





Uncertainty Treemaps

Max Sondag* 
TU Eindhoven

Wouter Meulemans* 
TU Eindhoven

Christoph Schulz† 
University of Stuttgart

Kevin Verbeek* 
TU Eindhoven

Daniel Weiskopf† 
University of Stuttgart

Bettina Speckmann* 
TU Eindhoven

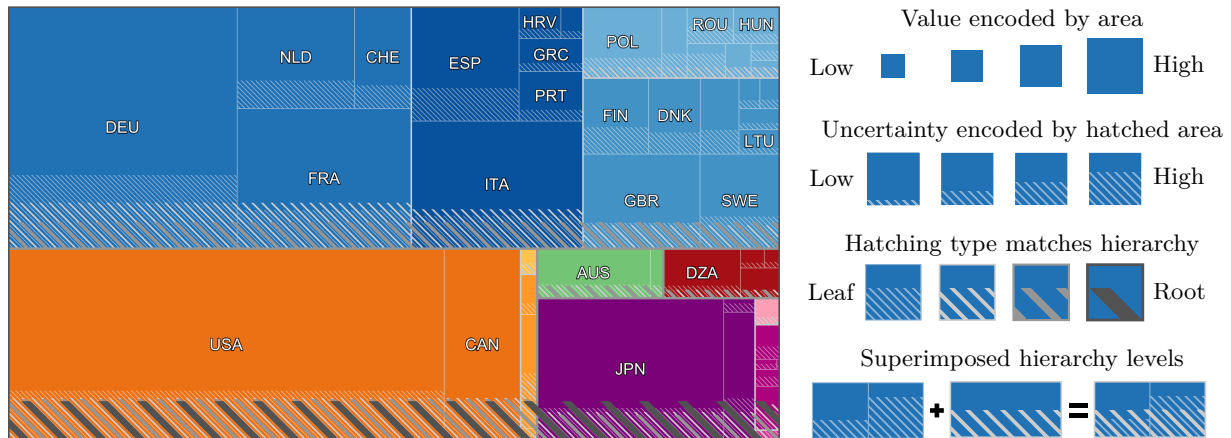


Figure 1: Uncertainty Treemap of coffee imports from 1994 to 2014, decomposed into continents, subregions, and countries [39]. Our visualization encodes uncertainty using nested hatched lines in the areas of the corresponding values. Of the North American countries (■), the United States (USA) import much more coffee than Canada (CAN). While Canada has much less uncertainty in its value (area of ■), both countries share a similar relative fluctuation (height of ■ relative to containing rectangles). Similarly, going up two levels (■) reveals that Europe (■, ■, ■, ■) has lower relative fluctuation than the Americas (■, ■).

ABSTRACT

Rectangular treemaps visualize hierarchical numerical data by recursively partitioning an input rectangle into smaller rectangles whose areas match the data. Numerical data often has uncertainty associated with it. To visualize uncertainty in a rectangular treemap, we identify two conflicting key requirements: (i) to assess the data value of a node in the hierarchy, the area of its rectangle should directly match its data value, and (ii) to facilitate comparison between data and uncertainty, uncertainty should be encoded using the same visual variable as the data, that is, area. We present *Uncertainty Treemaps*, which meet both requirements simultaneously by introducing the concept of *hierarchical uncertainty masks*. First, we define a new cost function that measures the quality of Uncertainty Treemaps. Then, we show how to adapt existing treemapping algorithms to support uncertainty masks. Finally, we demonstrate the usefulness and quality of our technique through an expert review and a computational experiment on real-world datasets.

Index Terms: Human-centered computing—Visualization—Visualization techniques—Treemaps

1 INTRODUCTION

Treemaps are a well-established method to visualize hierarchical numerical data. A treemap recursively partitions a two-dimensional shape into regions whose areas correspond to the input data. There

are a variety of treemaps using different shapes, such as Voronoi Treemaps [3], Orthoconvex and L-shaped Treemaps [13], and Jigsaw Treemaps [41]. Here, we focus on rectangular treemapping algorithms that partition an input rectangle R into a set of smaller rectangles, such that the area of each such rectangle corresponds to the data value associated with the corresponding leaf in the input hierarchy. Furthermore, the rectangles associated with the children of an interior node of the hierarchy form again a rectangle, that is associated with their parent and has the area corresponding to the node value. It has been shown that rectangular treemaps satisfy important design consideration for the visualization of hierarchical data: they present the input data in a distortion-free manner, they clearly visualize the input hierarchy, and they are maximally space-efficient [25, 29].

Numerical data often has uncertainty (measurement, completeness, inference) associated with each data value. When plotting singular data values, it is straightforward to indicate the error, for example, using error bars. Such visualizations anchor the uncertainty at the data value. While this strategy works well for low-dimensional data, it does not lend itself directly to more complex or higher-dimensional data. In the context of treemaps, an additional challenge is the fact that uncertainty does not aggregate in the same way as data values do: relative uncertainty tends to become smaller, the higher a node is in the hierarchy. Despite these challenges, any visualization that aims to be trustworthy and faithful to the data it represents has to visualize uncertainty together with the data. Nevertheless, techniques that support plotting certain and uncertain data simultaneously for more complex data types are rather scarce [20].

In this paper we show how to visualize both hierarchical data and their associated uncertainty in rectangular treemaps. To do so, we identify two conflicting key requirements:

Visual Aggregation. To assess the value of an interior node, the

*e-mail: {m.f.m.sondag,w.meulemans,k.a.b.verbeek,b.speckmann}@tue.nl

†e-mail: {christoph.schulz,daniel.weiskopf}@visus.uni-stuttgart.de

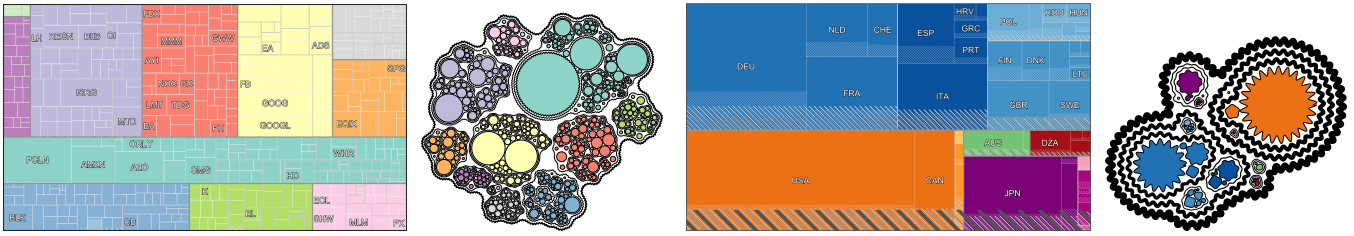


Figure 2: Uncertainty Treemaps and Bubble Treemaps [15]. Left: the S&P dataset [15]. The Uncertainty Treemap shows that this dataset has only very few data points with high uncertainty, whereas this is harder to establish for the Bubble Treemap. Right: the Coffee dataset [35]. The Bubble Treemap produces thick and folded boundaries that prohibit value comparison; in contrast, the Uncertainty Treemap indicates the aggregated uncertainty for each level in the hierarchy.

area of its rectangle (or generally, its visual size) should directly match its value.

Uncertainty Encoding by Area. To facilitate comparison between data and uncertainty, uncertainty should be encoded using the same visual variable as the data, that is, area.

Either requirement is straightforward to satisfy in isolation. For example, visual aggregation can be maintained if uncertainty is encoded using a sequential colormap or any other visual variable that does not hamper the perception of area. Similarly, uncertainty can be encoded using additional area adjacent to each rectangle. Doing so, however, will make visual aggregation next to impossible, as an inner node will have additional area inside its rectangle. In contrast to these decoupled approaches, our technique meets both requirements simultaneously. As an integral part of our uncertainty Treemaps, we introduce *hierarchical uncertainty masks* as a tool to visualize data and uncertainty in the same visual space.

Definitions and Notation. Let T be a rooted tree. Each node $v \in T$ has a set of children $C(v)$. A node without children is a leaf; non-leaf nodes are interior nodes. We denote the value of each node v in T by $\mu(v)$ and we have $\mu(v) = \sum_{u \in C(v)} \mu(u)$ for any interior node v . We assume that this value μ corresponds to the mean and the sum of means, respectively. We denote the uncertainty of a node $v \in T$ by $\sigma(v)$. Generally, the uncertainty of an interior node can be derived from its children. We assume that the uncertainty is the standard deviation and that the children are independent, thus $\sigma(v) = \sqrt{\sum_{u \in C(v)} \sigma(u)^2}$ for an interior node v . Hence, the relative uncertainty $\frac{\sigma(v)}{\mu(v)}$ tends to become smaller higher in the hierarchy.

Given an input rectangle R and a hierarchy T , a treemap layout (algorithm) recursively partitions R into rectangles. We describe this layout using a function R_μ that maps each node $v \in T$ to a rectangle with area $\mu(v)$. $R_\mu(v_{root})$ maps to R .¹ The rectangle for each interior node v is the disjoint union of the rectangles of its children. The quality of a treemap is typically measured via the aspect ratio of the rectangles R_μ and denoted by $\rho(R_\mu(v))$ for a node v .

Contribution and Organization. Uncertainty Treemaps visualize the uncertainty in the same space as the data, by using so-called *uncertainty masks*. These masks anchor a region $R_\sigma(v)$ of area $\sigma(v)$ to $R_\mu(v)$ for each node v . Since R_μ already covers the graphical space, we require that $R_\sigma(v) \subseteq R_\mu(v)$. Our design is based on screen-door transparency and renders the regions $R_\sigma(v)$ on top of the regions $R_\mu(v)$. In Section 3, we describe the design of our uncertainty masks in greater detail and explain how to overlay the masks hierarchically on the treemap layout R_μ . We also show how to answer questions on real-world data using Uncertainty Treemaps

¹For simplicity, we assume that μ is normalized such that $\mu(v)$ equals the area of the input rectangle R for the root v_{root} of T .

and summarize the result of an expert review. Furthermore, we introduce a new quality metric that measures the quality of a mask.

While our hierarchical uncertainty masks can be applied to any treemap layout computed with any treemapping algorithm, certain layouts result in better mask quality. To compute Uncertainty Treemaps with high mask quality, we could focus on this optimization criterion, while ignoring the aspect ratio of the rectangles—the most common quality criterion for rectangular treemaps. However, the (summed) mean values still need to remain legible and comparable. Thus, in Section 4, we show how to adapt existing treemapping algorithms to take the uncertainty masks into account. Here, we distinguish two types of algorithms: *mask-friendly* algorithms that use only the fact that a mask will be placed and *mask-aware* algorithms that use the uncertainty values to compute the layout.

In Section 5, we experimentally compare variants of our mask-quality metric to establish how well these measures are able to capture different aspects of quality. Furthermore, we investigate the effectiveness of mask-friendly and mask-aware algorithms on several real-world datasets. Finally, in Section 6, we discuss various design alternatives for the shape, placement, and rendering of the masks, as well as the effects of uncertainty and unbalanced hierarchies.

2 RELATED WORK

There are many different methods to visualize hierarchical data, Schulz *et al.* [33] provide an extensive survey of the entire range of techniques. We focus specifically on treemapping methods, which show the data values of a hierarchy using the area of nested shapes. Most treemapping techniques, such as Squarified Treemaps [9], Circular Treemaps [40], Voronoi Treemaps [3], or GosperMap [1] implement a rather straightforward tree traversal to subdivide space for the layout. However, there are other techniques, such as treemaps with bounded aspect ratio [13], Stable Treemaps [35], or Bubble Treemaps [15] that attempt to optimize perceptually effective quantities. We count our present paper among such techniques.

In general, we may derive uncertainty from temporal variation of a numerical value. Uncertainty Treemaps can then straightforwardly visualize temporally varying data in a static hierarchy. Although this representation loses information about temporal patterns beyond the magnitude of change, Uncertainty Treemaps provide a simple, static overview visualization, removing the need for animation.

Uncertainty visualization is a broad field, ranging from data acquisition to mapping to representation, and reasoning about uncertainty [8, 21]. We consider uncertainty visualization to be a process of multiplexing certain and uncertain data in such a way that humans can successfully demultiplex it [10, 19] and reason about it [16, 31]. Within this information-theoretical consideration, we can distinguish between encoder, channel, and decoder. Correspondingly, an uncertainty-aware treemap has to have at least three channels, i.e., visual variables, to communicate tree structure, certain data, and uncertain data simultaneously. We map both data value and uncertainty to size, and use texture to separate value from uncertainty.

Bertin [5] pioneered the study of visual variables many years ago. However, the study of those variables in the context of uncertainty visualization is a fairly recent trend. Specifically, in the context of maps and graph edges, fuzziness appears to be an intuitive and sufficient choice [18, 27]. Clean geometry, e.g., boxplot whiskers, appears to be slightly less prone to error in general [17, 27]. However, there are some pitfalls such as unprofessional appearance [7] and the within-box bias [12]. These visual variables are not suitable for our technique because we have to depict two types of data in one area, i.e., we have to find an uncertainty-aware blending operator.

Holliman *et al.* [19] recently coined the term *Visual Entropy*. Here, the idea is to couple high noise with an expected frequency to communicate uncertainty. Using white noise to depict uncertainty was evaluated and recommended in the context of maps [24]. Kale *et al.* [22] show that animation is also viable for framing uncertainty as frequency. To us, screen-door transparency seems particularly promising, since it avoids color blending issues without requiring additional space [23, 32]. Bair *et al.* [2] show that the amount of fully overlapping terrain-like layers and choice of texture strongly influence the separability of those layers. Thus, our work adapts various types of screen-door transparency while preventing data from being fully concealed by the uncertainty, since partial overlap influences separability to a lesser degree.

Slingsby *et al.* [34] use a geospatial variant of treemaps in an interactive system, to support the exploration of uncertainty in the context of geospatial data. In contrast to our Uncertainty Treemaps, their approach does not anchor data value to uncertainty and relies on interaction to explore uncertainty. Most closely related to our work are the Bubble Treemaps proposed by Görtler *et al.* [15]—a variant of circular treemaps that can depict uncertainty, e.g., using modulated splines on region boundaries. Their approach separates the encoding of certain and uncertain aspects of data, which also hampers with comparison due to missing alignment and anchoring. Furthermore, Bubble Treemaps require ample white space and hence deliberately violate the visual aggregation requirement. See Figure 2 for a visual comparison between Uncertainty Treemaps and Bubble Treemaps. We argue that Uncertainty Treemaps provide a clearer and cleaner picture of the data, in particular, because our approach does not introduce unequally distributed whitespace to compensate for conflicting propagation models.

3 HIERARCHICAL UNCERTAINTY MASKS

Our hierarchical uncertainty masks are key to enabling Uncertainty Treemaps, in which $R_\sigma(v) \subseteq R_\mu(v)$ for any node v . Here, we first discuss the design of our uncertainty masks, then show how to read the resulting Uncertainty Treemaps, report the results of a brief expert review, and finally describe how we measure mask quality.

3.1 Mask Design

Consider some (rectangular) treemap layout described by R_μ . We want to augment the layout with regions dedicated to showing σ : the region $R_\sigma(v)$ of a node v must be contained in $R_\mu(v)$ and have size $\sigma(v)$. Due to the recursive nature of treemaps, the regions $R_\sigma(v)$ overlap the rectangles (and masks) of nodes lower in the hierarchy. We must thus consider how to effectively render the masked regions, to ensure that both R_μ and R_σ remain visible.

In terms of placement, our mask spans a fraction of $\sigma(v)/\mu(v)$ of $R_\mu(v)$.² The mask could be essentially of arbitrary shape as long as it has the correct area as prescribed by σ . However, it is natural to use rectangles for the mask as well, since the mask is integrated into a rectangular treemap. In particular, we place the mask as a rectangle at the bottom of the node along the full width of the node; Figure 3 (left) and (middle) show masks for a single node.

²The observant reader will notice that for $\sigma(v) > \mu(v)$, this may be problematic. We refer to Section 6.3 for a brief discussion of this issue.



Figure 3: Hatched uncertainty masks: (left) a leaf node, (middle) a node above leaf level doubles line width and spacing, (right) a 2-level treemap with 4 leaves.

Figure 3 also illustrates that we render the masks using hatching of slanted, parallel, equidistant lines. The line width and gap depend on the node’s hierarchy level, both doubling with each higher level. The line gap is three times the line width, such that at least every other line is fully visible. Figure 3 (right) illustrates the overlay of two masks at the bottom of the treemap: both the lower-level masks in R_σ as well as the full extent of rectangles in R_μ remain visible. Note that, contrary to other approaches, the pattern density does *not* encode the uncertainty value but matches the hierarchy level.

We place the hatching such that mask lines at a certain level coincide with half of the mask lines one level lower. To this end, we define the hatching pattern globally within R . The mask for a specific node v is then the intersection of $R_\sigma(v)$ with the global mask, which ensures that the intended visibility through the masks is maintained regardless of the positioning of the nodes. A discussion of alternative mask shapes and patterns can be found in Section 6.

3.2 Reading an Uncertainty Treemap

Figure 1 shows an Uncertainty Treemap of the Coffee dataset [35]: the mean amount of coffee imported yearly per country between 1994 and 2014, with the associated standard deviation [39]. We illustrate our technique by answering an example question. Section A of the supplementary material contains additional example questions as well as a basic guide on how to read Uncertainty Treemaps.

How do Europe (■, ■, ■, ■) and North America (■) compare in terms of mean import of coffee as well as absolute and relative fluctuation? The rectangles for Europe (EU) and North America (NA) have different width and height, so we rely on area to assess summed mean and absolute standard deviation. As there is significantly more blue area (■, ■, ■, ■) than dark orange area (■), EU imports more coffee on average. For assessing absolute fluctuation for EU, we look at the hatching at continent level (■), which spans the entire width of the treemap. NA is one level lower in the hierarchy, and thus we consider the subcontinent level hatching (■). These areas show slightly higher fluctuation in NA, despite EU having a larger mean.

In relative terms, we consider the same hatched areas with respect to their containing rectangles. We see that EU has approximately 15% relative fluctuation, whereas NA has about a fourth: NA has higher relative fluctuation. Note that we can always rely purely on the height of the hatched areas and their containing rectangles to infer the relative fluctuation, as they have the same width by design.

3.3 Expert Review

To investigate whether our uncertainty masks yield usable treemaps, we conducted a review of our method with a visualization and perception expert. The expert was not involved with, or informed of, our new method before the review session, but is generally knowledgeable about, and familiar with, treemaps. Below, we briefly summarize the results of this review session; we refer to the supplementary material (Section B) for details.

The expert review suggests that Uncertainty Treemaps are usable and potentially preferable over two other methods (Bubble Treemaps [15] and a Juxtaposed variant). The main issue identified is that hatching may trigger height assessment even when this is not

appropriate. This can be attributed to a learning curve—which is to be expected for any treemap representation, according to the expert. We made minor changes to the design to reduce such risks.

3.4 Mask Quality

Hierarchical uncertainty masks can be applied to any treemap layout. However, the layout has a large effect on the readability of the masks, as the mask of an interior node is rendered on top of its descendants. Though aspect ratio is a prominent measure for assessing treemap layouts, our masks are immediately derived from such a layout and relative uncertainty can be derived from height only. Therefore, we introduce a new measure for mask quality of a given layout, to enable comparison and optimization, that more closely ties to the hierarchical nature of the masks.

Generally, we prefer the mask of a node to overlap few of its children to increase the visibility of the individual child nodes. However, mask overlap cannot be avoided due to the partitioning nature of treemaps. When a child mask extends beyond its parent, it is more clearly visible as no mask is rendered on top of it. This also matches the fact that relative uncertainty decreases higher in the hierarchy. In other words, a child mask extending beyond its parent reduces the number of layers that a reader must see through to assess the area, which has been shown to be beneficial for the separability of layered terrain-like surfaces [2]. Our metric hence captures how much the mask of a parent extends beyond the mask of its child nodes, as this hides the lower-level masks.

Node Quality. To capture this idea for a single node u , we define its *excess overlap* with respect to its parent or ancestor v . The excess overlap that v causes for u is the part of the rectangle of u that is covered by the mask of v but not by the mask of u itself: $\varepsilon(u, v) = (R_\mu(u) \cap R_\sigma(v)) \setminus R_\sigma(u)$ (see Figure 4). Let $S(u)$ be the set of the relevant ancestors of u . We define the excess overlap of u as $EO(u) = \sum_{v \in S(u)} |\varepsilon(u, v)|$. We consider two options for the set $S(u)$: either it contains all ancestors of u (option A) or it contains only the parent of u (option P). Although option P is less complex, it may miss repeated excess overlap by many ancestors.

Treemap Quality. We now derive a mask quality measure for an entire layout using excess overlap. There are multiple options for aggregating the values of individual nodes. The excess overlap measures the area of overlap, typically putting more focus on large nodes. We can either choose to directly aggregate the area or size (option S) of the excess overlap of individual nodes or to first normalize (option N) the excess overlap of each node u by dividing $EO(u)$ by $\mu(u)$. Although option S captures visually salient excess overlap, it could result in hiding that many smaller nodes have excess overlap. If each node is equally important, regardless of area, then option N is more adequate to capture the overall visibility of the masks.

The various options lead to four different quality measures for an individual node u , which we denote by $EO_{xy}(u)$, where $x \in \{A, P\}$ and $y \in \{S, N\}$. For example, $EO_{AS}(u)$ indicates the excess overlap of u measured with respect to all ancestors (A) of u , using the size

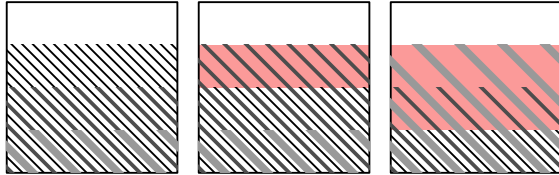


Figure 4: Examples of mask quality, with excess overlap indicated by the shaded region. (Left) Properly nested: each lower-level mask (thinner lines) extends beyond the higher-level masks. (Middle) The middle-level mask extends beyond the lowest-level mask. (Right) Each higher-level mask extends beyond the lower level masks.

(S) of the overlap directly; $EO_{PN}(u)$ indicates the excess overlap of u measured only with respect to the parent (P) of u , normalized (N) by the area $\mu(u)$ of u . Typically, we drop the argument u when we talk about the quality measure in general.

Using these measures, we can compute mask quality for an entire layout by taking the average or the maximum over all nodes in the hierarchy. In Section 5, we study the correlation between these measures for rectangular treemaps. Note that these quality measures can be directly applied to any type of treemap and mask shape.

4 ALGORITHMS FOR UNCERTAINTY TREEMAPS

There exist many different treemapping algorithms for hierarchical data without uncertainty, typically focusing on optimizing the aspect ratio of the individual rectangles in the layout. When overlaying hierarchical uncertainty maps, we should also consider mask quality (Section 3.4). However, the primary data values need to remain legible and comparable and thus we cannot focus solely on mask quality; indeed aspect ratio for the rectangles remains a primary concern. The resulting problem is thus a bicriteria optimization problem: we want to optimize the aspect ratios but additionally achieve high quality of the uncertainty masks. We therefore adapt existing treemapping algorithms to take the uncertainty masks into account. Treemapping algorithms often include choices that do not influence the final aspect ratios of the resulting treemap. We aim to modify such choices to improve the quality of the masks. We can distinguish two types of mask-based modifications to the algorithms:

Mask-friendly: the algorithm uses the fact that the mask will be placed at the bottom of a rectangle but does not consider the actual uncertainty values.

Mask-aware: the algorithm uses the data uncertainty values to compute the layout.

The degree to which we can make a particular treemapping algorithm mask-friendly or mask-aware depends on how many choices are arbitrary regarding the aspect ratios and thus how much freedom the algorithm offers. We illustrate our approach with the algorithm that arguably offers the most freedom: the Approximation algorithm.

Approximation. The Approximation algorithm [30] works as follows. Consider a node v with a given rectangle $R_\mu(v)$. Let u_1, \dots, u_d denote the d children of v , sorted in decreasing order by value. Let k denote the smallest number such that $\sum_{i=1}^k \mu(u_i) \geq \frac{1}{3} \mu(v)$. Let $A = \{u_1, \dots, u_k\}$ and $B = \{u_{k+1}, \dots, u_d\}$ denote the resulting partition. We now split $R_\mu(v)$ into two rectangular containers of the correct size: one for A and one for B . If the width of $R_\mu(v)$ exceeds its height, this split is *horizontal*: the two containers are side-by-side, where A is assigned to the left container and B to the right. Otherwise, the split is *vertical*: one container is above the other, with the upper one being assigned to A and the lower to B . This strategy is then recursively applied to the two containers, where we may now consider the container as a “virtual node” with the assigned children. When the container has only a single child u , it defines $R_\mu(u)$ and we may continue the recursion on u if it is not a leaf. It can be shown that, using this approach, all aspect ratios are bounded by $\max\{\rho(R_\mu(v)), 3, 1 + \max_{1 \leq i < d} \mu(u_{i+1})/\mu(u_i)\}$ [30].

4.1 Mask-Friendly Algorithms

Mirroring a given treemap horizontally or vertically (or any of the rectangles of interior nodes) does not affect the aspect ratios of the rectangles in the treemap. However, it can affect the quality of the uncertainty mask. Some algorithms construct their layout based on the order of the children of a given node by area/value, typically placing the rectangles in a left-to-right and/or top-to-bottom pattern. Such algorithms naturally place children with small area at the bottom. However, since the uncertainty mask is drawn at the bottom of rectangles, the mask will typically overlap with the smaller children, resulting in a lower-quality mask.

Algorithm 1 MFAPPROXIMATION(U, R)

Input: U is a set of d nodes with the same parent and R is a rectangle of size $\sum_{u \in U} \mu(u)$.
Output: a treemap layout R_μ for U and their descendants inside R .

- 1: **if** $d = 1$ **then**
- 2: $u \leftarrow$ the single node in U
- 3: Set $R_\mu(u) = R$
- 4: MFAPPROXIMATION($C(u), R$)
- 5: **else**
- 6: $s, A, B \leftarrow$ APPROXIMATIONSPPLIT(U, R)
- 7: **if** $s = \text{vertical}$ **then**
- 8: $R_A, R_B \leftarrow$ vertical split of R with R_A below R_B
- 9: **else**
- 10: $R_A, R_B \leftarrow$ horizontal split of R with R_A left of R_B
- 11: MFAPPROXIMATION(A, R_A)
- 12: MFAPPROXIMATION(B, R_B)

Algorithm 2 APPROXIMATIONSPPLIT(U, R)

Input: U is a set of $d > 1$ nodes; R is a rectangle of size $\sum_{u \in U} \mu(u)$.
Output: a split direction and two sets A and B , such that $A \cup B = U$, $A \cap B = \emptyset$.

- 1: Sort $U = \{u_1, \dots, u_d\}$ by decreasing μ value
- 2: $k \leftarrow$ smallest value such that $\sum_{i=1}^k \mu(u_i) \geq \frac{1}{3} \sum_{u \in U} \mu(u)$
- 3: $A, B \leftarrow \{u_1, \dots, u_k\}, \{u_{k+1}, \dots, u_d\}$
- 4: **if** height of R is greater than its width **then**
- 5: **return** vertical, A, B
- 6: **else**
- 7: **return** horizontal, A, B

We can avoid this problem by mirroring the construction vertically, placing larger children at the bottom and smaller children at the top. Our experiments (Section 5) show that this approach can be expected to generally improve mask quality. This very simple approach can readily be applied to many different treemapping algorithms, making them mask-friendly.

We illustrate this approach explicitly for the Approximation algorithm. When the original algorithm intends to make a vertical split, then the larger children (set A) are placed above the smaller children (set B). We can easily mirror this approach by placing B above A instead, without affecting aspect ratio. For a horizontal split, we do not need to change anything. The pseudocode for this mask-friendly version of the Approximation algorithm can be found in Algorithm 1 and Algorithm 2. The only change to the original algorithm is on line 8 of Algorithm 1, where “below” is used instead of “above”. Given a tree T with root v and input rectangle R , we compute a layout by calling MFAPPROXIMATION($C(v), R$).

4.2 Mask-Aware Algorithms

The ability of mask-aware algorithms to use the data uncertainty values opens up many possibilities to optimize mask quality. It allows us to directly use one of the quality measures for masks (Section 3.4) to guide the construction of the layout. However, optimizing for mask quality is difficult and likely cannot be computed efficiently. Nonetheless, we can use a mask quality measure to make more informed heuristic choices in the treemapping algorithm.

We illustrate this approach by adapting again the Approximation algorithm to make a mask-aware variant: we explore the choices we can make in the algorithm further. The main choice involves the partitioning of the children u_1, \dots, u_d (ordered decreasingly by μ) of a node v into A and B . We recall that the aspect ratio of the final rectangles in the treemap is bounded by $\max\{\rho(R_\mu(v)), 3, 1 + \max_{1 \leq i < d} \mu(u_{i+1})/\mu(u_i)\}$. Thus, as long as

the same bound applies to the subsets of children A and B (in particular, the ratios of consecutive values) and the aspect ratios of their container rectangles, the resulting partition will satisfy the aspect-ratio bound regardless of the choice of A and B . We call a split (that is, the sets A and B along with the splitting direction) *valid* if it satisfies these properties. By allowing all valid splits, we have more options to optimize mask quality. To increase flexibility even further, we introduce a parameter $q \geq 3$, and relax the aspect ratio bound to $\max\{\rho(R_\mu(v)), q, 1 + \max_{1 \leq i < d} \mu(u_{i+1})/\mu(u_i)\}$.

Below we describe the modifications in detail. The overall algorithm remains the same, but we replace the splitting algorithm APPROXIMATIONSPPLIT by MASKAWARESPPLIT (Algorithm 3), which attempts to find a valid split that optimizes mask quality.

Choosing Splits. As horizontal splits always cause both container rectangles to overlap the mask of v , we prefer vertical splits over horizontal splits. With a vertical split, the mask of v may overlap rectangles and masks in the bottom container. However, if the bottom container is large enough, the mask of v will not overlap the top container. Thus, we first attempt to find a valid vertical split and use a horizontal split only if no valid vertical split exists.

Vertical Splits. As before, let u_1, \dots, u_k be the children of a node v in decreasing order on the μ value. We initially consider a partition $A = \{u_1, \dots, u_k\}$ and $B = \{u_{k+1}, \dots, u_d\}$, such that A and B form a valid vertical split, and k is as small as possible. If we cannot find a valid split of this form, we use a horizontal split instead. A will be assigned to the lower container and B to the upper container for a vertical split. Thus, we want to achieve a high mask quality (measured as a low score in one of our measures) for A in the lower container. Instead of recursively computing the optimal mask quality for A (which would be too expensive), we estimate mask quality using a simple layout for A : all remaining splits in A are horizontal. This way we can efficiently compute an estimate for any mask quality measure (e.g., EOPN) in A .

Having found a valid vertical split, we use a local search algorithm to further improve the estimated mask quality in A . We repeatedly pick the node $b \in B$ such that: (1) moving b from B to A still results

Algorithm 3 MASKAWARESPPLIT(U, R)

Input: U is a set of $d > 1$ nodes; R is a rectangle of size $\sum_{u \in U} \mu(u)$.
Output: A valid split with direction and sets A and B , such that $A \cup B = U$, $A \cap B = \emptyset$.

- 1: Sort $U = \{u_1, \dots, u_d\}$ by decreasing μ value
- 2: $k, k' \leftarrow$ smallest and largest value such that $\{u_1, \dots, u_k\}, \{u_{k+1}, \dots, u_d\}$ is a valid vertical split
- 3: **if** k and k' exist **then** {a valid vertical split exists*}
- 4: $A, B \leftarrow \{u_1, \dots, u_k\}, \{u_{k+1}, \dots, u_d\}$
- 5: $A', B' \leftarrow \{u_{k'+1}, \dots, u_d\}, \{u_1, \dots, u_{k'}\}$
- 6: Improve estimate e (e') for split A, B (A', B') by repeatedly moving the node from B (B') to A (A') that most improves the estimate and maintains validity, until no such element exists
- 7: **if** $e \leq e'$ **then**
- 8: **return** vertical, A, B
- 9: **else**
- 10: **return** vertical, A', B'
- 11: **else**
- 12: $k \leftarrow$ smallest value such that $\{u_1, \dots, u_k\}, \{u_{k+1}, \dots, u_d\}$ is a valid horizontal split
- 13: $A, B \leftarrow \{u_1, \dots, u_k\}, \{u_{k+1}, \dots, u_d\}$
- 14: Reduce imbalance $|\sum_{a \in A} \sigma(a) - \sum_{b \in B} \sigma(b)|$ for split A, B by repeatedly moving the node from B to A that most reduces the imbalance while maintaining validity, until no such element exists
- 15: **return** horizontal, A, B

in a valid vertical split, and (2) the estimated mask quality in A (after moving b to A) is improved the most. We repeat this step until no further improvement is possible.

Finally, we also try the same approach with the order of the children reversed. That is, we choose the largest k' such that $A' = \{u_{k'+1}, \dots, u_d\}$ and $B' = \{u_1, \dots, u_{k'}\}$ is a valid vertical split. That is, A' contains the smaller children and B' the larger children. Note that this is different from the mask-friendly strategy, placing smaller children at the bottom. However, since the mask-aware algorithm uses the uncertainty values and small children may have large uncertainty, this approach can sometimes improve mask quality.

Horizontal Splits. If we apply a horizontal split, then the split into A and B does not influence the mask-quality estimation. However, for both A and B , it is beneficial for mask quality to put elements with large uncertainty at the bottom in future vertical splits. Hence, we aim to make the total uncertainty in A and B as equal as possible.

As above, we start with a partition $A = \{u_1, \dots, u_k\}$ and $B = \{u_{k+1}, \dots, u_d\}$, such that A and B form a valid horizontal split, and k is as small as possible. Next, we aim to minimize $|\sum_{a \in A} \sigma(a) - \sum_{b \in B} \sigma(b)|$. We again try to minimize this difference via a local search: we repeatedly move the node $b \in B$ from B to A that reduces the difference as much as possible, while ensuring that the horizontal split remains valid. Since A and B are essentially symmetric for horizontal splits, there is no need to evaluate the reverse order of the children, like we did for the vertical splits.

5 EXPERIMENTAL RESULTS

We use computational experiments to relate our quality measures (Section 5.1) and investigate the effectiveness of mask-friendly and mask-aware algorithms (Section 5.2 and 5.3, respectively).

General Setup. We consider the following six original algorithms in our experiments: APP, the approximation algorithm [30]; SPL, the Split algorithm [14]; SQR, the Squarified algorithm [9]; STR, the Strip algorithm [4]. SQR and STR are also implemented with a look-ahead [4], denoted by SQRL and STRL, respectively. We use the indicated abbreviations to refer to the original algorithm, and append -F to indicate mask-friendly implementations.

For our parameter q (see Section 4), we indicate our mask-aware algorithm using APP- q N and APP- q S, depending on whether we base the estimate on EOPN or EOPS. We set q to 3 or 5, thus resulting in four mask-aware algorithms.

We run each of these 16 algorithms on the four real-world datasets below³, providing a rectangle of width 1920 and height 1080 as the input rectangle. We measure the excess overlap according to the schemes described in Section 3.4. We do not investigate the running time in detail, as each algorithm is effectively instantaneous on the datasets (less than 1 millisecond on a normal laptop).

Coffee: mean amount of coffee import per country from 1994 to 2014 with the associated standard deviation [39].

Infant: mean number of infant deaths per country from 1992 to 2016 with the associated standard deviation.⁴

S&P: mean closing price per stock in the Standard & Poor's 500 Index from 03-11-2016 to 10-11-2016 with the associated standard deviation.

CES: mean expenditure per (consumer) household in 2014⁵; standard deviation represents the measurement uncertainty [6].

5.1 Relating Mask-Quality Metrics

We introduced four variants of our mask quality measure, depending on whether we measure excess overlap to the parent or all ancestors,

³Available at <https://github.com/tue-aga/UncertaintyTreemaps>. doi:10.5281/zenodo.3624804

⁴<https://data.worldbank.org/indicator/SH.DTH.IMRT>, accessed 04-07-2018.

⁵<https://stats.bls.gov/cex>, accessed 13-02-2017.

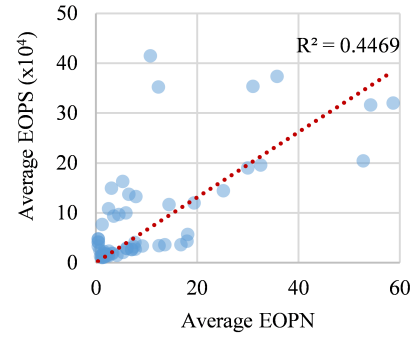


Figure 5: Average EOPN and EOPS do not show a clear correlation, thus capturing mask quality differently.

and whether or not we normalize to fractions or measure based on actual area. We investigate here how different these measures are in capturing aspects of quality. To this end, we measure the correlation coefficient (R^2 score for the linear regression) between each of the two measures, based on the quality of all dataset-algorithm pairs.

The choice between parent and all ancestors is highly correlated (0.95 for mean EOPN and EOAN, and 0.99 for mean EOPS and EOAS). This high correlation implies that, at least on our datasets, the choice is not distinctive. In other words, the overhead of considering all ancestors does not change the effective quality ranks or differences significantly. Thus, we consider only excess overlap measured with respect to the parent in the remainder of this section.

The normalized variant and size variant (average EOPN and EOPS) are not as strongly correlated. The correlation is below 0.45 and Figure 5 shows clear outliers. As such, we may consider these measures distinct. The size variant captures the amount of screen space (roughly matching visual saliency) of excess overlap but may underestimate small nodes with low quality; the normalized variant treats all nodes equally, independent of their size.

5.2 Effect of Mask-Friendly Algorithms

We implemented six algorithms, both in the original and in a mask-friendly manner (see Section 4.1). By design, mask-friendly implementations keep the same quality in terms of aspect ratio but may have a considerable impact on mask quality, essentially without cost. Here, we investigate this impact: we consider the average and maximum excess overlap of the nodes in the layout, averaged over all datasets, measured for both the original algorithms as well as the mask-friendly implementations. Figure 6 shows the results. Both in terms of average and maximum excess overlap, we see considerable improvement for most algorithms, but specifically for APP and SPL.

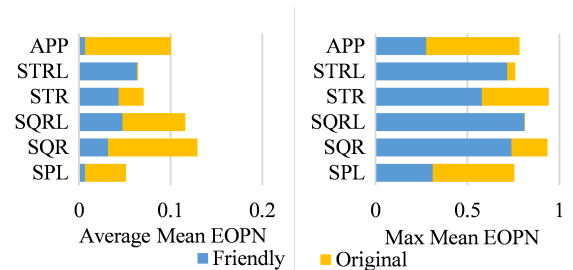


Figure 6: Improvement in mask quality for mask-friendly algorithms compared to original implementations, based on average (left) and maximum (right) excess overlap. Mean is taken over all datasets.

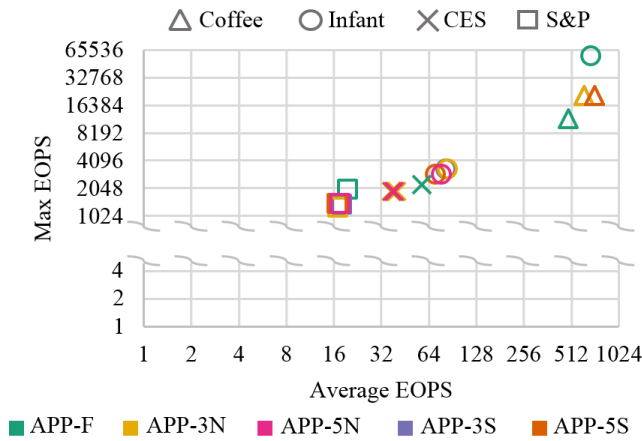


Figure 10: Scatterplot of mean and maximum excess overlap (EOPS) for each APP variant and dataset combination.

normalized, yet such normalization is difficult to achieve reasonably: the actual maximum achievable depends not only on the number of nodes, but also on hierarchy depth and structure, as well as the (relative and absolute) uncertainty values. Not surprisingly, the APP-*N variants generally outperform the APP-*S variants, as we measure normalized excess overlap and also use this to heuristically guide the layout algorithm. APP-3N performs best on all datasets except for S&P, on which APP-5S performs best, very closely followed by APP-5N. The differences are most clearly observed for the Infant and S&P datasets, but also marginally present in the others.

Let us now briefly consider the effect of our parameter q . Conceptually, increasing its value gives us more slack and thus room to improve mask quality. However, we observe that this parameter does not quite achieve that effect in practice: APP-3* versions quite consistently outperform APP-5* versions in mask quality (again, for the S&P dataset). This can likely be attributed to the extra slack admitting a different split high in the hierarchy, when the estimate is likely further off from the excess overlap that is eventually achieved.

Area-Based Quality. We now briefly consider EOPS, as it is not strongly correlated to EOPN. Figure 10 shows the same data as before, but using the mean and maximum EOPS measure. Our first observation is that for the Coffee dataset, APP-F performs best. This seems to be caused by an unfortunate split in the mask-aware algorithms at the root level. In contrast, for the other datasets, APP-F performs considerably worse, by a factor 17 for Infant and factor of around 1.17 for CES and 1.34 for S&P. Surprisingly, APP-*S is quite consistently outperformed by APP-*N, even though the latter uses an estimate not matching the measure we consider here. This is due to the more severe changes made at the higher levels by APP-*S, which turn out later to be detrimental to the overall quality.

Conclusion. We recommend APP-3N as the best algorithm to compute Uncertainty Treemaps: it generally provides the best or otherwise competitive performance in terms of aspect ratio and mask quality (both normalized and area-based) and demonstrates more consistent performance between datasets.

6 DISCUSSION

We now discuss our visualization approach and potential alternatives. In particular, we consider alternatives for mask shape, placement, and rendering, which easily combine with our layout algorithms. We also discuss effects of high uncertainty and unbalanced hierarchies.

6.1 Mask Shape and Placement

A mask is a generic representation of uncertainty using the area covered by it. This still leaves a big design space for precise shape and

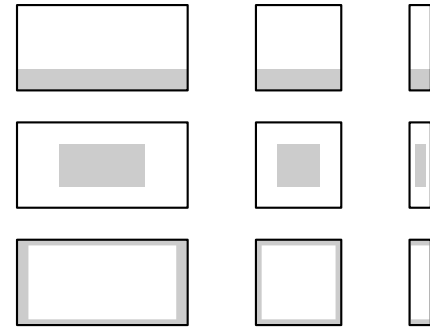


Figure 11: Options for mask placement, from top to bottom: bottom rectangle, inner rectangle, and passe-partout. Each row shows nodes with different aspect ratios and 25% relative uncertainty.

placement. We recommend placing rectangular masks at the bottom and along the full width of a node. All examples so far used this approach, as it simplifies relative uncertainty assessment. However, other variants are possible, for example, placing the mask at the center or as four rectangles along the boundary. We denote the latter type as *passe-partout* because it resembles this matting construction from picture framing. Figure 11 illustrates our recommended bottom rectangle placement and the two alternatives.

The three placement variants have different characteristics. The inner-rectangle placement has the advantage that it is symmetric within the node. However, it has several shortcomings. First, because the inner part is not visually connected to the node, we expect that it is harder to relate the mask to its containing rectangle if many nodes overlap. Second, the visual encoding of uncertainty now fully relies on the accuracy of area perception. Unfortunately, the human visual system is not ideal for area estimation. Steven's original power law already indicated a nonlinear mapping between physical area in the stimulus and perceived sensation [36, 37]. Later work discusses other aspects of area perception [26, 38]. In essence, area encoding is not the best option for a reliable visual representation.

Length perception is more accurate, following Cleveland and McGill [11] or Mackinlay [28]. With bottom-rectangle placement, a reader can use length perception to estimate relative uncertainty. Comparisons between nodes with identical height (horizontally adjacent) can use height directly to accurately compare relative uncertainty; similarly, height can be used directly to compare absolute uncertainty between nodes with identical width (vertically adjacent). Inner-rectangle and passe-partout placements cannot directly rely on length perception for estimating relative uncertainty. Deciding which node has higher relative uncertainty is possible using length percep-

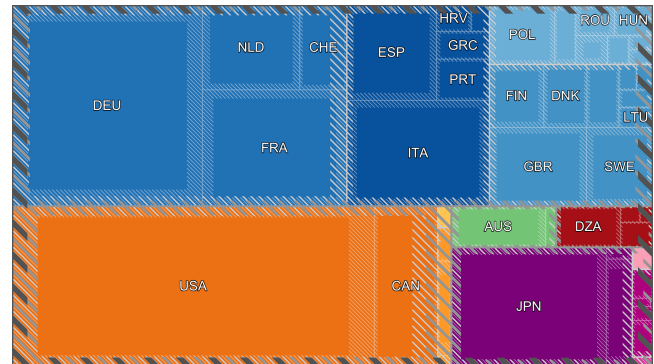


Figure 12: Coffee dataset using APP layout with passe-partout mask.

tion if the width or height of the nodes is the same. But the lengths to be compared are more separated (especially for inner-rectangle) compared to bottom-rectangle placement.

Finally, aspect ratios of the masked areas tend to already be low in bottom-rectangle placement. This is exacerbated in the passe-partout placement. As extreme aspect ratios hinder comparing area [25], this is undesirable.

Therefore, for robust uncertainty visualization that supports relative as well as absolute uncertainty assessment, we recommend bottom-rectangle placement. However, the passe-partout placement might be preferable if a more symmetric visualization is required, for example, to compare along the horizontal and vertical axes of the treemap, or to implicitly emphasize the nesting hierarchy by the uncertainty borders. Figure 12 shows a passe-partout rendering of the Coffee dataset, to be compared with Figure 1.

Completely different shapes might also be used, such as piecewise linear or even curved mask boundaries. The mask might also be split into disjoint regions. With this flexibility, other optimization cost functions could be explored. However, such shapes differ significantly from the general look of rectangular treemaps and we thus expect that this does not result in a harmonious whole.

6.2 Mask Rendering

Besides mask layout, its rendering is critical for an appropriate visualization and should be matched to hierarchy level for visibility. We chose slanted hatching; below, we discuss two other options.

Bar Hatching. We could use vertical bars in an otherwise identical design. However, such vertical bars are less visually separable from the treemap partitioning itself due to the use of the same angles. Also, we expect bars to give the impression that the uncertainty value is encoded as height instead of area.

Checkerboard. This design uses a checkerboard pattern of opaque and (fully) transparent squares. The size of the squares doubles with every level of the hierarchy, ensuring that the next-level mask can fit a 2×2 pattern with a transparent square. Thus, lower-level masks always remain partially visible. The drawback of this design is that it occludes a significant portion of the lower-level masks, as is also aligned with the treemap partitioning. As a result, this pattern remains hard to assess. Moreover, the visual complexity of the pattern is high as well, possibly distracting from the actual data.

All three variants follow general design recommendations for uncertainty visualization: they implicitly modify the perceived luminance and saturation in the mask areas; these two visual channels are suitable to show uncertainty, as, for example, discussed by Guo *et al.* [18] or MacEachren *et al.* [27]. Our techniques also resemble screen-door transparency for uncertainty visualization [32]. Sketchiness would have been a different kind of visual mapping, but as Boukhelifa *et al.* [7] point out, might lead to an unprofessional look.

Figure 13 compares our choice and the two alternatives. It suggests that (diagonal) hatching yields a good mask rendering, especially for several layers of overlay. Future work may include a formal evaluation of these and other rendering options in terms of effectiveness for conveying uncertainty in treemaps.

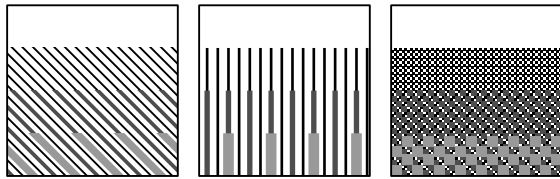


Figure 13: Mask rendering: hatched, bar, and checkerboard (left to right). All show the same three-level hierarchy in a single node.

6.3 Uncertainty and Hierarchy

Uncertainty Models. We generally assume that uncertainty is defined at the leaves as a standard deviation and that the underlying distributions are independent. We use this in establishing our datasets, but our techniques (masks, mask quality, algorithms) are otherwise agnostic to the source and relations between uncertainty. Of course, using different models of uncertainty may influence the performance of the various improvements.

For example, each node may be characterized by an interval of potential values. We could use the upper bound as μ and the interval size as σ . The aggregation of values through the hierarchy has to be done using interval arithmetic. Though the aggregation method changes, no changes to the techniques are necessary.

High Uncertainty. Our method is designed for scenarios where $\sigma(v) \leq \mu(v)$ for any node v . As treemaps are used for nonnegative data, this seems a reasonable condition but may not always be met for all nodes. In rare cases, we observe that the uncertainty is larger than the data value. In such cases, the mask covers the full node, though that does not quite accurately capture the actual uncertainty. It does strongly signal nodes with (very) high uncertainty that warrant further investigation, and as such may be sufficient.

To show such high uncertainty more precisely, we may re-mask the node. That is, we show the hatched mask over the entire node, and then appropriate apply a second mask with area $\sigma(v) - \mu(v)$ using a hatched pattern using diagonals in the other direction. This effectively creates a cross-hatching, where the cross-hatched area of the mask counts twice toward visualizing the uncertainty. This allows visualizing uncertainty up to twice the data value of a node.

Generally, we could allow k orientations of hatching to communicate uncertainty up to k times the data value. However, such patterns would likely obscure too much of the node itself, making it more difficult to assess its data value and compare it to other nodes. Hence, we do not apply such generalized cross-hatching.

Unbalanced Hierarchies. Thus far we have shown only hierarchies in which the leaves have the same depth. All our real-world datasets indeed have this property. However, more generally, some leaves may have more ancestors than others. We can readily apply our algorithms and masks, but it raises one question: which pattern do we choose for the uncertainty mask? Specifically, do we use the *depth* or *height* of a node in the hierarchy? See also Figure 14.

Using depth, going down one level in the hierarchy results in the same mask level. Thus, children of the same parent (or nodes on the same level) have the same mask detail. However, leaves do not necessarily use the most fine-grained mask, possibly de-emphasizing them as leaves. Using height, each leaf has the most fine-grained mask. However, a node with children of different height then uses a mask detail that may be coarser by multiple steps compared to some of its children. The benefits and drawbacks of these variants warrant further consideration in future research. We expect that height-based masks are more efficacious, as they mark leaves more clearly: starting at a coarser mask may incorrectly suggest that the mask is part of a higher-level node and that the leaf has no uncertainty.

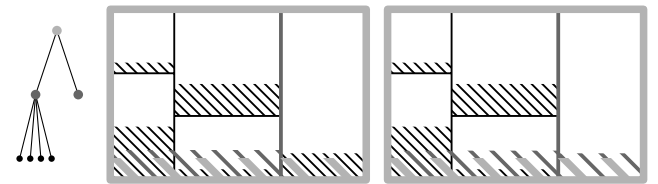


Figure 14: Two different masking strategies for an unbalanced hierarchy (left): height-based (middle) and depth-based (right). Note the different rendering of the mask in the rightmost leaf.

7 CONCLUSION AND FUTURE WORK

We introduced a technique for visualizing uncertainty in treemaps that maintains the strict recursive nature of treemaps by showing uncertainty using area. To do so, we designed hierarchical uncertainty masks and applied them to rectangular treemaps. A brief expert review suggests that this technique is practically usable. We modified algorithms that compute rectangular treemaps, and experimentally showed that these algorithms achieve high quality in terms of both aspect ratio and mask quality.

We also discussed various alternatives that uncover potential challenges and directions for future work. This includes different mask designs, case studies, and user studies. Another venue for future research is a level-of-detail rendering to make Uncertainty Treemaps scalable for very large datasets—in terms of number of data points (nodes) and the depth of the hierarchy—for example through data-driven pruning of the hierarchy and adaptive mask rendering. Our technique, with its general concepts and metrics, provides a framework upon which such future research can be built.

ACKNOWLEDGMENTS

The authors would like to thank Michel Westenberg for participating in our expert review. The German Research Foundation (DFG) is supporting CS and DW under project no. 251654672 – TRR 161 (Project A01). The Netherlands Organisation for Scientific Research (NWO) is supporting MS and BS (partially) under project no. 639.023.208 and KV (partially) under project no. 639.021.541.

REFERENCES

- [1] D. Auber, C. Huet, A. Lambert, B. Renoust, A. Sallaberry, and A. Saulnier. GosperMap: Using a gosper curve for laying out hierarchical data. *IEEE TVCG*, 19(11):1820–1832, 2013.
- [2] A. S. Bair, D. H. House, and C. Ware. Factors influencing the choice of projection textures for displaying layered surfaces. *Proc. 6th Symp. Appl. Perception Graphics Vis.*, page 101, 2009.
- [3] M. Balzer, O. Deussen, and C. Lewerentz. Voronoi treemaps for the visualization of software metrics. In *Proc. ACM Symp. Softw. Vis.*, pages 165–172, 2005.
- [4] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Trans. Graphics*, 21(4):833–854, 2002.
- [5] J. Bertin. *Semiology of graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983.
- [6] J. L. Blaha. Standard errors in the Consumer Expenditure Survey. <https://www.bls.gov/cex/anthology/csxnth5.pdf>, 2003.
- [7] N. Boukhelifa, A. Bezerianos, T. Isenberg, and J.-D. Fekete. Evaluating sketchiness as a visual variable for the depiction of qualitative uncertainty. *IEEE TVCG*, 18(12):2769–2778, 2012.
- [8] K. Brodlie, R. AllendesOsorio, and A. Lopes. A review of uncertainty in data visualization. In J. Dill, R. Earnshaw, D. Kasik, J. Vince, and P. C. Wong, editors, *Expanding the Frontiers of Visual Analytics and Visualization*, pages 81–109. Springer, 2012.
- [9] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In *Proc. VisSym*, pages 33–42, 2000.
- [10] M. Chen, S. Walton, K. Berger, J. Thyagalingam, B. Duffy, H. Fang, C. Holloway, and A. E. Trefethen. Visual multiplexing. *CGF*, 33(3):241–250, 2014.
- [11] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *J. Amer. Stat. Assoc.*, 79(387):531–554, 1984.
- [12] M. Correll and M. Gleicher. Error bars considered harmful: Exploring alternate encodings for mean and error. *IEEE TVCG*, 20(12):2142–2151, 2014.
- [13] M. de Berg, B. Speckmann, and V. van der Weele. Treemaps with bounded aspect ratio. *CGTA*, 47(6):683–693, 2014.
- [14] B. Engdahl. Ordered and unordered treemap algorithms and their applications on handheld devices, 2005. MSc thesis, Dept. Numer. Anal. Comp. Sci., Stockholm Royal Inst. Technology, Sweden.
- [15] J. Görtler, C. Schulz, D. Weiskopf, and O. Deussen. Bubble treemaps for uncertainty visualization. *IEEE TVCG*, 24(1):719–728, 2018.
- [16] M. Greis, P. E. Agroudy, H. Schuff, T. Machulla, and A. Schmidt. Decision-making under uncertainty: How the amount of presented uncertainty influences user behavior. *Proc. 9th Nordic Conf. Human-Comput. Interaction*, pages 2–5, 2016.
- [17] T. Gschwandtner, M. Bogl, P. Federico, and S. Miksch. Visual encodings of temporal uncertainty: A comparative user study. *IEEE TVCG*, 22(1):539–548, 2016.
- [18] H. Guo, J. Huang, and D. H. Laidlaw. Representing uncertainty in graph edges: an evaluation of paired visual variables. *IEEE TVCG*, 21(10):1173–1186, 2015.
- [19] N. S. Holliman, A. Coltekin, S. J. Fernstad, M. D. Simpson, K. J. Wilson, and A. J. Woods. Visual entropy and the visualization of uncertainty. *arXiv preprint arXiv:1907.12879*, 2019.
- [20] J. Hullman. Why authors don't visualize uncertainty. *IEEE TVCG*, 26(1):130–139, 2019.
- [21] J. Hullman, X. Qiao, M. Correll, A. Kale, and M. Kay. In pursuit of error: A survey of uncertainty visualization evaluation. *IEEE TVCG*, 25(1):903–913, 2019.
- [22] A. Kale, F. Nguyen, M. Kay, and J. Hullman. Hypothetical outcome plots help untrained observers judge trends in ambiguous data. *IEEE TVCG*, 25(1):892–902, 2019.
- [23] R. Khlebnikov, B. Kainz, M. Steinberger, and D. Schmalstieg. Noise-based volume rendering for the visualization of multivariate volumetric data. *IEEE TVCG*, 19(12):2926–2935, 2013.
- [24] C. Kinkeldey, J. Mason, A. Klippel, and J. Schiewe. Evaluation of noise annotation lines: using noise to represent thematic uncertainty in maps. *CaGIS*, 41(5):430–439, 2014.
- [25] N. Kong, J. Heer, and M. Agrawala. Perceptual guidelines for creating rectangular treemaps. *IEEE TVCG*, 16(6):990–998, 2010.
- [26] A. Longjas, E. F. Legara, and C. Monterola. Power law mapping in human area perception. *Int. J. Mod. Phys. C*, 22(5):495–503, 2011.
- [27] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, and M. Gahegan. Visual semiotics & uncertainty visualization: An empirical study. *IEEE TVCG*, 18(12):2496–2505, 2012.
- [28] J. D. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graphics*, 5(2):110–141, 1986.
- [29] M. J. McGuffin and J.-M. Robert. Quantifying the space-efficiency of 2D graphical representations of trees. *Inf. Vis.*, 9(2):115–140, 2010.
- [30] H. Nagamochi and Y. Abe. An approximation algorithm for dissecting a rectangle into rectangles with specified areas. *Discrete Appl. Math.*, 155(4):523–537, 2007.
- [31] D. Sacha, H. Senaratne, B. C. Kwon, G. Ellis, and D. A. Keim. The role of uncertainty, awareness, and trust in visual analytics. *IEEE TVCG*, 22(1):240–249, 2016.
- [32] C. Schulz, K. Schatz, M. Krone, M. Braun, T. Ertl, and D. Weiskopf. Uncertainty visualization for secondary structures of proteins. In *2018 IEEE Pacific Vis. Symp.*, pages 96–105. IEEE, 2018.
- [33] H.-J. Schulz, S. Hadlak, and H. Schumann. The design space of implicit hierarchy visualization: A survey. *IEEE TVCG*, 17(4):393–411, 2011.
- [34] A. Slingsby, J. Dykes, and J. Wood. Exploring uncertainty in geodemographics with interactive graphics. *IEEE TVCG*, 17(12):2545–2554, 2011.
- [35] M. Sondag, B. Speckmann, and K. Verbeek. Stable treemaps via local moves. *IEEE TVCG*, 24(1):729–738, 2018.
- [36] S. S. Stevens. On the psychophysical law. *Psych. Rev.*, 64(3):153–181, 1957.
- [37] S. S. Stevens. The psychophysics of sensory function. *Amer. Scientist*, 48(2):226–253, 1960.
- [38] M. Teghtsoonian. The judgment of size. *Amer. J. Psych.*, 78(3):392–402, 1965.
- [39] United Nations. UN comtrade database. <https://comtrade.un.org>, 2016. Accessed on 15-02-2017.
- [40] W. Wang, H. Wang, G. Dai, and H. Wang. Visualization of large hierarchical data by circle packing. In *Proc. SIGCHI Conf. Human Factors Computing Syst.*, pages 517–520, 2006.
- [41] M. Wattenberg. A note on space-filling visualizations and space-filling curves. In *Proc. IEEE Symp. Inf. Vis.*, 2005.