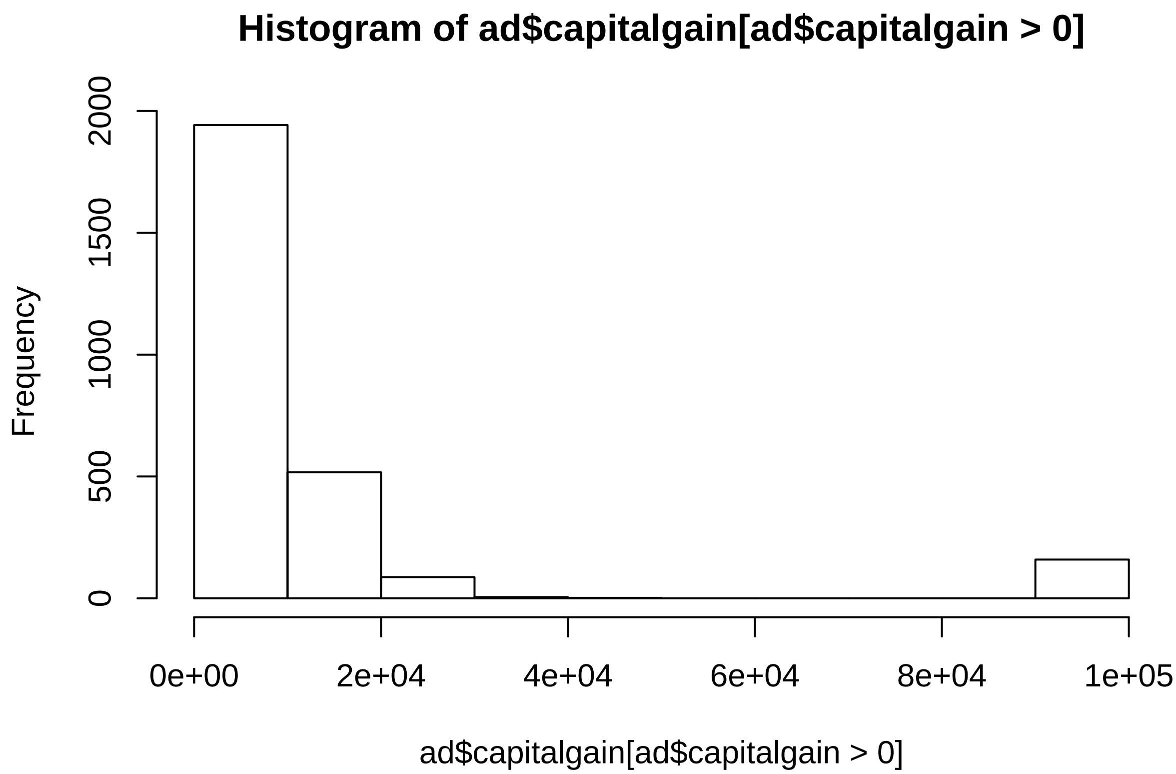


As we can see, a vast majority of cases have 0 capital gains, so maybe it was interesting to exclude that value and see what's in there.

```r
summary(ad$capitalgain[ad$capitalgain>0])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    114    3411    7298   12939   14084   99999 
```

```r
hist(ad$capitalgain[ad$capitalgain>0])
```



Considering capital-gains are something most people don't have, maybe the obvious binning is binary one, "doesn't have", "has capital gains". For this we don't need to use `cut`. In this case, we'll use `levels` with a logical function applied to the values in the column.

```
summary(as.factor((ad$capitalgain > 0)))
```

```
FALSE  TRUE
29849  2712
```

Now, it might be interesting to see what a contingency table of `capitalgain` and `income` looks like.

```
tmp <- data.frame(capitalgain = as.factor((ad$capitalgain > 0)))
tmp$income <- ad$income
table(tmp)
```

```
           income
capitalgain <=50K  >50K
      FALSE 23685  6164
      TRUE   1035  1677
```

It looks like *not* having capital gains could possibly be a good predictor of income, but having capital gains doesn't seem to be very discriminatory. This is something we can explore in a state-based model search.

That said, there are other binning strategies we can pursue with this variable. For these, we'll use functions from the `OneR` package. `OneR` is primarily a library that create 1-level decision trees with each variable to see which "One" variable model is the best predictor.

It offers a function called `bin`, which can use various methods for binning, such as equal-width binning, equal-content binning, and cluster binning.

In DMIT we've discussed binning by 12 because it can easily be re-grouped into 6, 4, 3, or 2 bins.

```
table(bin(data = ad$capitalgain, method = "length", nbins = 12))
```

```
  (-100,8.33e+03] (8.33e+03,1.67e+04]  (1.67e+04,2.5e+04]
            31710                 596                  40
(2.5e+04,3.33e+04] (3.33e+04,4.17e+04]    (4.17e+04,5e+04]
               49                   7                   0
  (5e+04,5.83e+04] (5.83e+04,6.67e+04]  (6.67e+04,7.5e+04]
                0                   0                   0
(7.5e+04,8.33e+04] (8.33e+04,9.17e+04]    (9.17e+04,1e+05]
                0                   0                 159
```

In this case, 12 equal-width bins (`length`), we get quite a few 0's, which can cause problems with the results in OCCAM.

Another approach is to make 12 bins based on `content`, trying to make each bin as equal in size as possible (this will have problems due to the huge relative number of 0s).

```r
table(bin(data = ad$capitalgain, method = "content", nbins = 12))
```

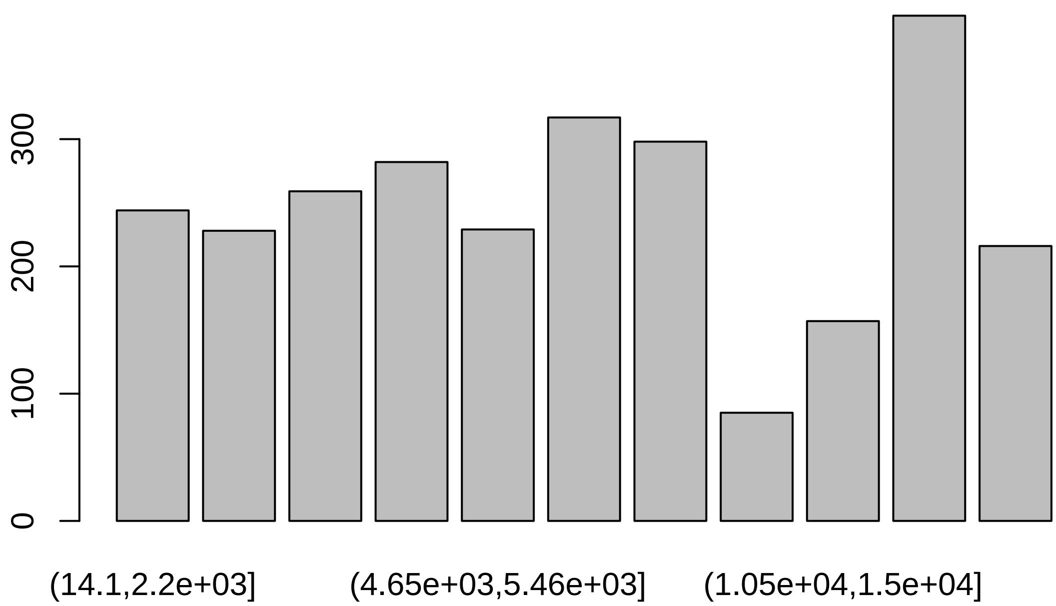
```
(-100,0] (0,1e+05]
   29849      2712
```

The algorithm can't easily deal with the 0 problem, so let's look at binning everything except the 0s.

```r
# 0 is interesting and large
table(bin(data = ad$capitalgain[ad$capitalgain > 0], method = "content", nbins = 11))
```

```
     (14.1,2.2e+03]    (2.2e+03,3.1e+03]  (3.1e+03,3.67e+03]
                244                  228                 259
(3.67e+03,4.65e+03] (4.65e+03,5.46e+03]  (5.46e+03,7.3e+03]
                282                  229                 317
 (7.3e+03,7.69e+03] (7.69e+03,1.05e+04]  (1.05e+04,1.5e+04]
                298                   85                 157
 (1.5e+04,2.01e+04]    (2.01e+04,1e+05]
                397                  216
```

```r
plot(bin(data = ad$capitalgain[ad$capitalgain > 0], method = "content", nbins = 11))
```

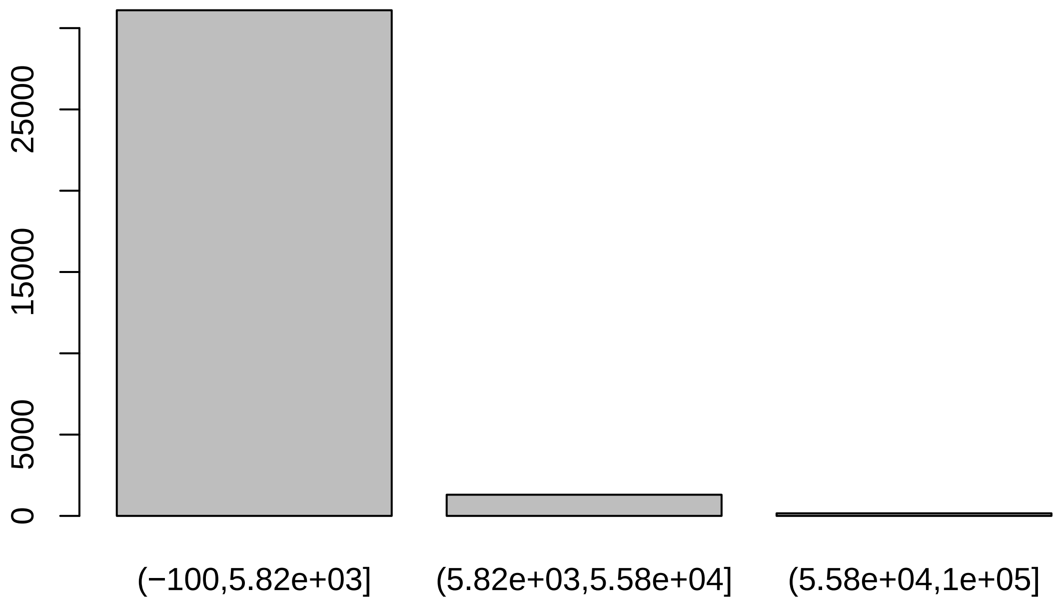


Another approach is to use the `cluster` method. Given a number of bins it will try to find the best clustering by that number. One caveat is that it will generate an error if the cluster size is 0. One can try to get around this by providing initial cluster positions.

```
table(bin(data = ad$capitalgain, method = "cluster", nbins = 3))
```

```

   (-100,5.82e+03] (5.82e+03,5.58e+04]    (5.58e+04,1e+05]
             31099                1303                 159
```

```
plot(bin(data = ad$capitalgain, method = "cluster", nbins = 3))
```



One last option offered by `OneR` is to use targeted binning. This may be questionable because it might be biasing the results or adding hidden additional degrees of freedom (Harrell).

To do this, I create a temporary data frame with just the variables involved. Optbin then operates on that data frame and based on the method chosen, will bin the data optimally.

```
tmp <- data.frame(capitalgain = ad$capitalgain)
tmp <- cbind(tmp, ad$income)

summary(optbin(tmp, method = "logreg"))
```

```
          capitalgain    ad$income
 (-100,4.01e+03] :30665   <=50K:24720
 (4.01e+03,1e+05]: 1896   >50K : 7841
```

```
summary(optbin(tmp, method = "infogain"))
```

```
          capitalgain    ad$income
 (-100,7.07e+03] :31162   <=50K:24720
 (7.07e+03,1e+05]: 1399   >50K : 7841
```

The difference between these two methods is that `logreg` puts the split at around $4,010, and `infogain` puts it at $7,070.

Because this variable is so skewed with 0, I'll just choose to use the `infogain` method to come up with a low/high value.

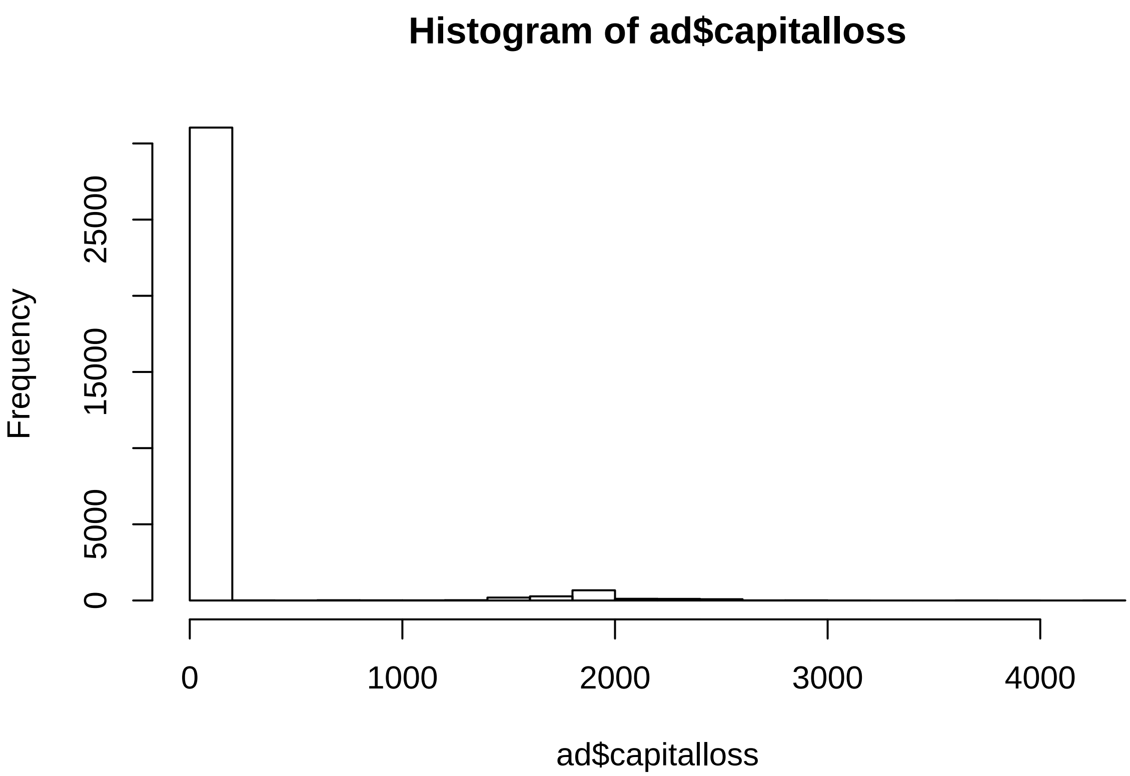
```r
tmp <- data.frame(capitalgain = ad$capitalgain)
tmp <- cbind(tmp, ad$income)

ad$capitalgain <- optbin(tmp, method = "infogain")$capitalgain
levels(ad$capitalgain) <- c("low", "high")

summary(ad$capitalgain)
```

```
  low  high
31162  1399
```

Capital-Loss

Let's look at `capitalloss`.

```r
hist(ad$capitalloss)
```



It looks a lot like `capitalgain`, so for expedience, we can apply the targeted `optbin` method.

```r
tmp <- data.frame(capitalloss = ad$capitalloss)
tmp <- cbind(tmp, ad$income)

ad$capitalloss <- optbin(tmp, method = "infogain")$capitalloss
levels(ad$capitalloss) <- c("low", "high")

summary(ad$capitalloss)
```

```
  low  high
31570   991
```

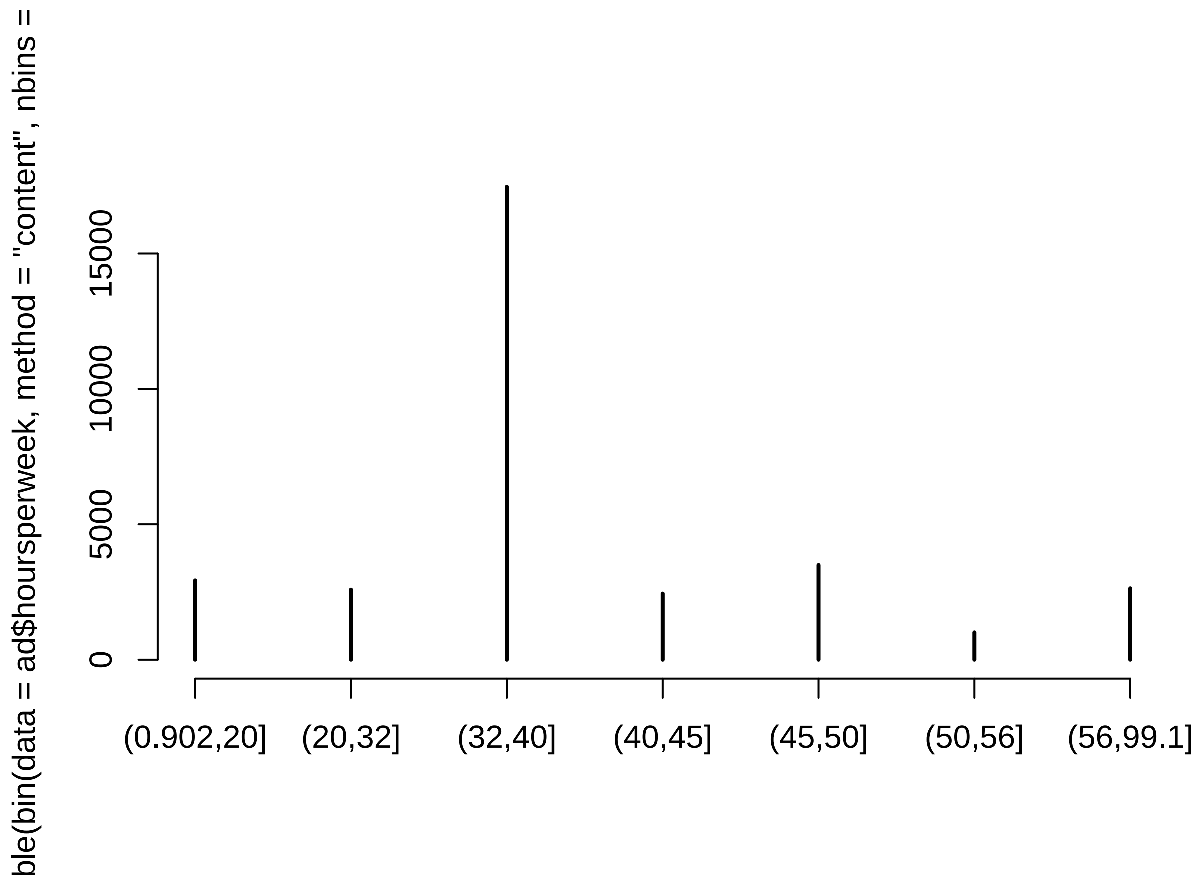
Hours-per-Week

With Hours-per-Week, there is probably a good basis for 3-bins like: “half-time”, “full-time”, “over-time”. But in this case, I’d like to bin with the DMIT-recommended 12 bins, using a content-based method.

```
table(bin(data = ad$hoursperweek, method = "content", nbins = 12))
```

```
(0.902,20]    (20,32]    (32,40]    (40,45]    (45,50]    (50,56]
      2928       2588      17464       2442       3496       1008
 (56,99.1]
      2635
```

```
plot(table(bin(data = ad$hoursperweek, method = "content", nbins = 12)))
```



In this case, the `quantile` method used only allowed for 7 bins, but the results look reasonable and we can always group some of the bins if necessary.

```
ad$hoursperweek <- bin(data = ad$hoursperweek, method = "content", nbins = 12)
```

The Data Looks Ready!

We’ve now discretized all the variables in our data frame.

```
str(ad)
```

```
'data.frame':   32561 obs. of  13 variables:
 $ age          : Factor w/ 12 levels "(15,25]","(25,30]",..: 4 6 4 7 2 4 6 7 3 5 ...
 $ workclass    : Factor w/ 9 levels "?","Federal-gov",..: 8 7 5 5 5 5 5 7 5 5 ...
```

```
 $ education    : Factor w/ 16 levels "Preschool","1st-4th",..: 13 13 9 7 13 14 5 9 14 13 ...
 $ maritalstatus: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
 $ occupation   : Factor w/ 15 levels "?","Adm-clerical",..: 2 5 7 7 11 5 9 5 11 5 ...
 $ relationship : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
 $ race         : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
 $ sex          : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
 $ capitalgain  : Factor w/ 2 levels "low","high": 1 1 1 1 1 1 1 1 2 1 ...
 $ capitalloss  : Factor w/ 2 levels "low","high": 1 1 1 1 1 1 1 1 1 1 ...
 $ hoursperweek : Factor w/ 7 levels "(0.902,20]","(20,32]",..: 3 1 3 3 3 3 1 4 5 3 ...
 $ nativecountry: Factor w/ 42 levels "?","Cambodia",..: 40 40 40 40 6 40 24 40 40 40 ...
 $ income       : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

Good Machine Learning Practice - Designating Test Data

In machine learning, it's important to set aside some data for testing your models on. Your training and parameter tuning should never see this data. It becomes the final, and hopefully unbiased, yard-stick by which to measure your models.

There is plenty of debate about how much data should be set aside. In this case, I'll randomly choose 20% of the data to be set aside as *test* data. I'll set up a vector with a `TRUE` or `FALSE` for each row in the dataset. This will be used by our function for taking a data frame and making it into an OCCAM file.

(Side note: UCIML already offers separate training and test files for this data set. It might have been more appropriate to load both files and explicitly mark the test data as such.)

```r
set.seed(3141593)    # setting the random seed to ensure the same results every time we generate this d

test_prob <- 0.2
test_rows <- sample(x = c(TRUE, FALSE), size = nrow(ad), replace = TRUE, prob = c(test_prob, 1 - test_p

# now investigate:
head(test_rows)
```

```
[1] FALSE FALSE FALSE FALSE FALSE  TRUE
```

```r
summary(test_rows)
```

```
   Mode   FALSE    TRUE 
logical   25964    6597 
```

```r
# to check the percentage
paste("% Test Rows: ", 100 * as.numeric(summary(test_rows)[["TRUE"]]) / nrow(ad))
```

```
[1] "% Test Rows:  20.260434261847"
```

Generating an OCCAM Data Set

I've created a function, `make_OCCAM_data` that will create an OCCAM compliant data file from our data frame.

```r
occam_data <- make_OCCAM_data(ad, DV=13, test_rows = test_rows)

# the first 30 rows of the OCCAM data
options(width = 10)
head(occam_data, n = 30)
```

```
 [1] "# OCCAM DATA FILE"
 [2] ""
 [3] ":nominal"
 [4] "age, 12, 1, A"
 [5] "workclass, 9, 1, B"
 [6] "education, 16, 1, C"
 [7] "maritalstatus, 7, 1, D"
 [8] "occupation, 15, 1, E"
 [9] "relationship, 6, 1, F"
[10] "race, 5, 1, G"
[11] "sex, 2, 1, H"
[12] "capitalgain, 2, 1, I"
[13] "capitalloss, 2, 1, J"
[14] "hoursperweek, 7, 1, K"
[15] "nativecountry, 42, 1, L"
[16] "income, 2, 2, Z"
[17] ""
[18] ":no-frequency"
[19] ""
[20] ":data"
[21] "4 8 13 5 2 2 5 2 1 1 3 40 1"
[22] "6 7 13 3 5 1 5 2 1 1 1 40 1"
[23] "4 5 9 1 7 2 5 2 1 1 3 40 1"
[24] "7 5 7 3 7 1 3 2 1 1 3 40 1"
[25] "2 5 13 3 11 6 3 1 1 1 3 6 1"
[26] "3 5 14 5 11 2 5 1 2 1 5 40 2"
[27] "5 5 13 3 5 1 5 2 1 1 3 40 2"
[28] "4 5 10 3 5 1 3 2 1 1 7 40 2"
[29] "2 8 13 3 11 1 2 2 1 1 3 20 2"
[30] "1 5 13 5 2 4 5 1 1 1 2 40 1"
```

And now write the data to a file.

```
filename <- "adult01.txt"
fileConn <- file(filename)
writeLines(occam_data, fileConn)
close(fileConn)
```

Now to ANALYZE THE DATA!!!

Before we get going with OCCAM, it might be interesting to use `OneR` to find which single variable is the most predictive.

OneR

Running `OneR` will let us know which variable is most predictive and how the values predict the dependent variable. With the `verbose` output, this should be the equivalent of done a 1-level, bottom-up search in OCCAM, with single-variable models. We're using the same training/test split for `OneR` as we are for OCCAM.

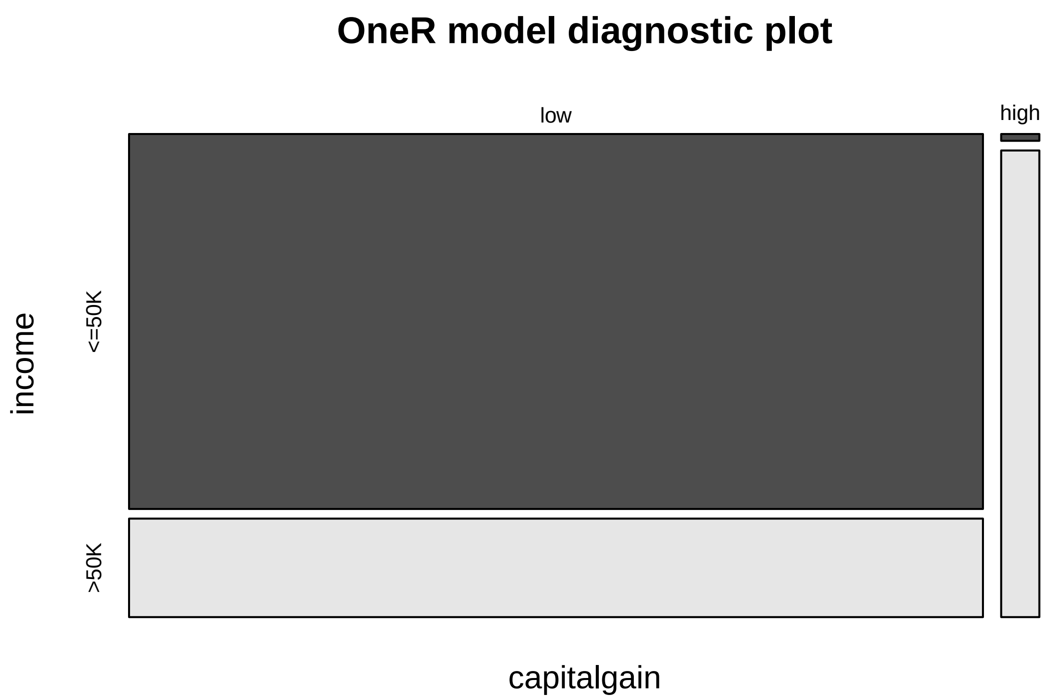
```
options(width = 80)
training <- ad[test_rows==FALSE, ]
test     <- ad[test_rows==TRUE, ]

model <- OneR(training, verbose = TRUE)
```

```
    Attribute     Accuracy
1 * capitalgain   80.02%
2   education     77.95%
3   capitalloss   77.19%
4   workclass     76.27%
5   age           75.87%
5   maritalstatus 75.87%
5   occupation    75.87%
5   relationship  75.87%
5   race          75.87%
5   sex           75.87%
5   hoursperweek  75.87%
5   nativecountry 75.87%
---
Chosen attribute due to accuracy
and ties method (if applicable): '*'
```

```r
summary(model)
```

```

Call:
OneR.data.frame(x = training, verbose = TRUE)

Rules:
If capitalgain = low  then income = <=50K
If capitalgain = high then income = >50K

Accuracy:
20777 of 25964 instances classified correctly (80.02%)

Contingency table:
       capitalgain
income      low   high   Sum
  <=50K * 19682     16 19698
  >50K     5171 * 1095  6266
  Sum     24853   1111 25964
---
Maximum in each column: '*'

Pearson's Chi-squared test:
X-squared = 3507.2, df = 1, p-value < 2.2e-16
```

```r
plot(model)
```




```r
prediction <- predict(model, test)

eval_model(prediction, test)
```

```
Confusion matrix (absolute):
          Actual
Prediction <=50K >50K  Sum
     <=50K  5018 1291 6309
     >50K      4  284  288
     Sum    5022 1575 6597

Confusion matrix (relative):
          Actual
Prediction <=50K >50K  Sum
     <=50K  0.76 0.20 0.96
     >50K   0.00 0.04 0.04
     Sum    0.76 0.24 1.00

Accuracy:
0.8037 (5302/6597)

Error rate:
0.1963 (1295/6597)

Error rate reduction (vs. base rate):
0.1778 (p-value < 2.2e-16)
```

According to `OneR`, `capitalgain` is the best single variable for predicting `income`.

OCCAM Variable-Based Model - Single Variable

To compare with `OneR`, I decided to start with running a single-variable/1-level search in OCCAM.

ID	MODEL	Level	H	dDF	dLR	Alpha	Inf	%dH(DV)	dAIC	dBIC	Inc.Alpha	Prog.	%C(Data)	%cover	%C(Test)	%miss
13*	IV:FZ	1	14.2151	5	5932.4430	0.0000	0.25221430	20.6734	5922.4430	5881.6206	0.0000	1	75.8666	100.0000	76.1255	0.0000
12*	IV:DZ	1	14.2232	6	5642.7549	0.0000	0.23989838	19.6639	5630.7549	5581.7681	0.0000	1	75.8666	100.0000	76.1255	0.0000
11*	IV:AZ	1	14.2855	11	3400.0163	0.0000	0.14454968	11.8484	3378.0163	3288.2071	0.0000	1	75.8666	100.0000	76.1255	0.0000
10*	IV:EZ	1	14.2860	14	3383.2744	0.0000	0.14383791	11.7900	3355.2744	3240.9719	0.0000	1	75.8666	100.0000	76.1255	0.0000
9*	IV:CZ	1	14.2862	15	3373.7147	0.0000	0.14343148	11.7567	3343.7147	3221.2477	0.0000	1	77.9502	100.0000	77.9900	0.0000
8*	IV:IZ	1	14.2936	1	3109.9911	0.0000	0.13221943	10.8377	3107.9911	3099.8267	0.0000	1	80.0223	100.0000	80.3699	0.0000
7*	IV:KZ	1	14.3250	6	1979.5250	0.0000	0.08415833	6.8982	1967.5250	1918.5382	0.0000	1	75.8666	100.0000	76.1255	0.0000
6*	IV:HZ	1	14.3427	1	1342.7868	0.0000	0.05708779	4.6793	1340.7868	1332.6224	0.0000	1	75.8666	100.0000	76.1255	0.0000
5*	IV:JZ	1	14.3571	1	823.6600	0.0000	0.03501742	2.8703	821.6600	813.4956	0.0000	1	77.1915	100.0000	77.5049	0.0000
4*	IV:BZ	1	14.3575	8	810.1635	0.0000	0.03444362	2.8233	794.1635	728.8478	0.0000	1	76.2710	100.0000	76.4742	0.0000
3*	IV:LZ	1	14.3706	41	336.0890	0.0000	0.01428862	1.1712	254.0890	-80.6541	0.0000	1	75.8666	100.0000	76.1255	0.0000
2*	IV:GZ	1	14.3715	4	304.3610	0.0000	0.01293973	1.0606	296.3610	263.7032	0.0000	1	75.8666	100.0000	76.1255	0.0000
1*	IV:Z	0	14.3800	0	0.0000	1.0000	0.00000000	0.0000	0.0000	0.0000	0.0000	0	75.8666	100.0000	76.1255	0.0000
ID	**MODEL**	**Level**	**H**	**dDF**	**dLR**	**Alpha**	**Inf**	**%dH(DV)**	**dAIC**	**dBIC**	**Inc.Alpha**	**Prog.**	**%C(Data)**	**%cover**	**%C(Test)**	**%miss**

```
Best Model(s) by dBIC:
13*  IV:FZ  1   14.2151  5   5932.4430  0.0000  0.25221430  20.6734  5922.4430  5881.6206  0.0000  1   75.8666  100.0000  76.1255  0.0000
Best Model(s) by dAIC:
13*  IV:FZ  1   14.2151  5   5932.4430  0.0000  0.25221430  20.6734  5922.4430  5881.6206  0.0000  1   75.8666  100.0000  76.1255  0.0000
Best Model(s) by Information, with all Inc. Alpha < 0.05:
13*  IV:FZ  1   14.2151  5   5932.4430  0.0000  0.25221430  20.6734  5922.4430  5881.6206  0.0000  1   75.8666  100.0000  76.1255  0.0000
Best Model(s) by %C(Test):
Warning: models should not be selected based on %correct(test).
8*   IV:IZ  1   14.2936  1   3109.9911  0.0000  0.13221943  10.8377  3107.9911  3099.8267  0.0000  1   80.0223  100.0000  80.3699  0.0000
```

We can see that OCCAM and `OneR` agree that `I`, or `capitalgains` are the best predictor when looking at accuracy on the training data.

OCCAM Variable-Based Model w/o Loops

Running OCCAM with the defaults (directed and loopless), the results are below.

ID	MODEL	Level	H	dDF	dLR	Alpha	Inf	%dH(DV)	dAIC	dBIC	Inc.Alpha	Prog.	%C(Data)	%cover	%C(Test)	%miss
22	IV:BFGHIJKZ	7	14.0641	15119	11367.4390	1.0000	0.48327994	39.6132	-18870.5610	-142309.1261	1.0000	14	83.1998	8.9616	81.6735	2.3192
21*	IV:CFIZ	3	14.0752	191	10968.0074	0.0000	0.46629834	38.2213	10586.0074	9026.5944	0.0000	7	84.2205	76.5625	84.3262	0.0758
20	IV:DFGHIJKZ	7	14.0791	11759	10830.3880	1.0000	0.46044753	37.7417	-12687.6120	-108693.5706	1.0000	13	82.0598	8.2568	81.6280	1.3643
19*	IV:EFIZ	3	14.0805	179	10777.9764	0.0000	0.45821929	37.5591	10419.9764	8958.5369	0.0000	6	83.8892	80.5556	83.9624	0.1364
18	IV:BDFGHIJZ	7	14.0894	15119	10460.2134	1.0000	0.44470978	36.4517	-19777.7866	-143216.3517	1.0000	14	82.6067	6.0847	81.9615	1.6371
17	IV:FGHIJKZ	6	14.0911	1679	10398.1892	0.0000	0.44207286	36.2356	7040.1892	-6667.9497	1.0000	16	81.9288	27.3214	81.7644	0.5154
16*	IV:FHIJKZ	5	14.0991	335	10107.6260	0.0000	0.42971974	35.2230	9437.6260	6702.5298	0.0000	9	81.7902	50.2976	81.8554	0.0910
15*	IV:FIJKZ	4	14.1013	167	10031.5380	0.0000	0.42648490	34.9579	9697.5380	8334.0722	0.0000	8	81.7825	67.8571	81.8554	0.0303
14	IV:BFGHIJZ	6	14.1013	2159	10029.1188	0.0000	0.42638205	34.9494	5711.1188	-11915.9639	1.0000	11	82.5104	20.6019	82.0828	0.5760
13	IV:DFGHIJZ	6	14.1115	1679	9663.6589	0.0000	0.41084474	33.6759	6305.6589	-7402.4800	1.0000	12	81.6977	18.4524	82.0070	0.2425
12	IV:DFHIJZ	5	14.1171	335	9462.8934	0.0000	0.40230931	32.9763	8792.8934	6057.7972	0.9928	9	81.6785	34.8214	82.0676	0.0455
11	IV:FGHIJZ	5	14.1193	239	9381.1470	0.0000	0.39883391	32.6914	8903.1470	6951.8396	0.9382	10	81.6631	46.6667	82.0828	0.0303
10*	IV:FGIJZ	4	14.1220	119	9284.0481	0.0000	0.39470581	32.3530	9046.0481	8074.4767	0.0014	8	81.6592	62.5000	82.0979	0.0000
9*	IV:FHIJZ	4	14.1235	47	9230.1281	0.0000	0.39241343	32.1651	9136.1281	8752.3982	0.0000	8	81.6592	66.6667	82.0828	0.0000
8*	IV:FIJZ	3	14.1260	23	9141.6485	0.0000	0.38865178	31.8568	9095.6485	8907.8658	0.0000	6	81.6592	75.0000	82.0828	0.0000
7*	IV:CFZ	2	14.1319	95	8930.3405	0.0000	0.37966814	31.1204	8740.3405	7964.7162	0.0000	4	82.0829	100.0000	82.1131	0.0000
6*	IV:FIZ	2	14.1436	11	8508.7034	0.0000	0.36174249	29.6511	8486.7034	8396.8943	0.0000	4	80.0223	100.0000	80.3699	0.0000
5*	IV:DIZ	2	14.1497	13	8289.1094	0.0000	0.35240658	28.8859	8263.1094	8156.9713	0.0000	3	80.0223	100.0000	80.3699	0.0000
4*	IV:FZ	1	14.2151	5	5932.4430	0.0000	0.25221430	20.6734	5922.4430	5881.6206	0.0000	1	75.8666	100.0000	76.1255	0.0000
3*	IV:DZ	1	14.2232	6	5642.7549	0.0000	0.23989838	19.6639	5630.7549	5581.7681	0.0000	1	75.8666	100.0000	76.1255	0.0000
2*	IV:AZ	1	14.2855	11	3400.0163	0.0000	0.14454968	11.8484	3378.0163	3288.2071	0.0000	1	75.8666	100.0000	76.1255	0.0000
1*	IV:Z	0	14.3800	0	0.0000	1.0000	0.00000000	0.0000	0.0000	0.0000	0.0000	0	75.8666	100.0000	76.1255	0.0000
ID	**MODEL**	**Level**	**H**	**dDF**	**dLR**	**Alpha**	**Inf**	**%dH(DV)**	**dAIC**	**dBIC**	**Inc.Alpha**	**Prog.**	**%C(Data)**	**%cover**	**%C(Test)**	**%miss**

```
Best Model(s) by dBIC:
21*  IV:CFIZ  3   14.0752  191   10968.0074  0.0000  0.46629834  38.2213  10586.0074  9026.5944  0.0000  7   84.2205  76.5625  84.3262  0.0758
Best Model(s) by dAIC:
21*  IV:CFIZ  3   14.0752  191   10968.0074  0.0000  0.46629834  38.2213  10586.0074  9026.5944  0.0000  7   84.2205  76.5625  84.3262  0.0758
Best Model(s) by Information, with all Inc. Alpha < 0.05:
21*  IV:CFIZ  3   14.0752  191   10968.0074  0.0000  0.46629834  38.2213  10586.0074  9026.5944  0.0000  7   84.2205  76.5625  84.3262  0.0758
Best Model(s) by %C(Test):
Warning: models should not be selected based on %correct(test).
21*  IV:CFIZ  3   14.0752  191   10968.0074  0.0000  0.46629834  38.2213  10586.0074  9026.5944  0.0000  7   84.2205  76.5625  84.3262  0.0758
```

The best model, by all criteria, is `IV:CFIZ`, where C is education, F is relationship, and I is capitalgain.

Relationship looks like a strange variable, and it's not clear what it represents. Comparing it to `sex` yields:

```r
table(ad[,c(6,8)])
```

```
                sex
relationship     Female  Male
  Husband             1 13192
  Not-in-family    3875  4430
  Other-relative    430   551
  Own-child        2245  2823
  Unmarried        2654   792
  Wife             1566     2
```

The best I can tell from reading the US Census site is that on a Census form, all the people in a house-hold are listed in a single report. Then relationship tells how the additional people are related to the primary person filling the form. I'm just puzzled about why that would be relevant.

I have to admit I was pleased to see `capitalgain` made the list!

A model *with* loops

Now allowing for loops (which took about 3 minutes to run):

ID	MODEL	Level	H	dDF	dLR	Alpha	Inf	%dH(DV)	dAIC	dBIC	Inc.Alpha	Prog.	%C(Data)	%cover	%C(Test)	%miss
22*	IV:AZ:CZ:EZ:FZ:IZ:JZ:KZ	7	14.0224	53	12870.6196	0.0000	0.54718678	44.8515	12764.6196	12331.9029	0.0000	17	85.8535	2.0856	82.6588	23.7229
21*	IV:AZ:CZ:DZ:EZ:IZ:JZ:KZ	7	14.0232	54	12841.9297	0.0000	0.54596704	44.7515	12733.9297	12293.0485	0.0000	18	85.7611	1.6539	82.8255	22.2980
20*	IV:AZ:CZ:EZ:FZ:HZ:IZ:JZ	7	14.0283	48	12657.8789	0.0000	0.53814223	44.1101	12561.8789	12169.9845	0.0000	19	85.5800	4.4980	83.9624	12.1874
19*	IV:AZ:CZ:EZ:FZ:IZ:JZ	6	14.0311	47	12557.2482	0.0000	0.53386398	43.7595	12463.2482	12079.5183	0.0000	14	85.5800	7.6071	84.0382	9.9742
18*	IV:AZ:CZ:DZ:EZ:IZ:JZ	6	14.0315	48	12543.2786	0.0000	0.53327006	43.7108	12447.2786	12055.3842	0.0000	15	85.5377	6.0280	84.2504	9.4740
17*	IV:AZ:CZ:FZ:IZ:JZ:KZ	6	14.0352	39	12409.4084	0.0000	0.52757865	43.2443	12331.4084	12012.9942	0.0000	14	85.2604	11.9327	83.6592	6.3817
16*	IV:AZ:CZ:EZ:FZ:IZ	5	14.0413	46	12190.0207	0.0000	0.51825151	42.4797	12098.0207	11722.4553	0.0000	11	85.1294	13.9728	84.1898	8.5797
15*	IV:AZ:CZ:DZ:EZ:IZ	5	14.0418	47	12169.9936	0.0000	0.51740008	42.4099	12075.9936	11692.2637	0.0000	12	85.0639	11.0665	84.2656	8.2613
14*	IV:AZ:CZ:FZ:IZ:JZ	5	14.0463	33	12011.2321	0.0000	0.51065042	41.8567	11945.2321	11675.8047	0.0000	13	85.0254	28.5590	84.5081	1.6068
13*	IV:AZ:CZ:FZ:IZ	4	14.0573	32	11612.6397	0.0000	0.49370450	40.4677	11548.6397	11287.3768	0.0000	10	84.4824	47.6128	84.1746	1.1520
12*	IV:AZ:CZ:DZ:IZ	4	14.0581	33	11586.5383	0.0000	0.49259482	40.3767	11520.5383	11251.1109	0.0000	9	84.3591	39.2113	83.9321	1.1520
11*	IV:CZ:EZ:FZ:IZ	4	14.0602	35	11508.7782	0.0000	0.48928889	40.1057	11438.7782	11153.0219	0.0000	8	84.7597	39.3056	84.5384	1.1824
10*	IV:CZ:FZ:IZ	3	14.0788	21	10838.6238	0.0000	0.46079767	37.7704	10796.6238	10625.1700	0.0000	5	84.2012	76.5625	84.2959	0.0758
9*	IV:CZ:DZ:IZ	3	14.0801	22	10793.7955	0.0000	0.45889183	37.6142	10749.7955	10570.1772	0.0000	6	84.1280	66.9643	84.2656	0.0910
8*	IV:EZ:FZ:IZ	3	14.0862	20	10575.1252	0.0000	0.44959519	36.8522	10535.1252	10371.8358	0.0000	5	83.8546	80.5556	83.9624	0.1364
7*	IV:CZ:FZ	2	14.1345	20	8834.1549	0.0000	0.37557887	30.7852	8794.1549	8630.8656	0.0000	4	81.9904	100.0000	82.0676	0.0000
6*	IV:CZ:DZ	2	14.1363	21	8771.0163	0.0000	0.37289457	30.5652	8729.0163	8557.5625	0.0000	3	81.9481	90.1786	82.0524	0.0000
5*	IV:FZ:IZ	2	14.1439	6	8498.1299	0.0000	0.36129296	29.6143	8486.1299	8437.1431	0.0000	4	80.0223	100.0000	80.3699	0.0000
4*	IV:FZ	1	14.2151	5	5932.4430	0.0000	0.25221430	20.6734	5922.4430	5881.6206	0.0000	1	75.8666	100.0000	76.1255	0.0000
3*	IV:DZ	1	14.2232	6	5642.7549	0.0000	0.23989838	19.6639	5630.7549	5581.7681	0.0000	1	75.8666	100.0000	76.1255	0.0000
2*	IV:AZ	1	14.2855	11	3400.0163	0.0000	0.14454968	11.8484	3378.0163	3288.2071	0.0000	1	75.8666	100.0000	76.1255	0.0000
1*	IV:Z	0	14.3800	0	0.0000	1.0000	0.00000000	0.0000	0.0000	0.0000	0.0000	0	75.8666	100.0000	76.1255	0.0000
ID	MODEL	Level	H	dDF	dLR	Alpha	Inf	%dH(DV)	dAIC	dBIC	Inc.Alpha	Prog.	%C(Data)	%cover	%C(Test)	%miss

```
Best Model(s) by dBIC:
22*  IV:AZ:CZ:EZ:FZ:IZ:JZ:KZ  7  14.0224  53  12870.6196  0.0000  0.54718678  44.8515  12764.6196  12331.9029  0.0000  17  85.8535  2.0856  82.6588  23.7229
Best Model(s) by dAIC:
22*  IV:AZ:CZ:EZ:FZ:IZ:JZ:KZ  7  14.0224  53  12870.6196  0.0000  0.54718678  44.8515  12764.6196  12331.9029  0.0000  17  85.8535  2.0856  82.6588  23.7229
Best Model(s) by Information, with all Inc. Alpha < 0.05:
22*  IV:AZ:CZ:EZ:FZ:IZ:JZ:KZ  7  14.0224  53  12870.6196  0.0000  0.54718678  44.8515  12764.6196  12331.9029  0.0000  17  85.8535  2.0856  82.6588  23.7229
Best Model(s) by %C(Test):
Warning: models should not be selected based on %correct(test).
11*  IV:CZ:EZ:FZ:IZ  4  14.0602  35  11508.7782  0.0000  0.48928889  40.1057  11438.7782  11153.0219  0.0000  8  84.7597  39.3056  84.5384  1.1824
```

Here, the best model by dBIC is `IV:AZ:CZ:EZ:FZ:IZ:JZ:KZ`, giving us: age, education, occupation, relationship, capitalgain, capitalloss, and hoursperweek. However this is over half the variables available!

I lean towards choosing `IV:AZ:CZ:FZ:IZ`. It uses fewer variables (age, education, relationship, and capitalgain), losing only 8.4701129 % of dBIC, and only 1.3711 points of accuracy (against the data), while improving dDF by 21 degrees.

A State-Based Model Search

To do a state-based search, we'll need to tell OCCAM to ignore some variables, otherwise it won't run (the state-space is too large!). Here I set up the columns to ignore then see what we're keeping. Starting with the model from above: `IV:AZ:CZ:FZ:IZ`, we'll modify and generate a new OCCAM file by setting "ignore" on the unwanted variables.

```
keep <- c(1, 3, 6, 9, 13)
ignore <- setdiff(1:13, keep)

str(ad[ , keep])
```

```
'data.frame':   32561 obs. of  5 variables:
 $ age         : Factor w/ 12 levels "(15,25]","(25,30]",..: 4 6 4 7 2 4 6 7 3 5 ...
 $ education   : Factor w/ 16 levels "Preschool","1st-4th",..: 13 13 9 7 13 14 5 9 14 13 ...
 $ relationship: Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
 $ capitalgain : Factor w/ 2 levels "low","high": 1 1 1 1 1 1 1 1 2 1 ...
 $ income      : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

Adjusting Cardinality of Age

We should also consider reducing the cardinality of some of the larger variables such as `age` and `education`. We covered `age` earlier and can simply group the existing lower level hierarchy to the next level up.

```
# Old: 15-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59, 60-64, 65-69, 70-74, 75+
# New: 15-24, 25-34,        35-44,        45-54,        55-64,        65-74,        75+

# make a new vector with the new factor labels.  In the data above the non-end factors are grouped by 2
# so the new vector states the new factor twice for each of those
new_age_breaks <- c("15-24",
                    "25-34", "25-34",   # encompases 25-29 and 30-34
                    "35-44", "35-44",
                    "45-54", "45-54",
                    "55-64", "55-64",
                    "65-74", "65-74",
                    "75+")

# make a copy of the age variable because we don't want to make this change to our data (just yet)

levels(ad$age) <- new_age_breaks

str(ad$age)
```

```
 Factor w/ 7 levels "15-24","25-34",..: 3 4 3 4 2 3 4 4 2 3 ...
```

Adjusting Cardinality of Education

There is probably an obvious place to break `education`, since years of education and income are probably well-correlated. We can investigate how to group the levels using `OneR`.

```
OneR(ad[,c(1, 3, 6, 13)], verbose = TRUE)
```

```
    Attribute    Accuracy
1 * education    77.96%
2   age          75.92%
2   relationship 75.92%
---
Chosen attribute due to accuracy
and ties method (if applicable): '*'
```

```
Call:
OneR.data.frame(x = ad[, c(1, 3, 6, 13)], verbose = TRUE)

Rules:
If education = Preschool    then income = <=50K
If education = 1st-4th      then income = <=50K
If education = 5th-6th      then income = <=50K
If education = 7th-8th      then income = <=50K
If education = 9th          then income = <=50K
If education = 10th         then income = <=50K
If education = 11th         then income = <=50K
If education = 12th         then income = <=50K
If education = HS-grad      then income = <=50K
If education = Some-college then income = <=50K
If education = Assoc-voc    then income = <=50K
If education = Assoc-acdm   then income = <=50K
If education = Bachelors    then income = <=50K
If education = Masters      then income = >50K
If education = Prof-school  then income = >50K
If education = Doctorate    then income = >50K

Accuracy:
25384 of 32561 instances classified correctly (77.96%)
```

Those results that show that the states Masters, Prof-school, and Doctorate predict income > 50k. But maybe it would be useful to use 3 levels, "high school or less", "some college", "graduate college".

```
# "Preschool", "1st-4th", "5th-6th","7th-8th", "9th", "10th", "11th", "12th", "HS-grad", "Some-college"
# New: 15-24, 25-34,        35-44,        45-54,        55-64,        65-74,        75+

# make a new vector with the new factor labels.  In the data above the non-end factors are grouped by 2
# so the new vector states the new factor twice for each of those
new_ed_breaks <- c("K12", "K12", "K12", "K12", "K12", "K12", "K12", "K12", "<=HS",
                   "<=Bachelors", "<=Bachelors", "<=Bachelors", "<=Bachelors",
                   "Graduate", "Graduate", "Graduate")

# make a copy of the age variable because we don't want to make this change to our data (just yet)

levels(ad$education) <- new_ed_breaks

str(ad$education)
```

```
 Factor w/ 4 levels "K12","<=HS","<=Bachelors",..: 3 3 2 1 3 4 1 2 4 3 ...
```

And now modify our OCCAM file.

```
occam_data <- make_OCCAM_data(ad, DV=13, test_rows = test_rows, ignore_cols = ignore)


filename <- "adult04.txt"
fileConn <- file(filename)
writeLines(occam_data, fileConn)
close(fileConn)
```

To get SB-Search to even run, it was necessary to reduce the cardinality as above. That yielded the below.

ID	MODEL	Level	H	dDF	dLR	Alpha	Inf	%dH(DV)	dAIC	dBIC	Inc.Alpha	Prog.	%C(Data)	%cover	%C(Test)	%miss
14*	ACFIZ	5	6.3092	251	11206.1715	0.0000	1.00000000	39.0512	10704.1715	8654.8905	0.0000	13	83.7852	100.0000	83.7350	0.0000
13*	IV:ACFI1Z:Z	4	6.3150	126	10998.1381	0.0000	0.98143582	38.3263	10746.1381	9717.4154	0.0000	10	83.7698	100.0000	83.9321	0.0000
12*	IV:CFIZ:Z	4	6.3339	35	10315.9138	0.0000	0.92055648	35.9489	10245.9138	9960.1574	0.0000	8	82.2716	100.0000	82.3708	0.0000
11*	IV:AFIZ:Z	4	6.3565	83	9501.2922	0.0000	0.84786247	33.1101	9335.2922	8657.6415	0.0000	9	80.3459	100.0000	80.4760	0.0000
10*	IV:AFI1Z:Z	3	6.3617	42	9314.4816	0.0000	0.83119213	32.4591	9230.4816	8887.5740	0.0000	6	80.3382	100.0000	80.5669	0.0000
9*	IV:FIZ:Z	3	6.3841	11	8508.7034	0.0000	0.75928727	29.6511	8486.7034	8396.8943	0.0000	7	80.0223	100.0000	80.3699	0.0000
8*	IV:CFZ:Z	3	6.3931	17	8186.8905	0.0000	0.73056981	28.5296	8152.8905	8014.0946	0.0000	5	80.7002	100.0000	80.8398	0.0000
7*	IV:FZ:Z	2	6.4557	5	5932.4430	0.0000	0.52939070	20.6734	5922.4430	5881.6206	0.0000	4	75.8666	100.0000	76.1255	0.0000
6*	IV:AI1Z:Z	2	6.4606	7	5756.6350	0.0000	0.51370221	20.0607	5742.6350	5685.4837	0.0000	2	80.0223	100.0000	80.3699	0.0000
5*	IV:CF1Z:Z	2	6.4614	3	5727.4363	0.0000	0.51109661	19.9589	5721.4363	5696.9429	0.0000	4	80.1841	100.0000	80.0061	0.0000
4*	IV:F1Z:Z	1	6.5028	1	4235.9436	0.0000	0.37800096	14.7614	4233.9436	4225.7791	0.0000	1	75.8666	100.0000	76.1255	0.0000
3*	IV:I2Z:Z	1	6.5341	1	3110.0505	0.0000	0.27753016	10.8379	3108.0505	3099.8861	0.0000	1	80.0223	100.0000	80.3699	0.0000
2*	IV:A1I1Z:Z	1	6.5489	1	2579.0115	0.0000	0.23014207	8.9873	2577.0115	2568.8470	0.0000	1	75.8666	100.0000	76.1255	0.0000
1*	IV:Z	0	6.6205	0	0.0000	1.0000	0.00000000	0.0000	0.0000	0.0000	0.0000	0	75.8666	100.0000	76.1255	0.0000
ID	MODEL	Level	H	dDF	dLR	Alpha	Inf	%dH(DV)	dAIC	dBIC	Inc.Alpha	Prog.	%C(Data)	%cover	%C(Test)	%miss

```
Best Model(s) by dBIC:
12*  IV:CFIZ:Z    4    6.3339  35   10315.9138  0.0000  0.92055648  35.9489  10245.9138  9960.1574  0.0000   8    82.2716  100.0000  82.3708  0.0000
Best Model(s) by dAIC:
13*  IV:ACFI1Z:Z  4    6.3150  126  10998.1381  0.0000  0.98143582  38.3263  10746.1381  9717.4154  0.0000   10   83.7698  100.0000  83.9321  0.0000
Best Model(s) by Information, with all Inc. Alpha < 0.05:
14*  ACFIZ        5    6.3092  251  11206.1715  0.0000  1.00000000  39.0512  10704.1715  8654.8905  0.0000   13   83.7852  100.0000  83.7350  0.0000
Best Model(s) by %C(Test):
Warning: models should not be selected based on %correct(test).
13*  IV:ACFI1Z:Z  4    6.3150  126  10998.1381  0.0000  0.98143582  38.3263  10746.1381  9717.4154  0.0000   10   83.7698  100.0000  83.9321  0.0000
```

It appears that the cardinality reduction weakened the models so that the variable-based models with loops give the best result.

Appendix: Useful Links while Writing This Vignette

Here are some sites that help solve various problems used in generating this document.

- Embedding another `.R` file in this document without copying it in directly: Making use of external R code in knitr and R markdown
- Outputing the character vector one per line (non-interleaved) to make OCCAM file look correct in this document: Print an R vector vertically
- Making R output not have beginning hashes: Remove Hashes in R Output from RMarkdown and Knitr
- Getting unique pairs of values from two variables: unique() for more than one variable
- Changing single column names in a data frame: Changing column names of a data frame
- Changing the ordering of factors: Changing the order of levels of a factor
- Dropping a data frame column by name: Drop data frame columns by name
- Removing white-space while importing csv: R fread and strip white
- Extracting values from output functions like `summary`: How do I extract just the number from a named number (without the name)?
- Grouping factors - as in the hierarchies of age: Grouping 2 levels of a factor in R
- Using `optbin` from `OneR`: optbin

References

Kohavi R., Becker B., "Adult", http://archive.ics.uci.edu/ml/datasets/Adult, UCI Machine Learning Repository http://archive.ics.uci.edu/ml. (1996), Irvine, CA: University of California, School of Information and Computer Science. (2017)

Harrell, Frank, "Problems Caused by Categorizing Continuous Variables", http://biostat.mc.vanderbilt.edu/wiki/Main/CatContinuous, (2017)