MLVis: Machine Learning Methods in Visualisation for Big Data (2019)
D. Archambault, I. Nabney and J. Peltonen (Editors)

On KDE-based brushing in scatterplots and how it compares to CNN-based brushing

Chaoran Fan and Helwig Hauser

University of Bergen, Norway, `ii.UiB.no/vis`

Abstract

In this paper, we investigate to which degree the human should be involved into the model design and how good the empirical model can be with more careful design. To find out, we extended our previously published Mahalanobis brush (the best current empirical model in terms of accuracy for brushing points in a scatterplot) by further incorporating the data distribution information that is captured by the kernel density estimation (KDE). Based on this work, we then include a short discussion between the empirical model, designed in detail by an expert and the deep learning-based model that is learned from user data directly.

CCS Concepts

• ***Human-centered computing*** → *Interaction techniques;* • ***Computing methodologies*** → *Optimization algorithms;*

1. Introduction

Linking and brushing is a prevalent interaction technique for data exploration and analysis in coordinated multiple views. Becker and Cleveland [BC87] were the first to come up with a theory of brushing, defined as an interactive method to select data points by using simple geometries (square, circle, or a polygon).

Since brushing is central in modern visual analytics systems, it has attracted a considerable amount of research and most of the proposed techniques can be evaluated by the following two criteria.

- *efficiency*—how fast is the brushing interaction; does it enable a fluid data exploration [EVMJ*11,TKBH17]? Clicking one point, for example, is a minimal interaction that is also most efficient.
- *accuracy*—to which degree does the brushing interaction lead to an accurate selection of the data subset, which the user actually wished to select? The lasso, for example, is a brushing tool which can be used to specify the brush region with 100% accuracy.

In order to optimize both criteria in one technique as well as possible, data-driven interaction techniques became popular recently. This type of method is usually based on a fast, sketch-based user interaction (for example, click-and-drag) and a heuristic, which transforms the sketching interaction into the actual data selection, based on the underlying data visualization. To improve the accuracy, the parameters of the technique can be automatically optimized by using data from a user study.

Our previously published work—Mahalanobis brush [FH17]—is a typical example of a data-driven technique, taking the local covariance information into account, forming the basis for a local Mahalanobis metric. The parameters are then optimized based on data from a user study. The quantitative evaluation shows that it can achieve ≈92% accuracy based on a fast interaction (click-and-drag). Later, inspired by the success of deep learning methods in a wide range of fields, we developed a CNN-based brushing technique [FH18] and achieved the so far best accuracy (≈97%). For this method, we converted the interaction and data information into an image-based input to the network and allowed the network to learn a model itself.

As machine learning [FH18] had outperformed empirical modeling [FH17], we were curious about how much the empirical model can be improved with a more sophisticated design and whether it can beat the deep learning approach. In this paper, we report our attempt to construct a best-possible empirical model by further extending the Mahalanobis brush, incorporating kernel density estimation (KDE) [Par62], and informing a clustering step that returns one of the clusters as the data selection. The main contribution of this paper includes our extension of the empirical model for brushing points in a scatterplot, and a comparison of the two approaches, as well as an according discussion.

2. Related work

Brushing in scatterplots is often based on simple geometric shapes such as a rectangle or an ellipse to select data items, or a lasso is used to specify the brush accurately. Several extensions to simple brushing have been proposed over the years,

Martin and Ward [MW95] suggest logical combinations of brushes, including unions, intersections, negations, and XOR operations, enabling the user to configure composite brushes.

MyBrush was suggested by Koytek et al. [KPV*17] in order to extend brushing and linking by incorporating personal agency. It

DOI: 10.2312/mlvis.20191157

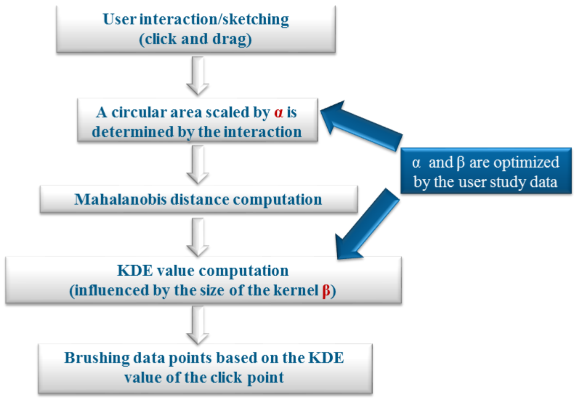
User interaction/sketching (click and drag)
A circular area scaled by α is determined by the interaction
α and β are optimized by the user study data
Mahalanobis distance computation
KDE value computation (influenced by the size of the kernel β)
Brushing data points based on the KDE value of the click point

Figure 1: *Overview of our KDE brushing technique: the user clicks into the middle of the data subset to be selected and drags the pointer to the border of the subset (sketching interaction); then a selection of points around the click-point is determined, based on the KDE values of the data. Two parameters, α and β, related to the sample size, and the size of the KDE bandwidth, influence the results and we optimize them using a user study (50 participants).*

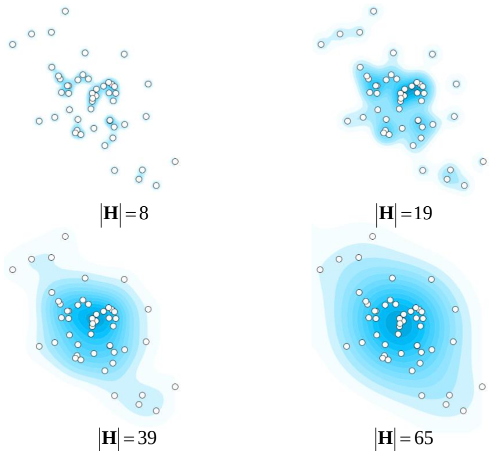

Figure 2: *Changing the size of the kernel in KDE: bigger kernels (to the lower right) bring forth larger structures in the data, while smaller kernels (to the upper left) represent details.*

offers users the flexibility to configure the source, link, and target of multiple brushes.

Similarity brushing [NH06b, MKO*08] is a classical example of sketch-based brushing, which is based on a fast and simple sketching interaction—the user uses a swift and approximate gesture (for ex., drawing an approximate shape that the data should follow) and then a similarity measure is designed to identify, which data items actually are brushed.

Our previously published Mahalanobis brush [FH17] is an extension of the original Mahalanobis brush, introduced by Rados et al. [RSM*16], in which the user clicks into the center of a coherent data subset to select it. The local covariance information is used to form a selection metric. The main drawbacks of this method are a non-optimized selection of the local context for the Mahalanobis computation and one off-screen parameter for the brush size. Both issues were successfully addressed in our method [FH17].

Later, we exploited machine learning and developed CNN-based brushing in scatterplots [FH18]. This technique uses the same interaction as the Mahalanobis brush [FH17], while achieving the so far best accuracy.

3. KDE brushing in scatterplots

Figure 1 shows an overview of the new, KDE-based brushing technique. We keep the simple click-and-drag interaction for sketching the data subset (click into the middle of the targeted data subset and drag the pointer to the outer boundary of the subset). The click-point $\mathbf{s} = (s_x, s_y)^\top$ and the end-point $\mathbf{e} = (e_x, e_y)^\top$ of the drag-interaction provide us with a first hint concerning the size of the data subset, which the user wishes to brush. Similarly to the Mahalanobis brush [FH17], we first consider a circular data subset, centered around the start-point of the interaction, and estimate the shape and orientation of the data in this region by looking at the local covariance information. We then start a short iteration that refines this data subset selection, based on the local covariance information. After a sufficiently close convergence of this iteration, we make a selection of data points, based on a kernel density estimation of the data, using the local covariance information as a basis for specifying the kernel. In the following, we go into more details with respect to the individual components of our solution.

The Mahalanobis distance is a distance measure introduced by P. C. Mahalanobis [Mah36], which is used for helping with the identification and analysis of patterns in the data. The Mahalanobis distance between vectors **a** and **b** can be defined by

$$d_\Sigma(\mathbf{a},\mathbf{b}) = \sqrt{(\mathbf{a}-\mathbf{b})^\top \Sigma^{-1} (\mathbf{a}-\mathbf{b})} \qquad (1)$$

Initially, we consider a circular area with the radius $\alpha \cdot d_E(\mathbf{s},\mathbf{e})$, where α is a weighting factor and $d_E(\mathbf{s},\mathbf{e})$ is the Euclidean distance between **s** and **e**. All data points within this circle are used to compute the first instance of the local covariance information, Σ_1. Next, we consider all points within a Mahalanobis ellipse, based on Σ_1 and sized according to $d_\Sigma(\mathbf{s},\mathbf{e})$. Usually, this leads to a new data subset, which is similar to the data subset as determined by the initial circle, but more closely following the underlying data structure. To obtain an even better sample, we refine the sample iteratively by replacing them with the points in the Mahalanobis ellipse which is updated every iteration according to the covariance of the samples in last iteration. However, we observe that this process sometimes can lead to small fluctuations, including/excluding a few data points in consecutive iterations. Therefore, we stabilize the convergence by enabling the partial consideration of data points, leading to a solution that is based on the weighted covariance matrix [Gou09]. The details of this procedure is introduced in [FH17].

4. Kernel Density Estimation

Kernel density estimation (KDE) is a popular method for data analysis in the field of statistics, which was introduced by Rosenblatt [R*56] and Parzen [Par62]. It is a non-parametric way to estimate the probability density function of a random variable. KDE can be used, for example, to make inferences about data, based on a finite sample.

Assuming that $\{\mathbf{x}_i\}_{1\leq i\leq n}$ is a sample of n d-dimensional vectors drawn from a common distribution, described by a particular density function, then KDE can be used to estimate this density function as

$$f_{\mathbf{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{x}-\mathbf{x}_i) \tag{2}$$

with $\mathbf{H}$ being the $d\times d$ kernel matrix (symmetric and positive definite). The choice of matrix $\mathbf{H}$ is the single most important factor affecting the main characteristics of $f_{\mathbf{H}}$ [WJ95]. Figure 2 shows four results from a 2D KDE with increasing size of $\mathbf{H}$.

$K_{\mathbf{H}}$ is a kernel function with $K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-\frac{1}{2}}K(\mathbf{H}^{-\frac{1}{2}}\mathbf{x})$, where $K(\mathbf{x})$ is a symmetric multivariate density function with $K(\mathbf{x})\geq 0$ and $\int K(\mathbf{x})\,d\mathbf{x} = 1$. A variety of kernels has been studied, including the uniform kernel, the triangle kernel, the normal kernel (based on a Gaussian distribution), the Epanechnikov kernel [T*93], and others. The choice of the kernel function is actually not as important as the choice of the size (and shape) of $\mathbf{H}$. Being interested in the local mode of the data distribution, we use the normal kernel [MS93].

Since we wish to consider the local data distribution when modeling an appropriate kernel matrix $\mathbf{H}$, we can make direct use of the converged covariance matrix Σ, leading to the following (anisotropic) kernel function [Ton12]

$$K_{\mathbf{H}}(\mathbf{x}) = \frac{e^{-\frac{1}{2}\mathbf{x}^{\top}\Sigma\,\mathbf{x}}}{\sqrt{(2\pi)^d|\Sigma|}} \tag{3}$$

In order to realize an appropriate scaling of our kernel, we make use of an eigendecomposition of $\Sigma = \mathbf{V}\Lambda\mathbf{V}^{\top}$ with eigenvectors $\mathbf{V}$ and eigenvalues Λ. This leads to the following, scaled versions of $|\mathbf{H}|^{-\frac{1}{2}}$ and $\mathbf{H}^{-\frac{1}{2}}$:

$$|\mathbf{H}|^{-\frac{1}{2}} = |\beta\phi\Lambda|^{-\frac{1}{2}} \quad \& \quad \mathbf{H}^{-\frac{1}{2}} = (\beta\phi\Lambda)^{-\frac{1}{2}}\mathbf{V}^{\top} \tag{4}$$

Used with an isotropic kernel function $K(\mathbf{x}) = (2\pi)^{-\frac{d}{2}}e^{-\frac{1}{2}\mathbf{x}^{\top}\mathbf{x}}$, this amounts to a KDE with an accordingly scaled kernel matrix. We find the best possible scaling of $\mathbf{H}$ by choosing the two scaling parameters ϕ and β based on two separate solutions: On the one hand, we use a data-driven approach to determine ϕ (see section 4.2). On the other hand, we optimize β as a general parameter using the data from the user study (see section 6).

4.1. Selecting a data subset using clustering

The modes of the KDE represent groups of data items (at the scale determined by the size of $\mathbf{H}$). We use clustering (each mode leading to one cluster) to identify the one group of data items, which is associated with the click-and-drag interaction, and select it.

For clustering, we use a watershed algorithm [BM92, NH06a, FH11, FH10]: Starting with the mode with the highest KDE value, we iteratively include neighboring locations into the corresponding cluster, lowering the KDE threshold for doing so iteratively. For every new threshold, we either join a neighboring location to an existing cluster, or create a new one, if the corresponding location is not adjacent to an existing cluster. Figure 3B shows an according clustering result for a KDE with a particular kernel (Figure 3A). Figure 3C shows how data points are then selected.

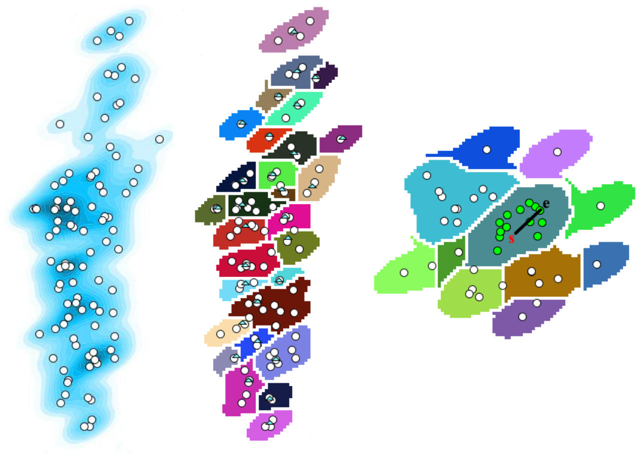

Figure 3: *A: KDE of a dataset (relatively small kernel). B: Clustering related to the modes of the KDE, indicated by the small blue triangles. C: The one cluster, which corresponds to the KDE mode near to* **s** *determines, which data points are selected (indicated as green points).*

4.2. Automatic adaption of the KDE kernel size

In general, the number of modes in a KDE is directly related to the kernel size: the bigger the kernel, the fewer modes [MS93]. Thus, there is a strong relation between the size of the kernel and the size of the cluster, which is used to select the data points. In the following, we describe a data-driven approach to determining an appropriate scaling factor ϕ.

Since we aim at a KDE that will provide one cluster in order to select the targeted data points, we optimize the size of the bandwidth kernel so that the size of the resulting cluster matches the size of the Mahalanobis ellipse around **s** and through **e**—as a measure of comparison, we are using the dice coefficient between the Mahalanobis ellipse $\mathbf{E}$ and the KDE cluster $\mathbf{C}(\phi)$ [Dic45]:

$$s(\phi) = \frac{2\,|\mathbf{E}\cap\mathbf{C}(\phi)|}{|\mathbf{E}|+|\mathbf{C}(\phi)|} \tag{5}$$

where $|\mathbf{E}|$ and $|\mathbf{C}(\phi)|$ are the sizes of the Mahalanobis ellipse and the KDE cluster, respectively (evaluated as a grid-based measure).

The example shown in Figure 4 demonstrates the influence of different kernel sizes on the brushing results. The true positives (correctly selected), true negatives (correctly non-selected), false positives (falsely selected) and the false negatives (falsely left out) are colored in yellow, white, pink and purple, accordingly. We see that there are more false negatives when the kernel size is too small (showed in the left) with a low similarity between the gray KDE cluster and the Mahalanobis ellipse. Conversely, more false positives appear when the kernel size is too big (on the right).

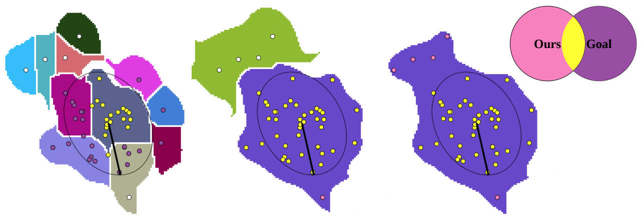

Figure 4: *Clustering based on different kernel sizes. Left: too small kernel,* $s(\phi) = 0.63$*; Middle: the optimal size of the kernel,* $s(\phi) = 0.72$*; Right: too big kernel,* $s(\phi) = 0.64$.

5. User study

Our technique, as described so far, has two not-yet-optimized parameters: α (initial selection, determining the context of local data shape analysis; too small values of α lead to underselection while too large values to overselection) and β (overall scaling parameter on top of ϕ, influencing the kernel size). In order to achieve as accurate as possible brushing, we have to get information about how users would use our technique to brush and what they actually wanted to select from the dataset (ground truth). As the interaction and brushing target designed in our method are same as the previously published Mahalanobis brush [FH17], we directly use the data from the published user study. Based on this information, we then did an optimization of α and β.

In the previously published user study, six representative datasets are provided. The datasets selected for the user study are based on scagnostics [TT88], aiming at a healthy spread of scatterplots of different type. For the user study, 50 participants were invited and asked to do 12 selections each. In every case, a particular scatterplot (one out of six) and a particular request (choose a large cluster/a small cluster/an elongated cluster) were given. Then the participant was instructed to choose a target data subset to select (ground truth, reported by the participants using a lasso tool), then also providing the corresponding click-and-drag interaction, which this participant would use to select the target group. Accordingly, 600 brushing cases were collected from the user study.

6. Optimization and Evaluation

From the 600 selections that we got from the published user study, we randomly chose 400 selections as training data. As a measure for how well the computed selection ($\mathbf{S}(\alpha,\beta)$) agrees with the user goal ($\mathbf{G}$), we compute the dice coefficient of these two sets:

$$s(\alpha,\beta) = \frac{2|\mathbf{G}\cap\mathbf{S}(\alpha,\beta)|}{|\mathbf{G}|+|\mathbf{S}(\alpha,\beta)|} \tag{6}$$

If the computed selection $\mathbf{S}(\alpha,\beta)$ matches the user goal $\mathbf{G}$ perfectly, $s(\alpha,\beta) = 1$; in the case of a complete mismatch, $s(\alpha,\beta) = 0$.

Having collected the ground truth (lasso data) from the study and the click- and release-points from the sketching interaction, we were able to conduct an optimization of α and β according to the following procedure (not involving users anymore): Based on varying choices of α and β, we could execute our selection heuristic, using the datasets from the user study and the recorded interaction data, leading to a particular $\mathbf{S}(\alpha,\beta)$—this was then straight-forward to compare to $\mathbf{G}$ as collected during the user study, leading to a corresponding $s(\alpha,\beta)$. We started with a large matrix of different combinations of the two parameters, covering a domain, which for sure was big enough. Inspecting the s-values for all these combination lead us to further examining a more detailed subset of the parameter space (basically, we refined our optimization hierarchically, doing the refinement manually). Eventually, we ended up with the following optimal values for both parameters: $\alpha = 1.05$ and $\beta = 1.05$. The whole optimization process is done offline once and it takes 4 hours for the computation.

Using the optimized parameters, we did an accuracy comparison with the previously published Mahalanobis brush [FH17] using the interaction information from the published study. The overall accuracy for the new technique is ≈90% which is slightly lower than the one of the Mahalanobis brush (≈92%) and does neither outperform the CNN-based brush [FH18] with its accuracy of ≈97%. So far, we have been unable to evaluate, why this the case – we had assumed that more carefully considering the local data distribution should help to further improve the technique's accuracy (as a nonlinear method, the KDE should have better power to adapt to nonlinear structures in the data). Unfortunately, we cannot rule out that we have overlooked another limitation when realizing the KDE-based approach – either a conceptual one, or a limitation of our implementation. Accordingly, we imagine that another solution can in fact achieve a further improved accuracy.

As another point of this discussion, we note that empirical modeling comes with the advantage of being explainable (for example, we know how different values of α and β influence the results in our approach), while the excellent performance of the deep learning model comes at the price of a poor interpretability (including some uncertainty concerning the stability of its predictive power). This comparison leads to the interesting question of how much accuracy we are willing to sacrifice for a good interpretability.

7. Conclusion and future work

In this paper, we present our attempt to improve the Mahalanobis brush by incorporating KDE to further improve its accuracy and we investigate the influence of human expertise during model design. Although more information is taken into account for modeling the KDE-based model, we have not seen an improvement compared to the simpler Mahalanobis brush. Based on this result, we realize that the incremental cost of incorporating KDE could be an over-design problem. In addition, when compared with deep learning, we found that its black-box mechanism leads to a poor interpretability, while the results based on the empirical model are explainable. It is unclear, however, how to weigh this factor in, when comparing the overall performance.

In the future, we see potential in taking innovative advantage of both sides—empirical modeling *and* machine learning. We imagine, for example, to automatically learn the kernel size or to design the deep learning input on the basis of the KDE. In addition, we are also interested to investigate the reasons why the KDE-based method performs worst than the Mahalanobis brush.

References

[BC87] Becker R. A., Cleveland W. S.: Brushing scatterplots. *Technometrics 29*, 2 (1987), 127–142. 1

[BM92] Beucher S., Meyer F.: The morphological approach to segmentation: the watershed transformation. *Optical Engineering 34* (1992), 433–433. 3

[Dic45] Dice L. R.: Measures of the amount of ecologic association between species. *Ecology 26*, 3 (1945), 297–302. 3

[EVMJ*11] Elmqvist N., Vande Moere A., Jetter H.-C., Cernea D., Reiterer H., Jankun-Kelly T. J.: Fluid interaction for information visualization. *Information Visualization 10*, 4 (Oct. 2011), 327–340. URL: http://dx.doi.org/10.1177/1473871611413180. 1

[FH10] Florek M., Hauser H.: Quantitative data visualization with interactive kde surfaces. In *Proceedings of the 26th Spring Conference on Computer Graphics* (2010), ACM, pp. 33–42. 3

[FH11] Florek M., Hauser H.: Interactive bivariate mode trees for visual structure analysis. In *Proceedings of the 27th Spring Conference on Computer Graphics* (2011), ACM, pp. 95–102. 3

[FH17] Fan C., Hauser H.: User-study based optimization of fast and accurate Mahalanobis brushing in scatterplots. In *Vision, Modeling and Visualization* (2017), The Eurographics Association. doi:10.2312/vmv.20171262. 1, 2, 4

[FH18] Fan C., Hauser H.: Fast and accurate cnn-based brushing in scatterplots. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 111–120. 1, 2, 4

[Gou09] Gough B.: *GNU scientific library reference manual*. Network Theory Ltd., 2009. 2

[KPV*17] Koytek P., Perin C., Vermeulen J., André E., Carpendale S.: Mybrush: Brushing and linking with personal agency. *IEEE Transactions on Visualization and Computer Graphics* (2017). 1

[Mah36] Mahalanobis P. C.: On the generalised distance in statistics. In *Proceedings National Institute of Science, India* (Apr. 1936), vol. 2, pp. 49–55. URL: http://ir.isical.ac.in/dspace/handle/1/1268. 2

[MKO*08] Muigg P., Kehrer J., Oeltze S., Piringer H., Doleisch H., Preim B., Hauser H.: A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data. *Computer Graphics Forum 27*, 3 (2008), 775–782. 2

[MS93] Minnotte M. C., Scott D. W.: The mode tree: A tool for visualization of nonparametric density features. *Journal of Computational and Graphical Statistics* 2, 1 (1993), 51–68. 3

[MW95] Martin A. R., Ward M. O.: High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of the 6th Conference on Visualization* (1995), VIS '95, IEEE Computer Society, pp. 271–278. URL: http://dl.acm.org/citation.cfm?id=832271.833844. 1

[NH06a] Novotný M., Hauser H.: Outlier-preserving focus+ context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 893–900. 3

[NH06b] Novotnỳ M., Hauser H.: Similarity brushing for exploring multidimensional relations. In *Journal of WSCG* (2006), vol. 14, Václav Skala-UNION Agency, pp. 105–112. 2

[Par62] Parzen E.: On estimation of a probability density function and mode. *The annals of mathematical statistics 33*, 3 (1962), 1065–1076. 1, 3

[R*56] Rosenblatt M., et al.: Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics 27*, 3 (1956), 832–837. 3

[RSM*16] Rados S., Splechtna R., Matković K., Duras M., Gröller E., Hauser H.: Towards quantitative visual analytics with structured brushing and linked statistics. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 251–260. 2

[T*93] Turlach B. A., et al.: *Bandwidth selection in kernel density estimation: A review*. Université catholique de Louvain, 1993. 3

[TKBH17] Turkay C., Kaya E., Balcisoy S., Hauser H.: Designing progressive and interactive analytics processes for high-dimensional data analysis. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 131–140. 1

[Ton12] Tong Y. L.: *The multivariate normal distribution*. Springer Science & Business Media, 2012. 3

[TT88] Tukey J. W., Tukey P. A.: Computer graphics and exploratory data analysis: An introduction. *The Collected Works of John W. Tukey: Graphics: 1965-1985 5* (1988), 419. 4

[WJ95] Wand M., Jones M.: Kernel smoothing london, 1995. 3