**Step to bin the data**

1. For each column with more than 8 levels, the data was binned to allow calculation of churn percentage for the levels of the factors.
   1. Reason: For factors such as internet minutes that can vary from 0 up to 2000 min / month, Each row is unique, so an average churn cannot be calculated. That is, at the row level, a customer either churned or was retained. By binning this data, enough rows are made available to calculate the average churn for the group or level.
2. “Raw” histograms were generated to visually suggest where the bins should be.
3. The following columns were binned:

|  |
| --- |
| tenure |
| phone\_min |
| internet\_min |
| phone\_monthly\_charges |
| internet\_monthly\_charges |
| avg\_monthly\_bill |
| number\_customer\_service\_calls |

1. The bin edges were generated with list comprehensions or by manually specifying them. In all cases these were lists of the breakpoints.
2. A mapping function was obtained (given an unbinned raw value, put it in the bin that it is closest to) from the internet (<https://www.geeksforgeeks.org/python-find-closest-number-to-k-in-given-list/> called “closest”.
3. Loop over the list of binned column names
   1. For each row in the column to be binned, calculate (using ‘closest’ function) the proper bin and place this in the binned column.
      1. Used apply and lambda for this exercise
4. Once the data was binned, use ‘groupby’ (in a loop over all the binned column names) to sum up the counts for each bin.
5. Graph the bin counts as a bar graph and compare with the original raw histograms.
6. The binned columns were no ready to be used by ‘dataframe\_utils’ to calculate the percent churns for the category.