histmdl: An R Package for Histogram-Like Data Visualization

Jouke Witteveen
Universiteit van Amsterdam

Abstract

We present a histogram-inspired visualization scheme for numeric data. The resulting visualizations are shown to have several advantages over ordinary histograms. The scheme is based on the minimum description length principle from information theory and hinges on a novel generalization of the Kraft–McMillan theorem. Effectively, the scheme performs binning by compression. An implementation is made available in the form of the R package **histmdl**.

Keywords: data visualization, density estimation, histograms, minimum description length.

1. Introduction

The classical histogram is a useful tool for getting an impression of the distribution of data. Its graphical nature gives it a considerable advantage over textual representation in the form of tables. By its simplicity, the classical histogram is computationally easy to work with and more sensitive to non-smooth structural properties such as outliers than smoothing methods such as those involving kernels.

Once normalized, a histogram can be thought of as a piecewise constant probability density function. Thus, generating a histogram comes down to finding a piecewise constant probability density function that is a reasonable estimate of a probability density function assumed to have generated the data at hand. Usually, this estimation is done by settling on a set of discontinuity points, which, in the context of histograms, are called *breakpoints*. The density between breakpoints is then chosen in accordance with the maximum likelihood estimate, that is, as the fraction of data that falls between the breakpoints. In this sense, generating a histogram consists of a discretization step and the definition of a multinomial distribution on the resulting bins.

Commonly, breakpoints are restricted to be equidistant and all effort is put in determining the distance between consecutive breakpoints. Around 1980, methods were proposed, and proven optimal under certain assumptions, to find a suitable bin width for given data (Scott 1979; Freedman and Diaconis 1981). However, histograms with constant bin width are known to be overly smooth at high density regions and they do not respond well to spikes (Denby and Mallows 2009). Moreover, histograms with constant bin width can be suggestive of features that are insignificant. These effects are demonstrated in Figure 1, which shows regular histograms of data sampled as follows, using R (R Core Team 2014).

```
R> x1 <- c (rnorm (1000, -6), rnorm (1000, 6))
```

```
R> x2 <- c (runif (50), runif (50, max=3))
```

For the bimodal distribution underlying `x1`, the binning of the histogram is too coarse. For the mixed uniform distribution underlying `x2`, the histogram gives an overly complex image.

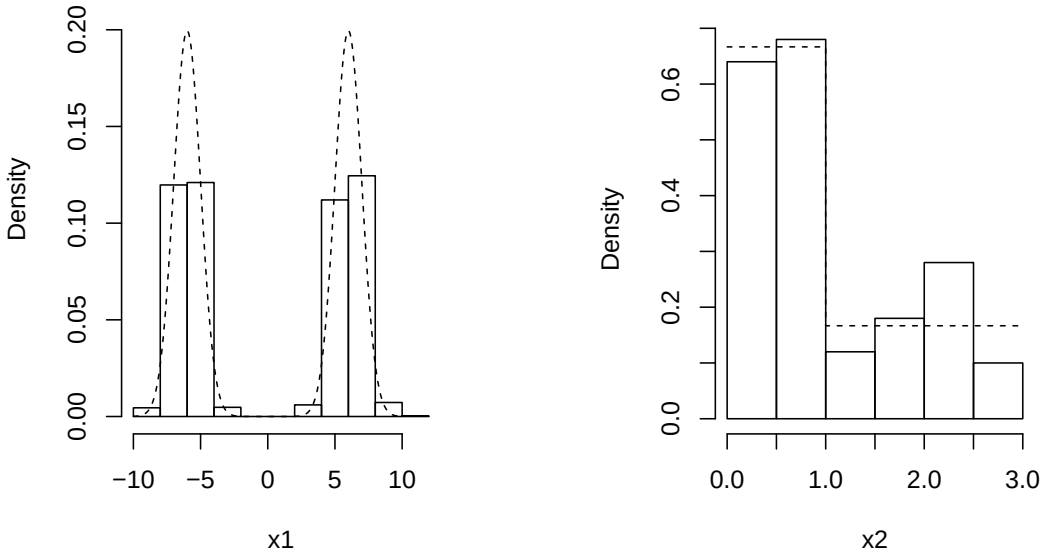

Figure 1: Regular histograms of data sampled according to mixed distributions. The generating density functions are shown using dashed lines.

As a remedy to the shortcomings just identified, we propose a method of determining breakpoints and densities that is based on local features of the data, instead of on global features such as the number of data points, the standard deviation (Scott 1979), or the inter-quartile range (Freedman and Diaconis 1981). Rather than looking at breakpoints in isolation, we will take the density between potential breakpoints into consideration too. Our piecewise constant density function will come about as a series of nested closed intervals to which densities are associated. Thus, our type of histogram is generated by the iterative selection of closed intervals, until some stopping criterion is met. The selection and stopping criteria are based on the compression of the data attainable using the constructed density function. While the application of complexity-based reasoning to density estimation is not new (Hall and Hannan 1988), we apply a novel generalization of the Kraft–McMillan theorem that allows us to accurately penalize the addition of extra breakpoints. In doing so, our approach is an application of the minimum description length principle (*MDL*) (Grünwald 2007).

2. Encoding with densities

In probability theory, MDL-based reasoning depends on the Kraft–McMillan theorem (Cover and Thomas 1991), which relates probabilities to code lengths. Although the usual formulation of the Kraft–McMillan theorem is about probability mass functions, the theorem can be generalized to cover probability density functions (Witteveen, Duivesteijn, Knobbe, and

Grünwald 2014). In effect, this allows MDL-based reasoning in settings with continuous random variables. We include the generalization of the Kraft–McMillan theorem here, together with an informal proof.

Theorem 1. *A function $\ell : \mathbb{R} \to \mathbb{R}$ that behaves like a code length is induced by a density function p on $\mathbb{R}$ as follows:*

$$\ell(x) = -\log p(x) + k,$$

where k is a discretization constant that goes to infinity as the fineness of the discretization increases.

Proof. By the *Radon–Nikodym theorem*, we can think of the probability of $x \in \mathbb{R}$ under p as $p(x)\mathrm{d}\lambda(x)$, where λ denotes the Lebesgue measure on $\mathbb{R}$. The Lebesgue measure is translation invariant, thus we can drop the dependency on x. This yields $p(x)\mathrm{d}\lambda$ as an expression for the probability of x under p. The *Kraft–McMillan theorem* now prescribes a function that behaves like a code length, of the form $\ell(x) = -\log(p(x)\mathrm{d}\lambda) = -\log p(x) - \log \mathrm{d}\lambda$. It is common to think of $\mathrm{d}\lambda$ as a limit of a sequence monotonically approaching zero from above. Using this convention, $-\log \mathrm{d}\lambda$ is the limit of a sequence monotonically approaching infinity. As the sequence does not depend on x, the term does not influence the behavior of $\ell(x)$, which justifies the expression in the theorem. □

Given some data and a probability density function, Theorem 1 enables us to determine, up to a constant, the code length needed to describe the data with respect to the probability density function. Denoting the data by ω and the probability density function by H, we will use $\ell_{\text{data}}(\omega \mid H)$ for this code length. Note that $\ell_{\text{data}}(\omega \mid H)$ is thus only defined up to an additive constant. If we restrict to piecewise constant H, we could say that $\ell_{\text{data}}(\omega \mid H)$ captures the complexity of data ω under histogram-like probability density function H. In order to apply MDL for the selection of a histogram-like probability density function, we confine our attention to a class of piecewise constant probability density functions for which we will define a complexity measure ℓ_{hist}. Selection then proceeds by trying to find a function H in this class that minimizes the two-part code length

$$\ell_{\text{hist}}(H) + \ell_{\text{data}}(\omega \mid H). \tag{1}$$

As said, our class will consist of probability density functions defined by nested closed intervals, where each interval is provided with a density. For mathematical convenience, we will not consider the specification of densities in our complexity measure ℓ_{hist}. In Section 4 we will see that our implementation allows for the reservation of a code length for the specification of densities up to some precision. Thus, we define ℓ_{hist} recursively as the code length of a specification of nested intervals. Given a closed interval $I \subset \mathbb{R}$, we use a single bit to encode whether or not there is an additional closed interval, $I' \subseteq I$. If there is, we encode it and recurse on it. Moreover, we also recurse on the outside of I'. That is, we transform I by contracting I' to a point and recurse on this transformed interval. For the specification of I' inside I, we use a uniform distribution on the space of possible interval endpoints inside I. The corresponding probability density function takes on the value $\frac{2}{\lambda(I)^2}$ on intervals inside I, hence, by Theorem 1, the code length of specifying I' this way is

$$-\log \frac{2}{\lambda(I)^2} + k_{\text{hist}}.$$

Thus, if the root of the nested intervals of some H is I and we designate, if applicable, the two constituents of the recursion by H_{in} and H_{out}, we get, in bits,

$$\ell_{\text{hist}}(H) = \begin{cases} 1 - \log \frac{2}{\lambda(I)^2} + k_{\text{hist}} + \ell_{\text{hist}}(H_{\text{in}}) + \ell_{\text{hist}}(H_{\text{out}}) & \text{with further nesting} \\ 1 & \text{otherwise} \end{cases}$$

Note that the above defines a proper code length to the class of nested intervals, instead of to the class of probability density functions based on nested intervals. However, we feel that this code adequately balances representing our interests and being mathematically pleasant to work with.

3. Estimation

For data ω, we are after histogram-like probability density functions H that minimize (1). We can approximate such H greedily, one interval at a time. Therefore, we want to find an $\hat{H}$ with only one level of nesting that minimizes $\ell_{\text{hist}}(\hat{H}) + \ell_{\text{data}}(\omega \mid \hat{H})$. Since $\ell_{\text{hist}}(\hat{H})$ takes on the same value for all $\hat{H}$ that have only one level of nesting, only $\ell_{\text{data}}(\omega \mid \hat{H})$ is relevant for our minimization. Now, let I be the root interval of $\hat{H}$ and assume I covers all of ω. Further, let I' be the interval inside I in $\hat{H}$, let n be the size of ω, and n_{in} the number of elements of ω that fall within I'. Setting $n_{\text{out}} = n - n_{\text{in}}$ and denoting by p_{in} and p_{out} the probability densities inside and outside I' respectively, we obtain, by Theorem 1,

$$\ell_{\text{data}}(\omega \mid \hat{H}) = -n_{\text{in}} \log p_{\text{in}} - n_{\text{out}} \log p_{\text{out}} + n k_{\text{data}}.$$

Note that probability densities are maximized, and the code length thus minimized, by taking I as small as possible and taking maximum likelihood estimates for p_{in} and p_{out}. Therefore, we take the extreme values of ω as endpoints of I and set p_{in} and p_{out} to $\frac{n_{\text{in}}}{\lambda(I')}$ and $\frac{n_{\text{out}}}{\lambda(I) - \lambda(I')}$ respectively. The main variable in our minimization thus becomes I'. Finding an optimal I' given ω is simplified by the following.

Assumption. We assume that the optimal I' is so that we have $p_{\text{in}} > p_{\text{out}}$. This holds for instance when a sufficiently smooth distribution with concave features underlies ω.

With this assumption in place, the optimal I' has elements of ω as its endpoints. Indeed, intervals of this kind maximize the inside data density. Under the above assumption, finding the interval with the lowest corresponding code length is possible using an algorithm based on Kruskal's Minimum Spanning Tree algorithm (Cormen, Leiserson, and Rivest 1991). Our algorithm, Algorithm 1, uses a disjoint-set data structure to keep track of locally optimal intervals and merges them to get to a globally optimal interval. As it is based on Kruskal's algorithm, our algorithm has a running time in $\mathcal{O}(n \log n)$, which we consider attractive.

We want to apply Algorithm 1 recursively and for that we need to know when to stop our recursion. Given an $\hat{H}$ that has a single level of nesting, we compare it to $\mathring{H}$ which has no nesting, and prefer the $H \in \{\hat{H}, \mathring{H}\}$ that roughly minimizes $\ell_{\text{hist}}(H) + \ell_{\text{data}}(\omega \mid H)$. We say roughly, since we will discard the cost of recursion in ℓ_{hist} because our candidate $\hat{H}$ was

Algorithm 1 BEST-INTERVAL(ω)

for each x **in** ω **do**
  MAKE-SET(x)
  $best[x] \leftarrow$ the interval covering only x
end for
{we assume ω is sorted}
for each neighbouring x, y **in** ω, in increasing order of distance between them, **do**
  $I_x \leftarrow best[\text{FIND-SET}(x)]$
  $I_y \leftarrow best[\text{FIND-SET}(y)]$
  $I_u \leftarrow$ the smallest interval covering I_x and I_y
  $u \leftarrow \text{UNION}(x, y)$
  {we abuse notation slightly by using I' in place of $\hat{H}$}
  $best[u] \leftarrow \arg\min_{I' \in \{I_x, I_y, I_u\}} \ell_{\text{data}}(\omega \mid I')$
end for
return $best[u]$

restricted to a single level of nesting a priori. With I, I', n, n_{in} and n_{out} as before, we have

$$\begin{aligned}\ell_{\text{hist}}(\hat{H}) + \ell_{\text{data}}(\omega \mid \hat{H}) &= 1 - \log \frac{2}{\lambda(I)^2} + k_{\text{hist}} \\ &\quad - n_{\text{in}} \log \frac{n_{\text{in}}}{\lambda(I')} - n_{\text{out}} \log \frac{n_{\text{out}}}{\lambda(I) - \lambda(I')} + n k_{\text{data}}, \\ \ell_{\text{hist}}(\mathring{H}) + \ell_{\text{data}}(\omega \mid \mathring{H}) &= 1 - n \log \frac{n}{\lambda(I)} + n k_{\text{data}}.\end{aligned}$$

Note how a comparison of $\hat{H}$ and $\mathring{H}$ is possible without deciding on k_{data}, as it influences both code lengths in the same way. We set k_{hist} to $-2 \log \frac{\lambda(I)}{n}$, for which Witteveen *et al.* (2014) give a rationale. Consequently, the difference between the above two code lengths becomes

$$- \log \frac{2}{n^2} - n_{\text{in}} \log \frac{n_{\text{in}}}{\lambda(I')} - n_{\text{out}} \log \frac{n_{\text{out}}}{\lambda(I) - \lambda(I')} + n \log \frac{n}{\lambda(I)}. \tag{2}$$

If, for some I', this difference turns out negative, I' is deemed informative and we will proceed with our recursion. The resulting procedure is included as Algorithm 2, which computes a histogram-like probability density function fitting data ω.

4. Analysis

The method outlined in Algorithm 2 follows the divide and conquer paradigm. Precisely how often the algorithm recurses depends on the whimsicality of the data. The more there is to show about the data, the longer it takes to generate our nested intervals. Nevertheless, the running time of Algorithm 1 ensures that our method has a worst case running time that is polynomial in the number of data points.

The procedure of estimating histogram-like probability density functions as described in the previous section is implemented in the R package **histmdl**. Using two parameters, it is possible to tune the behavior of the `histmdl` function. These parameters mainly serve to filter spurious spikes. The first parameter is `support=`, which is responsible for lower bounding the number

Algorithm 2 RECURSIVE-INTERVALS(ω)

Initialize H with the smallest interval covering ω
$I' \leftarrow$ BEST-INTERVAL(ω)
if I' makes (2) negative **then**
  recurse on $\omega \cap I'$
  contract $\omega \cap I'$ to a single point and recurse
  undo the contraction and add the recursion outcomes to H
else
  ignore I', as it is not informative
end if
return H

of data points in every interval. The second parameter is `gain=`, which is added to (2) and thus places a lower bound on the compression obtained per additional interval. This parameter enables us to account for the code length needed to specify the densities inside intervals, which we ignored in our framework.

Even though our examples in Section 1 were somewhat contrived, it is nice to see that our new visualization successfully addresses the shortcomings we have identified.

```
R> library (histmdl)
R> histmdl (x1, gain=2)
R> histmdl (x2)
```

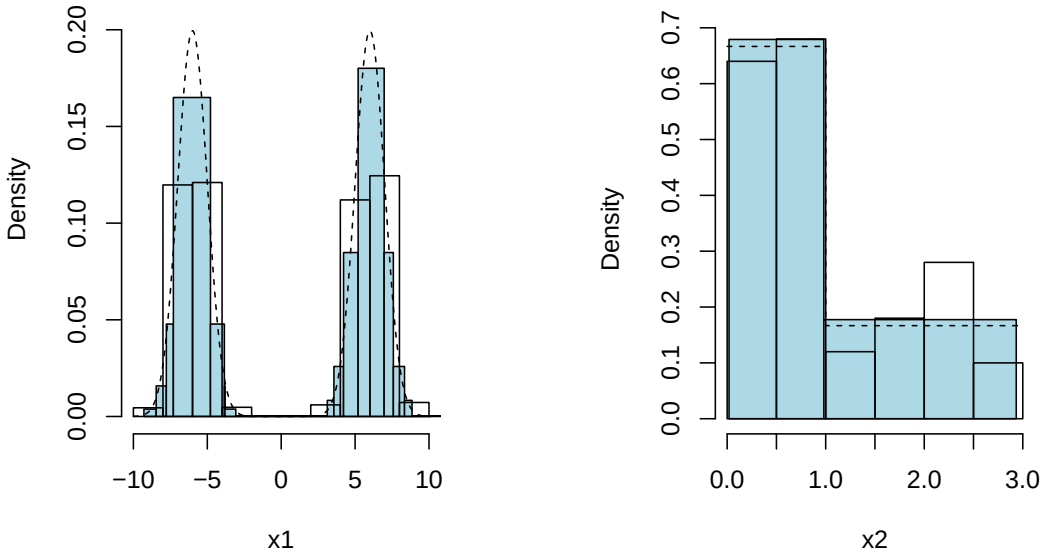

Figure 2: Our new histogram-like visualizations added to Figure 1.

In generating the Figure 2, we have increased the `gain=` parameter a little for data `x1` to filter two spikes covering 4 and 8 elements respectively. Using only 9 bins instead of 11, as

used by the classical histogram, our new visualization for `x1` follows the generating density function more closely than the classical histogram. The new visualization manages to increase detail precisely where relevant. For `x2`, the new visualization closely reconstructs the density function that was used to generate the data. This is not entirely surprising, as the class of piecewise constant probability density functions which our method deals with consists of probability density functions just like the one we used to generate `x2`. Yet the new visualization slightly overestimates the densities. This is to be expected by the greedy nature of Algorithm 1, which was motivated by an urge to maximize the likelihood of the data. Consequently, the new visualization for `x2` is based on an interval $[0.019, 2.934]$ with a subinterval $[0.019, 0.981]$. Indeed, these intervals are a little narrower than $[0, 3]$ and $[0, 1]$, respectively.

For a less contrived demonstration, we turn to a small real-world dataset: eruption data of the Old Faithful geyser. It is known that the dataset is a little distorted. Rounding and measurement errors are both present. To overcome the rounding, we reconstruct the eruption duration in seconds. Although the data is thus discrete, there are 124 different values, most occurring only once, which is enough to favor our approach for continuous data. The resulting Figure 3 does not give an indication to worry much about measurement errors.

```
R> histmdl (round (60 * faithful$eruptions))
R> hist (round (60 * faithful$eruptions))
```

What catches the eye in Figure 3 is the spike between 107 and 113 seconds. This spike covers 30 eruptions and is hardly present in the ordinary histogram. Contrary to seeing similar spikes in ordinary histograms, such spikes in histogram-like visualizations of our new kind deserve attention. After all, they survived a filter on their contribution to compressibility. In other words, they were deemed significant by the method. We will not research this particular spike further, but do remark that such observations add to the value of our histogram-like visualizations.

5. Summary

We identified two shortcomings in ordinary histograms, namely that they easily become too coarse for data sampled from multimodal distributions, and that they can present an overly complex image depicting insignificant density fluctuations. As a remedy, we introduced a class of histogram-like probability density functions and an associated method for the estimation of such functions given some data. This method is implemented in the R package **histmdl** and is shown able to overcome the shortcomings identified.

Acknowledgements

We thank Richard Gill for preaching the R language.

References

Cormen TH, Leiserson CE, Rivest (1991). *Introduction to Algorithms*. 1 edition. The MIT Press.

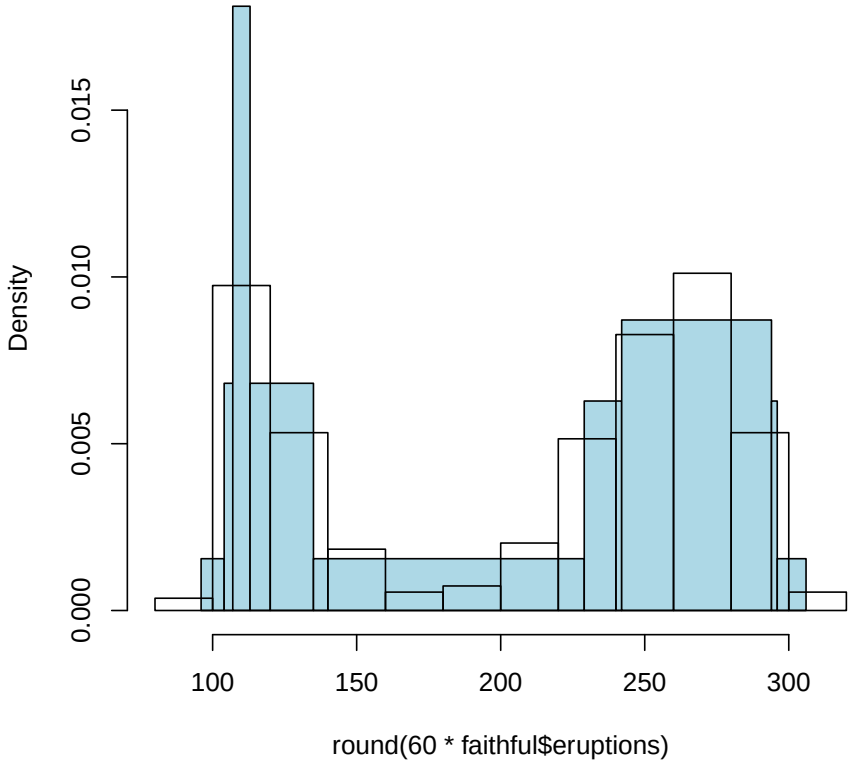

Figure 3: An ordinary histogram and our new histogram-like visualization of the eruption durations in seconds of the Old Faithful geyser.

Cover TM, Thomas JA (1991). *Elements of Information Theory*. Wiley-Interscience.

Denby L, Mallows C (2009). "Variations on the Histogram." *Journal of Computational and Graphical Statistics*, **18**(1), 21–31.

Freedman D, Diaconis P (1981). "On the Histogram as a Density Estimator: L_2 Theory." *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, **57**.

Grünwald PD (2007). *The Minimum Description Length Principle*. The MIT Press.

Hall P, Hannan EJ (1988). "On Stochastic Complexity and Nonparametric Density Estimation." *Biometrika*, **75**(4), 705–714.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Scott DW (1979). "On Optimal and Data-Based Histograms." *Biometrika*, **66**.

Witteveen J, Duivesteijn W, Knobbe A, Grünwald P (2014). "RealKrimp – Finding Hyperintervals That Compress With MDL for Real-Valued Data." In *Advances in Intelligent Data Analysis XIII*, volume 8819 of *Lecture Notes in Computer Science*, pp. 368–379. Springer International Publishing.

Affiliation:

Jouke Witteveen
Institute for Logic, Language and Computation
Universiteit van Amsterdam
1090 GE, Amsterdam, the Netherlands
E-mail: j.e.witteveen@uva.nl
URL: https://staff.fnwi.uva.nl/j.e.witteveen/