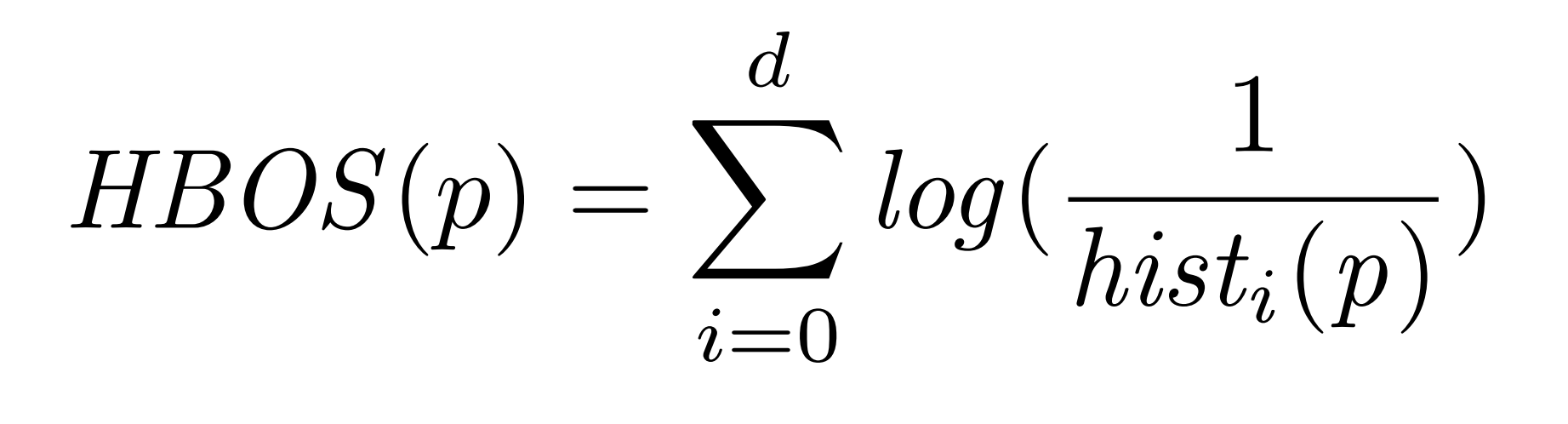
**Graham\_Crackers Outline**

Here’s the excerpt from our main article that describes the dynamic bin width method and overall HBOS score process we are trying to execute.

*“The dynamic binwidth is determined as follows: values are sorted first and then a fixed amount of N /k successive values are grouped into a single bin where N is the number of total instances and k the number of bins. Since the area of a bin in a histogram represents the number of observations, it is the same for all bins in our case. Because the width of the bin is defined by the first and the last value and the area is the same for all bins, the height of each individual bin can be computed. This means that bins covering a larger interval of the value range have less height and represent that way a lower density. However, there is one exception: Under certain circumstances, more than k data instances might have exactly the same value, for example if the feature is an integer and a long-tail distribution has to be estimated. In this case, our algorithm must allow to have more than N /k values in the same bin.*

*An often used rule of thumb is setting k to the square root of the number of instances N. Now, for each dimension d, an individual histogram has been computed (regardless if categorical, fixed-width or dynamic-width), where the height of each single bin represents a density estimation. The histograms are then normalized such that the maximum height is 1.0. This ensures an equal weight of each feature to the outlier score. Finally, the HBOS of every instance p is calculated using the corresponding height of the bins where the instance is located”*



Our workstations are in list\_of\_workstations[]. We want to do this process (the entire HBOS score) for each workstation (p) and then append its score to the object as a property. For each workstation, we have to combine individual log(…) scores obtained from each variable. Structuring the layout using order of operations, these are the steps we have to take to execute our plan. Tasks one through three are to be done for each variable, step four is the summation of each of these results, as indicated by placement of for loops.

Open for loop one, for each workstation in list.

Open for loop two, for each var in our set

**Task 1: Code hist\_i(p) process**

We first have to calculate hist\_i(p). This is the histogram-based density estimation. We don’t have to create a histogram for each var, only code the mathematical representation  
  
**Task 2: Figure out how to take an inverse in Python: Numpy or not?**

Next, we have to take its’ inverse, and it may not be necessary to use numpy, depending on how python handles division (refer to basic python operations to check).

**Task 3: Figure out how to take the log of the above**

Next, we have to take the log(base 10) of the inverse of the hist\_i(p). Probably will require numpy for this, but double check with basic python operations first.

For loop two ends.

**Task 4: Add the results from all vars**

Finally, we have to look into summations in python. We should be able to accomplish this strictly in python, adding the log(…) results in a for loop running through all the variables’ results (how should we store these?) and this final result is the HBOS score for the workstation. Or we can see if numpy can do this all in one, and is able to refer to specific properties as index values

**Task 5: Attach HBOS score to the workstation Object**

After getting the final HBOS score, add it to the object it corresponds to.

For loop one ends.

**Task 5: Sort list\_of\_workstations**

Sort this list in descending order by HBOS score

**Task 6: Print top ten offenders**

Print branch number and unique workstation id for first ten in list.

Notes and follow-up questions:

* We have to decide how to handle ranking across timestamps, different times might have different top ten offenders since not all sheets include all workstations (some workstations don’t have data until hours later than the rest).
* Have to figure out how to store log results before adding, if we can’t do in one step
* We should use same k and N values for all hist scores, even if different timestamp sheets contain different number of workstation information, for consistency
* See blue highlight: Is this the case?
* We will have to add HBOS score property into the class structure and leave blank until it runs through the process
* If syntax rules prevent Tasks one through three to run in a for loop through each var (since they’re properties of a class) we can either copy and paste to manually run it through for each var property, or maybe see if we can somehow refer to specific columns in the original df to obtain the information?
* This is based on reading in a timesheet with a single row of data for each workstation available.
* Will not work to have . (periods/decimals) in variable names, so must replace with underscores