A Novel Cross-Validation Framework for Identification of Atmospheric Aerosol Types: Cluster Number Validation Methods for Unsupervised Learning

Bakker, V. de

A Novel Cross-Validation Framework for Identification of Atmospheric Aerosol Types

Cluster Number Validation Methods for Unsupervised Learning

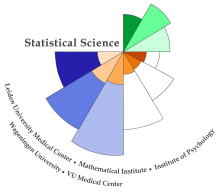
Vincent de Bakker (s1161326)

Thesis advisor: Dr. Wouter Weeda
External advisor: Dr. Johan de Vries

MASTER THESIS

Defended on August 27, 2018

Specialization: Statistical Science



STATISTICAL SCIENCE
FOR THE LIFE AND BEHAVIOURAL SCIENCES

To my father, who refused to give up, and my mother and sister, whose strength ensured no-one did.

Abstract

Aerosols are tiny particles of various kinds and compositions suspended in the atmosphere, some of which have a critical, adverse impact on public health. Hence, modelling the prevalence and distribution of these separate types is vital for giving shape to informed policy on air quality. In this work, methods are described to identify clusters of similar aerosol type mixtures in the Earth's atmosphere on a global scale, on the basis of microphysical data from the space-borne remote sensing instrument POLDER-3. We report an unsupervised learning approach using the Self-Organizing Map (SOM) and k-means clustering, which allows for clustering without *a priori* assumptions on existing aerosol types, nature or prevalence. Two methods are introduced to stabilize these clustering algorithms over multiple equal runs to manage their local optima convergence property: the k-means *nstart* option is extended to the SOM and a set-up is given for a new method, Expectation-Maximization-centered Mahalanobis clustering (EMcMc). A (repeated) v-fold cross-validation framework is presented to find the optimal number of clusters k in the data by means of cluster validation measures, currently including Prediction Strength and validated variants of the Silhouette Width. Using a separate test set, the method can be used to optimize a generic k, countering *overfitting*. A novel validation index is developed which extends the Silhouette Width to data sets with many observations (large N): the Gridded Silhouette Width. All described methods are implemented in the statistical software package R and shown to work for simulated examples, originating from scaled Gaussian distributions with varying degrees of overlap. Analysis of the POLDER-3 data indicated that using only four variables, 8 clusters can be found in a stable and reproducable fashion. The Silhouette indices did not appear to perform well for data so widely dispersed as here. The found clusters were characterized based on their variable distributions and geographical occurence, which proved to be feasible and meaningful for real-life interpretations. The proposed aerosol types were dust, marine, urban-industrial, smoke and mixtures thereof.

Keywords: aerosol typing; unsupervised learning; self-organizing map; k-means clustering; cluster validation measures; cross-validation; gridded silhouette.

Acknowledgements

I wish to thank dr. Johan de Vries for his dedicated help, advice and supervision at practically any moment I felt it was needed throughout the project. A heartfelt thank you to dr. Wouter Weeda is in place as well, for his contagious enthusiasm, always encouraging guidance, profound care and his willingness to step into this adventurous thesis project with me. Thank you both for making it possible for me to start this endavour at a motivating and stimulating site in the first place, for which I would also want to thank Matthijs van der Kooij for initiating that. I am grateful to our collaborators at the Netherlands Institute for Space Research (SRON), dr. Otto Hasekamp and dr. Antonio di Noia, for sharing the data, their experiences, knowledge and advice on how to proceed. I am indebted to my good friend Rhys Bird for tireless discussions on my work and in particular for his ideas which gave rise to the novel Gridded Silhouette Width method. I thank the Air Quality group, Instruments and Services department, other interns and remaining colleagues at Airbus Defence and Space Netherlands B.V. for creating a welcoming and stimulating environment, which allowed me to achieve my goals while feeling at home and having some good laughs too. Thank you, dr. Harald van Mil, for talking over my project and acting as a mentor on a personal level as well. I am truly grateful to all my friends, family and my girlfriend for their endless support and patient, understanding, but above all warm care in what has probably been the most difficult time of my private life, in parallel to my thesis project. You helped me manage.

Contents

1 Introduction

Atmospheric aerosols are short-lived, small solid or liquid particulate matter (PM) suspended in the Earth's atmosphere. They influence the Earth's energy budget by scattering and absorbing sunlight, thereby cooling and warming the planet, respectively. As aerosols vary greatly in chemical composition and radiative properties and can have both cooling and warming effects, their exact contribution to climate change has been difficult to establish. Therefore, they remain one of the largest uncertainties in models describing climate change, as also reported by the Intergovernmental Panel on Climate Change (IPCC) [1].

Aerosols are furthermore a prime threat to public health as a major constituent of air pollution. In this field, particulate matter is often categorized by size, where coarse particles have a diameter less than 10 μm (PM_{10}), fine particles less than 2.5 μm ($PM_{2.5}$) and ultrafine particles less than 0.1 μm ($PM_{0.1}$). Aerosols with a diameter less than 10 μm are not effectively filtered out by the nose and the upper airways when inhaled and especially finer particles have a high probability of being deposited deeper in the conducting airways and alveoli. Exposure to particulate air pollution has been associated with elevated morbidity and mortality levels and with a wide variety of adverse health effects, including cardiovascular and respiratory diseases and lung cancer [2, reviewed extensively by 3, 4, 5].

These critical global issues make identifying aerosol presence and measuring aerosol properties in the atmosphere vitally important to establish informed policies based on accurate atmospheric models. There are various ways in which this can be achieved. In situ filter measurements are mostly accepted as standard, but provide local information only and rely on the filters for particle discrimination. Ground-based remote sensing instruments, such as the sun photometers from the Aerosol Robotic Network (AERONET), can take the whole atmospheric column into account, but are still spatially limited to the site they are stationed at. In contrast, aircraft- and satellite-borne instruments allow covering regions and the latter even the complete globe. Several space-borne remote sensing instruments have already provided information on aerosols, including Multiangle Imaging Spectroradiometer (MISR) [6], Cloud-Aerosol Lidar and Infrared Pathfinder Satellite Observations (CALIPSO) [7], Moderate Resolution Imaging Spectroradiometer (MODIS) [8] and Polarization and Directionality of Earth's Reflectances (POLDER) [9, 10].

The POLDER-3 instrument on the PARASOL spacecraft is dedicated to measuring aerosol microphysical parameters from sunlight scattered by the atmosphere, via spectral, polarimetric and scattering angle dependencies of the observed radiance. These microphysical parameters are the starting point for aerosol classification and therefore a POLDER-3 dataset is suitable for this objective.

Nonetheless, identification of different aerosol types remains a difficult task. This is however of particular importance for the issue concerning public health. It is unlikely that all types of aerosols pose an immediate risk to human health; sea salt is presumably less hazardous than soot. Notwithstanding, it might be challenging to pinpoint principal damaging agents among aerosol classes [11, 12]. Multiple approaches already exist to identify and characterize aerosol types on the basis of microphysical parameters. In general, they make use of ground-based data in one way or another, sometimes in combination with space-borne data [e.g. 7, 13, 14, 15, 16, 17, 18, 19]. These are, however, always based on using pre-defined aerosol type labels and therewith on assumptions on the number, nature and prevalence of these aerosol types. Moreover, the remote sensing data involved is measured on massive atmospheric columns, in which it is unlikely that solely one type of aerosol is present. So, every observation represents a weighted average for each measured microphysical variable, depending on the aerosols that were present in the column corresponding to that observation.

Taylor et al. [20] present a clever way to deal with the fact that observations actually consist of aerosol mixtures. They applied NASA's GOCART algorithm to classify aerosols and consecutively used the k-means clustering algorithm to find commonly occuring mixes of these classes. GOCART does not use microphysical parameters as input data, but rather meteorological data [21]. Using AERONET data, the found mixtures are characterized with respect to their average microphysical

parameters. However, this still requires the definition of a set number of 'pure' aerosol types beforehand, in the first GOCART classification step.

Here, we propose to apply unsupervised learning methods as well (like k-means), but directly on space-borne data. The rationale behind this approach is that one would attempt to establish how many discernable clusters exist in the data, instead of fixing this number beforehand. The resulting clusters are then assigned a (mixed) aerosol type label afterwards, based on geographical prevalence and distributions of microphysical parameters within the clusters. In brief: we wish to classify clusters, instead of clustering classes as Taylor et al. [20] did.

To this end, an exploratory study has been carried out before, involving the application of a Self-Organizing Map (SOM) [22] as clustering method on POLDER-3 data of the year 2006 by Visser [23]. The data points represent atmospheric columns on a certain day in several dimensions (microphysical parameters) and were clustered into 9 groups by a single run of a SOM algorithm. Visser simply looked for a larger number of clusters (9) than he would expect to be able to distinguish in the data, reasoning he could always merge similar clusters in hindsight. Data within clusters should be comparable in their microphysical parameters and thus the corresponding atmospheric columns should be comparable in their aerosol contents. Finally, aerosol type labels were assigned to the clusters, based on their geographical occurences and the within-cluster variable distributions.

In this thesis, work is presented that elaborates on that study. We aimed to expand upon the same idea, creating a more standardized methodological framework around it, thereby adding to the scientific rigour. Practical implementation of the proposed and developed methods in the statistical software package R [24] enabled us to examine the workings and performance of this use of unsupervised learning methods for the specified goal of aerosol typing.

This extension of the work of Visser [23] hinged on three main pilars: (1) use of alternative clustering algorithms, (2) application of clustering stabilization techniques and (3) cluster number validation. Firstly, we aimed to develop a generic framework, which is not dependent on the use of SOM as clustering algorithm, but allows for the use of other ones as well. Here, we analyzed the outcomes for the k-means clustering algorithm too, which is much simpler than the more complex SOM. It is arguably the most popular clustering algorithm available and is well described and studied; therefore it can be regarded as a benchmark method [e.g. reviews by 25, 26].

Secondly, it is well known that clustering algorithms generally converge to local optima, meaning they yield different results every time they are invoked – even if all settings and data are kept the same [e.g. 27]. We describe and include two techniques to stabilize clusterings over multiple equal runs.

Lastly, and most crucial, our framework offers a way to take on what is arguably the most profound issue in unsupervised learning: to determine the number of clusters naturally occuring in the data. To this day, this topic is subject to discussion and development of new perspectives and approaches. There is an abundance of validation measures for cluster numbers at present and they can be based on different principles, as cluster stability or cluster cohesion and separation [28]. The framework developed here uses cross-validation to optimize the cluster number k, by means of (variants of) two of such measures: Prediction Strength [29] and Silhouette Width [30]. Solutions are presented to circumvent the computational problems that arise for both validation measures when the number of observations N in the data is large, which is typical for the specific case study at hand. In the case of the Silhouette Width, this has led to a novel approximation method, the Gridded Silhouette Width, which is presented in this work too.

Summarizing, I have developed an extensive methodological framework to determine the number of discernable clusters in a given data set using various clustering algorithms and validation indices by means of cross-validation. The described functionalities can be used for both exploratory and prediction purposes; in the latter case the cross-validation scheme also provides a means to avoid overfitting. Practical use of the methods is straightforward and additionally allows for stratified fold splitting. The codes are constructed in such a way that extension to other clustering algorithms and validation measures should be easy to accomplish.

The workings and performance of the proposed methods are demonstrated in both simulation

studies and on real-life data: the POLDER-3 data of the year 2006. The ultimate goal was to derive stable clusters from the microphysical variables and label the thus found clusters as (mixtures of) aerosol types. This can grant insight in the spatial and temporal occurence of aerosol types, which has especially important implications for public health. The thesis is set-up as follows: section 2 contains the theory behind and descriptions of the statistical methodology; in section 3 the results of the application of the presented methods to simulated and POLDER-3 data are outlined; this is followed by an overall conclusion and extensive discussion on the methods and outcomes in section 4, including some final remarks and future outlooks.

2 Theory and Methodology

For all practical purposes and implementation of the methods, the statistical software R (version 3.4.3) was employed [24]. Additional software packages and used functions will be named when they appear throughout the thesis. The main user-defined functions developed for this thesis will be described in order of appearance in this section and a list including documentation is provided in Appendix C.

2.1 Data

The full POLDER-3 data of the year 2006 was provided by the Netherlands Institute for Space Research (SRON) in one netCDF file per month. Each observation corresponded to a one-by-one latitude-longitude degree grid box on the surface of the Earth, measured on a daily basis. Only observations were used for which all variables were measured; so-called "complete cases". In some areas, measurements were not successful, or not on all days, e.g. towards the poles and above part of the Amazone rainforest. These missing values were discarded, leaving $N = 1,131,324$ observations.

The full set of 55 variables is included in Appendix B. Some of the variables that represent microphysical properties were measured at several wavelengths. In this work, only four parameters were used:

(1) Single Scattering Albedo;
(2) Real Refractive Index;
(3) Sphericity;
(4) Angstrom Exponent,

of which the interpretation is explained to a larger extent below. The corresponding feature space data plots are shown in Figure 2. These four were selected on grounds of previous experience and expert knowledge present at SRON. Since the goal here was to work out the scheme for aerosol typing with unsupervised learning and to provide more of a proof of concept, only this subset of variables used. Once the scheme is shown to work, one might consider shifting towards a more intense machine learning approach, in which all data is used as input. However, such an objective lied beyond the scope of this thesis.

2.1.1 Data Extraction to R

A wrapper function, `extract.nc()`, was constructed to read the data files into R. The wrapper calls to the existing `nc_open()` function of the **ncdf4** package [31]. It allows the user to select variables and specify time periods for which the data is to be loaded into R. The function then puts the data from the different months (thus files) together and returns an object of the `data.frame` or `array` class. Some extra functionalities are available, as setting upper and lower limits for certain relevant parameters and finding fill values above a specified threshold. If the user opted for a data frame to be returned, the data will be accompanied by latitude and longitude degrees of each observation, their day, month and season of measurement and a pixel ID number for every unique latitude by longitude degree grid box. Moreover, the function calculates the Angstrom Exponent from the Aerosol Optical Thickness variables, if desired. The next subsection will cover how this is done, along with a more detailed description per variable, supplemented with plots depicting their seasonal means on world maps.

2.1.2 Variable Description

The Single Scattering Albedo $\omega_0(\lambda)$ (SSA) is the ratio between the quantity of sunlight scattered in an atmospheric column and the total extinction of sunlight in that column at a certain wavelength λ:

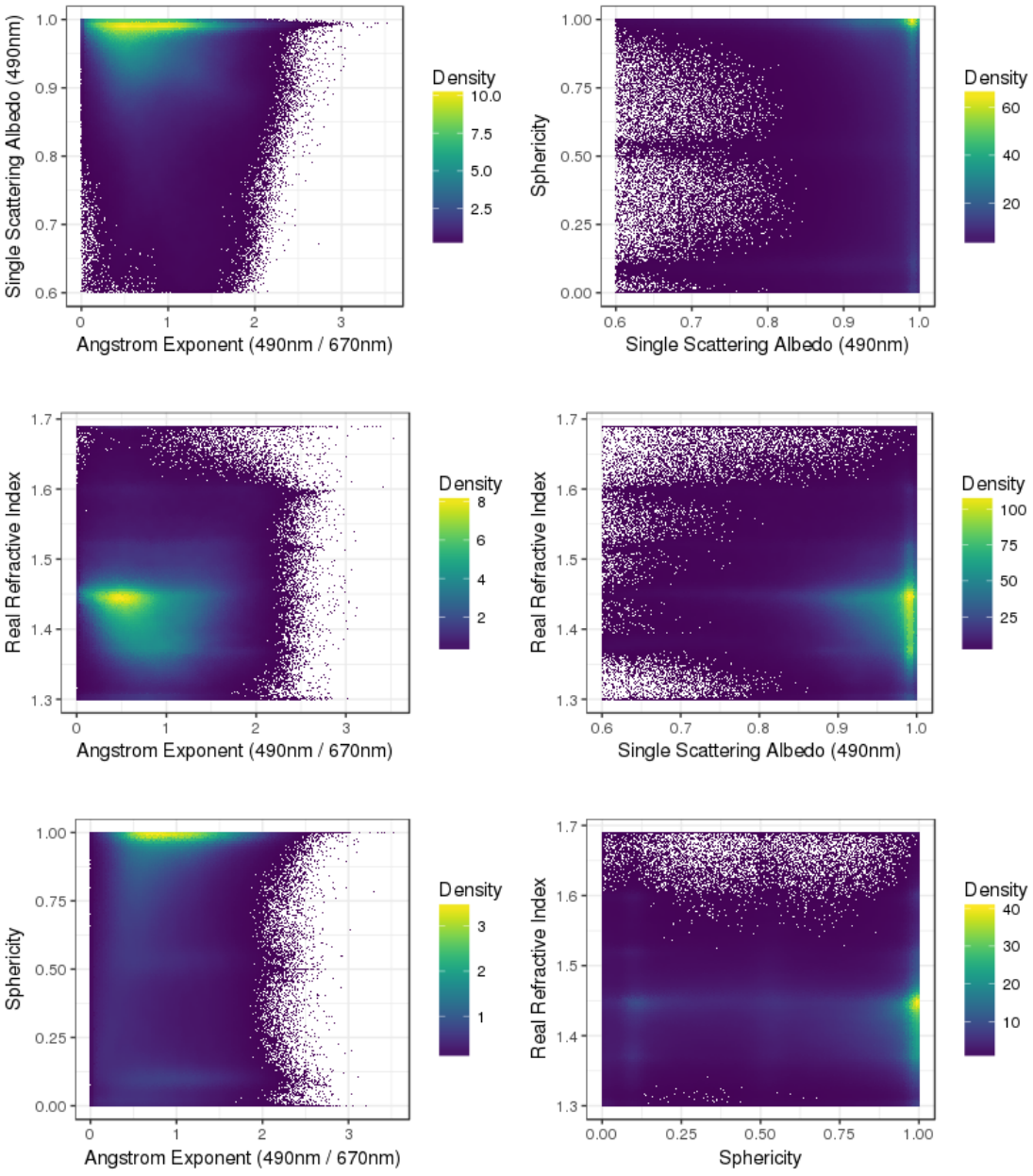


Figure 2: **The raw POLDER-3 data of 2006 in feature space.** Colour indicates density of observations. Only complete cases are depicted, N = 1,131,324.

$$SSA = \frac{Scattering}{Scattering + Absorption};$$

the extinction consists of scattering and absorption together, i.e. the sunlight that does not reach the surface of the Earth. Therefore, lower SSA values indicate that the aerosols in that column are more absorbing, such as soot, or smoke in general.

Here, the SSA is measured at 490 nm, in line with the original study, on which this work is based [23]. The variable has an upper bound of 1 (as it is a fraction) and is cut at a lower bound of 0.6 (Figure 2), which is the typical range reported for this parameter [14]. Indeed, studies that use (reference clusters based on) AERONET measurements ordinarily do not reveal lower SSA values even for the absorbing clusters [e.g. 18, 19, 32] and the same cut-off value was used by Visser [23]. Note however, that the `extract.nc()` function does allow for other lower bound cut-off values, if desired. Figure 3 depicts per grid cell the mean SSA values for each season (northern hemisphere). Low values can mainly be observed over areas known for wildfires, e.g. south of the Sahara desert in Africa, Australia and the Iberian peninsula [33, 34], although the average SSA values over the Amazone rainforest appear markedly high.

The Real Refractive Index m_r (RRI) is informative on the propagation of light through the atmosphere: higher values indicate the light is bent, or refracted, to a larger degree, which is dependent on the composition of the medium it travels through (here: the aerosol-containing atmosphere). Some microphysical variables, like RRI, typically follow a bimodal distribution with one peak for finer particles, and another for more coarse particles. For this reason, there are two RRI data products: one for the fine mode fraction and one for the coarse mode fraction. Although the collaborators at SRON indicated either of the two variables could be used, they advised to start with the fine mode fraction, which we did for this thesis project. The RRI is considered to be spectrally neutral in the retrieval, therefore no specific wavelength is reported. The RRI of water is known to be 1.33 - 1.34 for wavelengths between 350 nm and 1000 nm [35]. Indeed, higher values are associated with a decrease in water content [36]. As can be seen in Figure 2, RRI values are bound between 1.3 and 1.7, with the bulk of the data approximately between 1.35 and 1.45. The seasonal means of RRI are shown in Figure 4 and seemingly tend to be higher over land masses.

The sphericity parameter captures the fraction of aerosols present that are spherical in shape; the variable is thus bound between 0 and 1. Again, there is a fine mode fraction and coarse mode fraction variable. The latter was used in this study, initially to resemble the original work of Visser [23]. Presumably, the sphericity variable is particularly useful to identify dust aerosols, as these are highly non-spherical [e.g. 19]. Indeed, low sphericity values appear over desert regions, as the Sahara Desert and Arabian Desert (Figure 5). A zone that stands out for its extremely low mean sphericity, is the Taklamakan desert in the west of China: it is isolated from seas and presumably holds predominantly dust.

The last parameter included in this work, is the Angstrom Exponent α (AE). The AE reports on aerosol particle size and is calculated using the Aerosol Optical Thickness τ (AOT) at two different wavelengths:

$$\alpha = -\frac{\log(\frac{\tau(\lambda_1)}{\tau(\lambda_2)})}{\log(\frac{\lambda_1}{\lambda_2})},$$

where λ_1 is smaller than λ_2. The AOT is a measure of aerosol abundance in the atmosperic column. Since the AOT decreases faster with wavelength for smaller particles, the presence of more fine particles makes the numerator grow ($\tau(\lambda_2)$ decreases faster than $\tau(\lambda_1)$ in that case). This inflates α, as the denomenator is always negative. Therefore, low AE values represent the presence of larger particles (coarse), and high AE reveal the presence of more smaller particles (fine). Used AOT wavelengths in this study are $\lambda_1 = 490$ nm and $\lambda_2 = 670$ nm [as in 23], but others can be specified in the `extract.nc()` function as well. Only AOT (490 nm) values of 0.1 and higher were used, because lower values have been shown to be less reliable in the midvisible

spectrum [6]. Additionally, AE values were cut at a lower bound of 0 for reasons of reliability too, on advice of SRON experts. Low values can on average be observed over deserts and oceans (Figure 6). Indeed, sea salt and dust are relatively coarse aerosols. Strikingly, the most coarse particles appear to be over the Atlantic ocean west of the Sahara, which might correspond to dust transport.

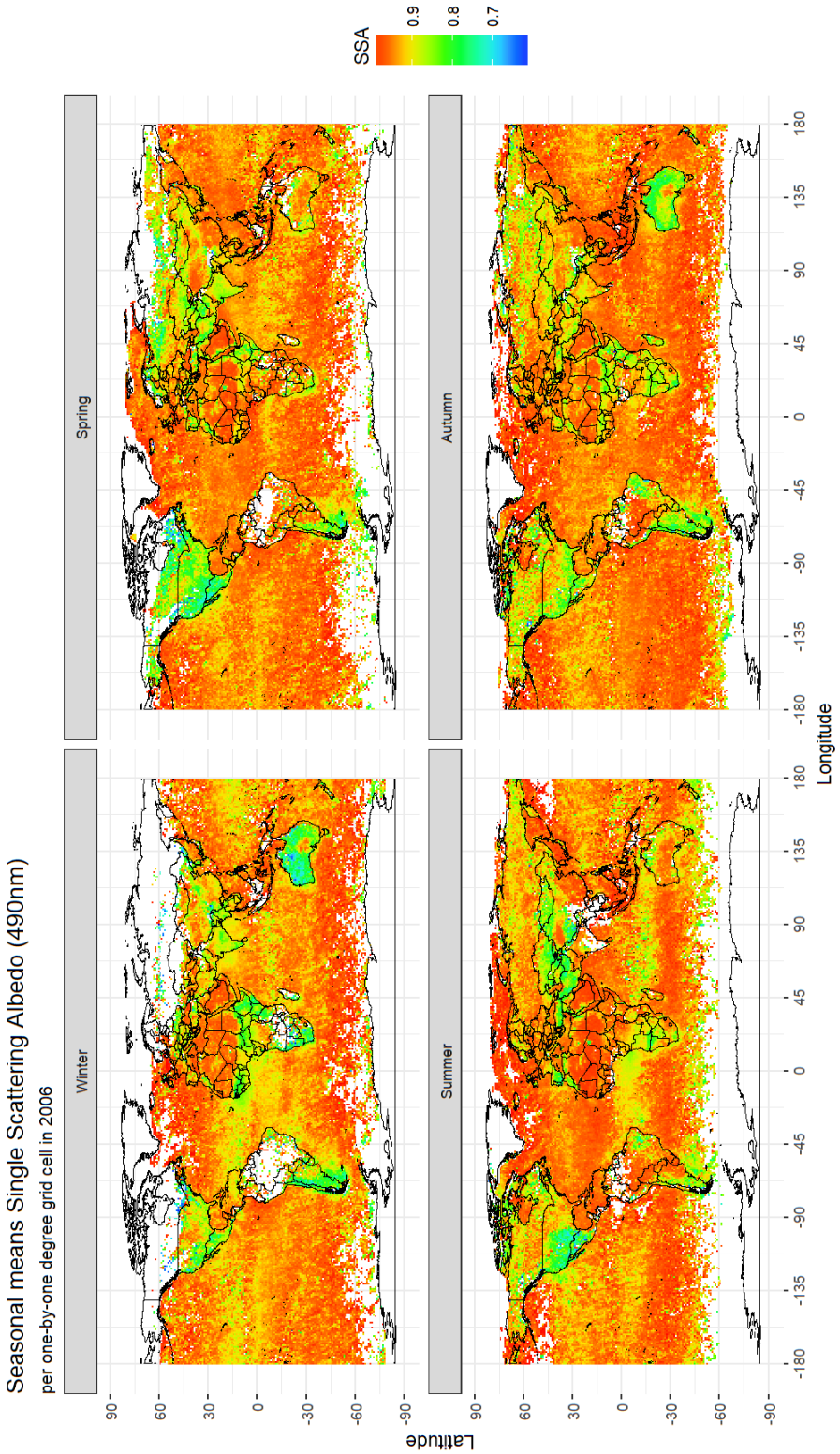


Figure 3: **Seasonal (northern hemisphere) means of Single Scattering Albedo (SSA) on a geographical map.** Number of observations is not equal over grid cells.

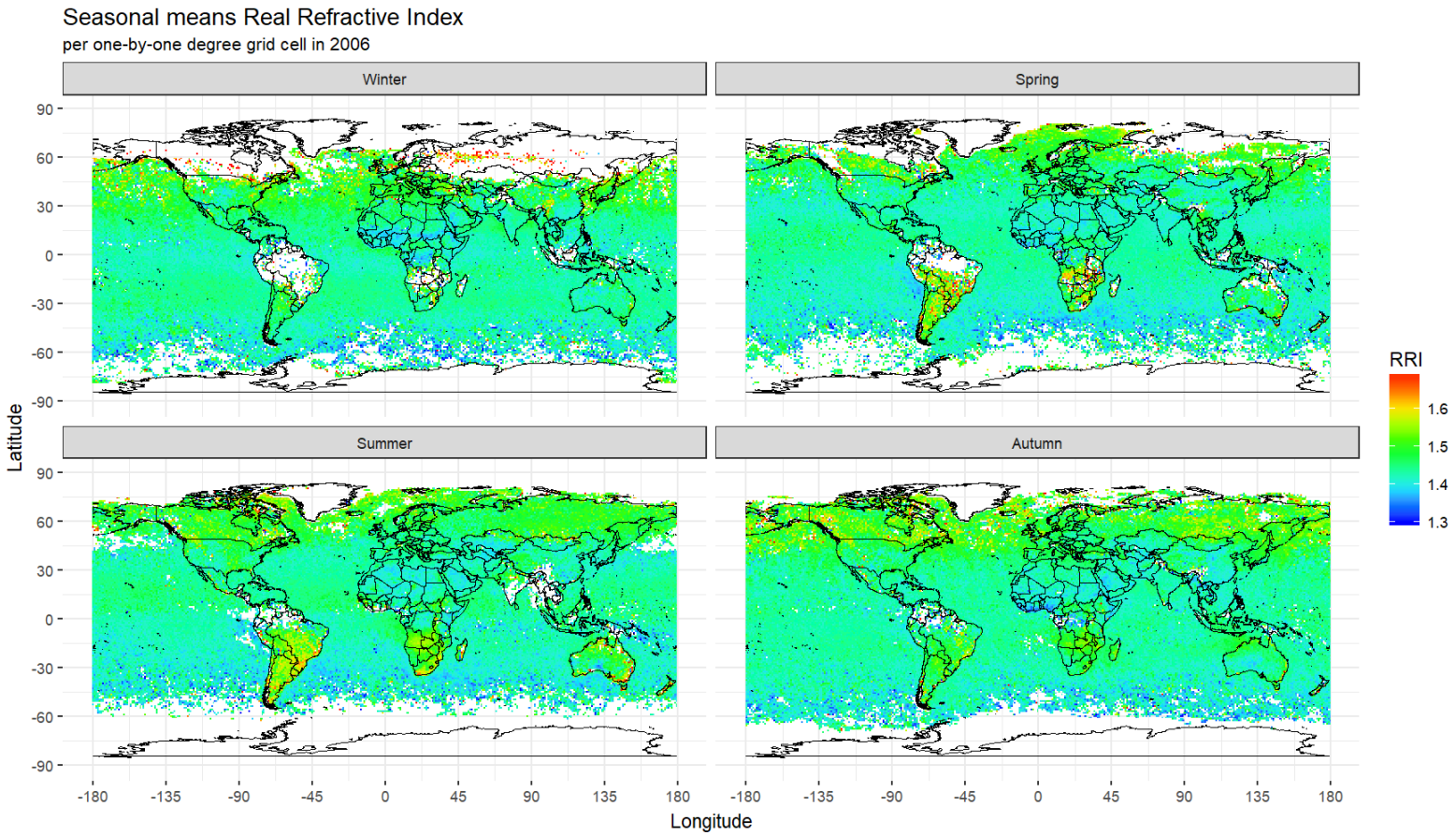


Figure 4: **Seasonal (northern hemisphere) means of Real Refractive Index (RRI) on a geographical map.** Number of observations is not equal over grid cells.

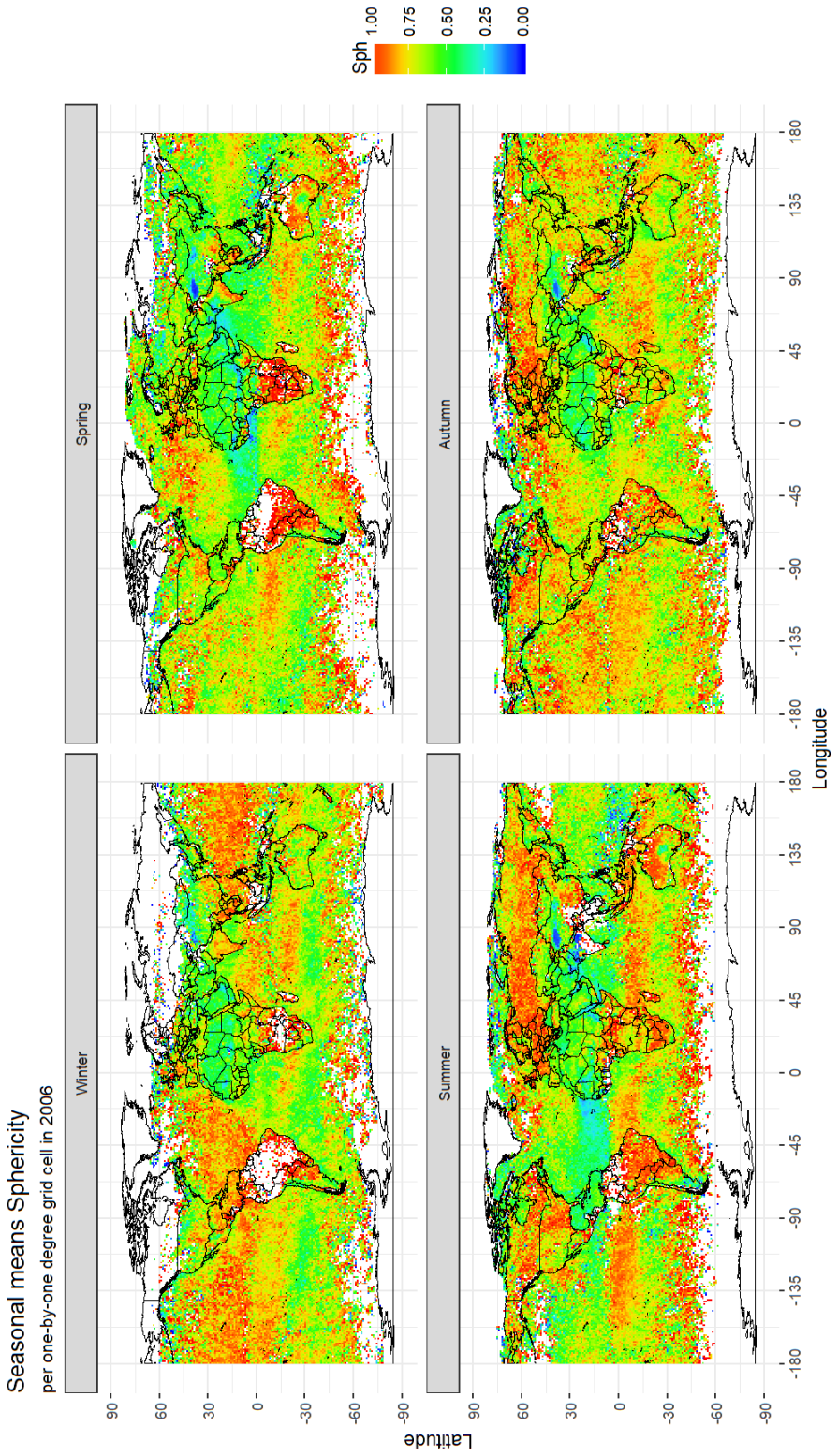


Figure 5: **Seasonal (northern hemisphere) means of Sphericity (Sph) on a geographical map.** Number of observations is not equal over grid cells.

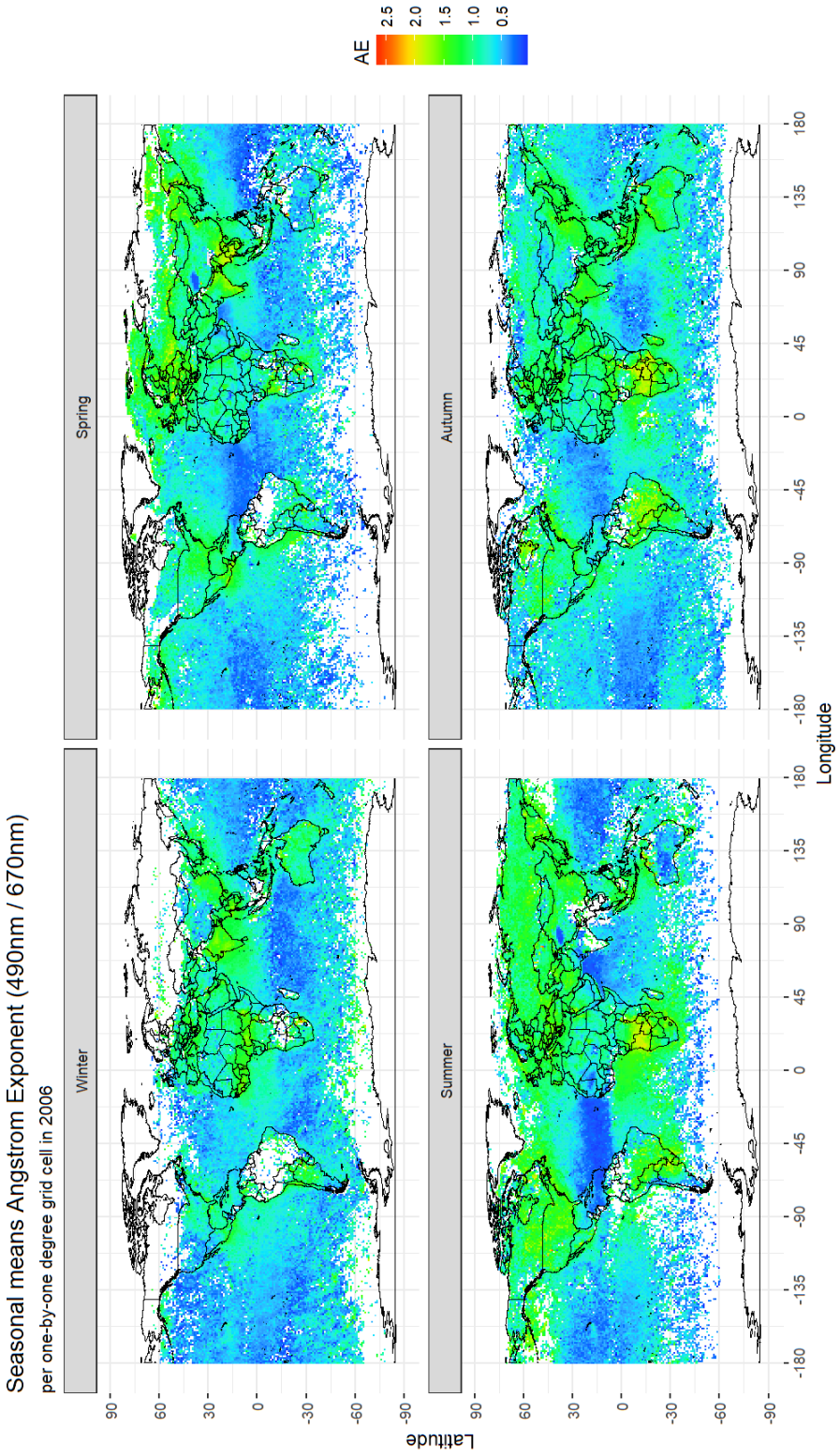


Figure 6: **Seasonal (northern hemisphere) means of Angstrom Exponent (AE) on a geographical map.** Number of observations is not equal over grid cells.

2.2 Clustering Methods

The core objective here essentially boils down to grouping the N observations of a p-dimensional data set $\mathbf{X}$ into a set number of k clusters based on some ground of similarity. This is a form of *unsupervised learning*, or "learning without a teacher". Opposed to *supervised learning*, we have no information on the "correct" class for the observations: we have no "teacher" to tell us whether we are doing well or not, by some measure.

Suppose that, besides the data $\mathbf{X}$, we do also have such a response variable $\mathbf{y}$, containing the correct class label y_i for each i^{th} observation of $\mathbf{X}$. Instead of **clustering** (unsupervised), we would be able to perform **classification** (supervised): finding the optimal function $f(\mathbf{X})$ for predicting $\mathbf{y}$ given the input values $\mathbf{X}$. Since we know what the answer should be, at least for the given values of $\mathbf{X}$, we can use some measure of how good or how bad a candidate function f is doing. Such a measure of "badness-of-fit" is called a *loss function* $L(\mathbf{y}, f(\mathbf{X}))$ (e.g. squared error in regression: $(\mathbf{y} - f(\mathbf{X}))^2$). We can then define the *risk* of the candidate function f by taking the expected loss of f (e.g. the mean squared error), in which case a smaller risk is better. Equivalently, we could state that we wish to find the function f, for which the *expected prediction error* (EPE) is minimal: a process called *empirical risk minimization*. In a prediction setting, the found function f could be used to assign classes to new observations, thereby labelling them.

In our case, we wish to assign aerosol type labels ($\mathbf{y}$) to our observations ($\mathbf{X}$). However, since we avoid making assumptions on the labels beforehand, $\mathbf{y}$ is not known and therefore application of supervised learning techniques is impossible here. Alternatively, the only available information is $\mathbf{X}$, and so the goal shifts to finding clusters of observations therein which are more similar in the p parameters to each other than to other clusters of observations. Since we are currently dealing with exclusively continuous numeric variables, we have chosen to define this similarity as distances in feature space. The clustering algorithms applied here thus attempt to identify clusters of data that are close to each other in feature space, representing these clusters, or feature space regions, by prototypes, or centroids. Such prototype-based clustering algorithms generally minimize some measure of within-cluster dispersion as loss function. This minimum then represents the "best" solution, optimally describing the structure in $\mathbf{X}$ alone and labelling the observations in the same cluster by their centroids.

The two clustering algorithms used in this work were k-means clustering and the Self-Organizing Map, which are described in the next subsections. Prior to clustering with any algorithm, the data were scaled with feature scaling:

$$\mathbf{x}^* = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})},$$

where $\mathbf{x}^*$ is the rescaled vector of $\mathbf{x}$. Feature (back)scaling functions were made in R, the application of which over all observations per variable standardizes them to the $[0, 1]$ interval.

2.2.1 k-means Clustering

Arguably the simplest and most widely used clustering algorithm is the *k-means* algorithm and it is therefore regarded as a benchmark algorithm in this work. The method has been reviewed extensively, e.g. by Steinley [25] and Jain [26]. The fundamentals of the clustering procedure were first published independently under various names, starting about 60 years ago [37, 38, 39, 40, 41]. Later, other versions of k-means algorithms have been developed, as the computationally more efficient one by Hartigan and Wong [42], which is the default for the standard R `kmeans()` function.

Given a p-dimensional data set $\mathbf{X}$ of size N, k-means attempts to divide $\mathbf{X}$ into a pre-defined k number of optimal clusters C so that the overall within-cluster sum of squared errors (WCSS) is minimized:

$$\arg\min_{C} \sum_{j=1}^{k} \sum_{i=1}^{n} \|\mathbf{x}_{ij} - \bar{\mathbf{x}}_j\|^2,$$

where $\mathbf{x}_{ij}$ is a vector of length p representing the i^{th} observation in cluster j of size n with mean $\bar{\mathbf{x}}_j$ as vector of equal length and where Euclidean (L_2) distances are used. The found means serve as cluster centroids, or protoypes, dividing the feature space in Voronoi cells.

This is an iterative algorithm, at its core consisting of two steps:

(1) Given a set of centroids, assign each observation to its closest centroid, thus creating clusters of observations;
(2) Given the assignments of the observations to the clusters, update the cluster centroids to become the means of their clusters.

When the assignments in step (1) no longer change more than a certain, very small threshold, the algorithm has converged. Nonetheless, k-means algorithms commonly have an option to specifiy the maximum number of iterations (e.g. 10 in the standard `kmeans()` function in R, as suggested by the algorithm developers Hartigan and Wong [42]). Initialization of the method is usually achieved by randomly picking k observations from $\mathbf{X}$ to use as first centroids.

2.2.2 Self-Organizing Map

The Self-Organizing Map (SOM), or Self-Organizing Feature Map (SOFM) is a type of artifical neural network (aNN) that is used for unsupervised learning tasks [22]. Even though multiple questions regarding the mathematical properties of the SOM remain to be answered [43], the method has been effectively applied in a wide range of fields and various variants of the SOM have been developed [reviewed in 44]. Practical implementation in R is possible by means of the **som** and **kohonen** software packages [45, 46], the latter of which is used in this work. Essentially, SOMs are employed for two purposes: to cluster data and to visualize high-dimensional data in two-dimensional space. The latter is also possible due to a particular characteristic of the SOM: it preserves the topology of the centroids on a 2D grid.

As in k-means clustering, the SOM approach aims to assign each datum to a representative, i.e. a centroid – also known as a protoype, or in the jargon of the SOM a node, neuron or even a model [44]. Again, the number of centroids k the algorithm looks for in the given data has to be defined *a priori*. Unlike k-means, however, these centroids are fixed on a grid, whereas their *weight vectors*, also called codebook vectors, move iteratively through the feature space. So, every centroid $1 \leq j \leq k$ has grid coordinates $\mathbf{m}_j$ and a weight vector $\mathbf{w}_j$, the latter of which has the same dimensionality as the data (and is in fact what we would call the centroid in k-means clustering). This idea is depicted schematically in Figure 7, in which you can see that the centroids are arranged on a grid (middle panel), which is square here, and have corresponding weight vectors in feature space (left and right panels). The other details shown in the figure will be explained later in this section. Whereas the grid coordinates of each centroid remain the same throughout the complete clustering process, their weight vectors change, virtually moving through the parameter space looking for dense masses of data and fitting to them.

Generally, these grids have either a rectangular or hexagonal shape, albeit other designs have been examined [47]. Toroidal grids are possible as well, connecting opposite edges of the grid (i.e. creating a donut shape). The aforementioned preservation of topology entails the fact that the SOM algorithm trains centroids in such a manner that those closer to each other on the grid will become more similar among themselves in their weight vectors than to centroids located farther away. In other words: the centroids *become spatially, globally ordered*, as put by the discoverer himself [44]. This ultimately yields the two-dimensional map (namely, the grid) on which the data is projected and which allows for relatively easy interpretation of thus clustered higher-dimensional data.

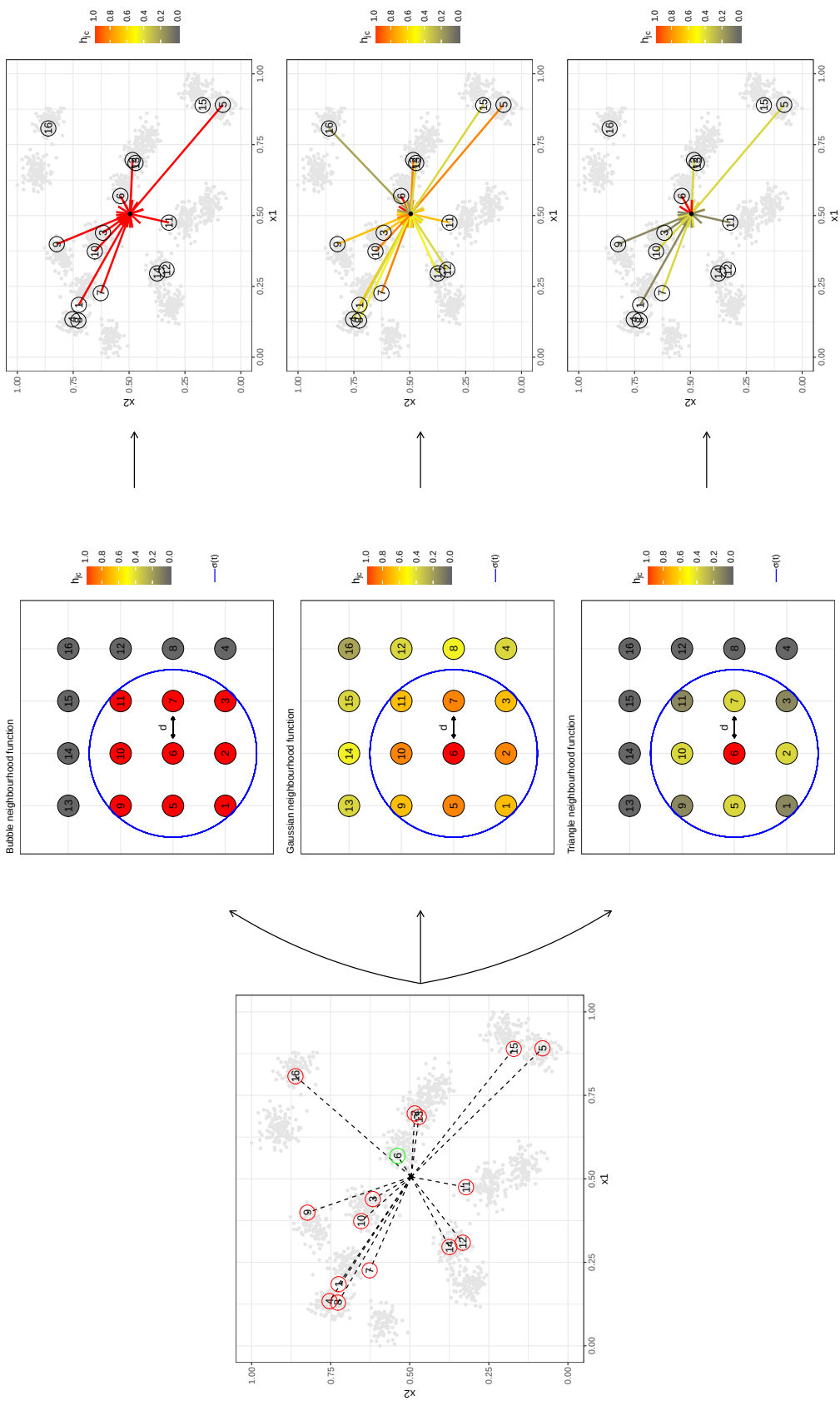


Figure 7: **Schematic illustration of one iteration of SOM training within a certain epoch.** The image shows how the centroids here get updated as they are presented with one observation. The left panel shows that centroid 6 is the Best Matching Unit (BMU, green) for the given datum, as it is closest in feature space. The radius σ for the current epoch was here set to 1.6: depending on the chosen neighbourhood function h_{jc} and the distance d between each centroid and the BMU, the weight update fraction is determined for each centroid in the middle panels, which depict the grid. Together with the learning rate α (not shown here, constant per epoch), h_{jc} determines which centroids' weight vectors move in the direction of the observation in feature space and to what extent; this is depicted in the panels on the right.

The original algorithm follows a recursive, step-wise procedure: all data points are presented to the centroids (or more specifically: to their weight vectors) one by one, mostly in a random order, updating the centroids consecutively. The whole process is then repeated a set number of cycles, or *epochs*, T, so that every datum has been used an equal number of times. The default number of epochs in the **kohonen** package is 100. Let again $\mathbf{X}$ be a p-dimensional data set containing N observations, which we wish to divide into k clusters. Hence, each j^{th} cluster centroid on the grid $\mathbf{m}_j$ has a weight vector $\mathbf{w}_j$ of length p as well. For each observation $\mathbf{x}_i$ of the data, the closest centroid, indexed by c, is determined, typically in Euclidean distance: the "winner", or the *Best Matching Unit* (BMU). Observe, thus, that this clustering algorithm is also based on measuring similarity in terms of distances. Now the BMU $\mathbf{m}_c$ for the i^{th} observation of $\mathbf{X}$ is found by:

$$c(i) = \arg \min_{1 \leq j \leq k} \|\mathbf{x}_i - \mathbf{w}_j\|.$$

This winner $\mathbf{m}_c$ is subsequently updated to better match the input vector $\mathbf{x}_i$; i.e. $\mathbf{w}_c$ is moved towards the datum. In addition, centroids within the spatial neighbourhood of the BMU on the grid are also updated; this principle of the SOM algorithm ascertains topology preservation. Although the initial neighbourhood size itself is simply set by a user-specified radius, a SOM is characterized by the fact that the neighbourhood size diminishes as a function of time, along the progression through the epochs. The neighbourhood radius σ for epoch $t \in T$ is for instance commonly expressed in the form of an exponential decay function:

$$\sigma(t) = \sigma_{t-1}\, e^{-\frac{t}{\lambda}},$$

where λ is a time constant determining the decay speed. Nevertheless, the SOM algorithm in the **kohonen** package shrinks σ linearly.

Accordingly, whether a centroid weight $\mathbf{w}_j$ is conformed to a datum $\mathbf{x}_i$ in a certain epoch t depends on its proximity on the grid $\mathbf{m}_j$ to the BMU $\mathbf{m}_c$. This is expressed in the form of the neighbourhood function $h_{jc}(t)$, the simplest version of which is the *bubble function*:

$$h_{jc}^{bubble}(t) = \begin{cases} 1, & \text{if } \|\mathbf{m}_j - \mathbf{m}_c\| \leq \sigma(t); \\ 0, & \text{if } \|\mathbf{m}_j - \mathbf{m}_c\| > \sigma(t), \end{cases}$$

where every $\mathbf{m}$ has the same dimensions as the grid (mostly two). With the bubble function, only the centroids within the radius of the given epoch from the BMU corresponding to that observation get the full weight update. Naturally, this includes the BMU itself as well, as in that case $\|\mathbf{m}_j - \mathbf{m}_c\| = 0$. A more elegant neighbourhood function is a Gaussian, which allows the algorithm to gradually decrease the update intensity of the centroids with their distance from the BMU on the grid:

$$h_{jc}^{Gaussian}(t) = \exp\left[-\frac{\|\mathbf{m}_j - \mathbf{m}_c\|^2}{2\sigma^2(t)} \right].$$

From the equation above, it can be observed that the neighbourhood function increases the closer a centroid $\mathbf{m}_j$ is to the BMU $\mathbf{m}_c$ of a certain input vector $\mathbf{x}_i$, with a maximum value (and therewith update intensity) of 1 when $\mathbf{m}_j = \mathbf{m}_c$. One final example of a neighbourhood function is a *triangular function*:

$$h_{jc}^{triangle}(t) = \begin{cases} 1 - \frac{\|\mathbf{m}_j - \mathbf{m}_c\|}{\sigma(t)}, & \text{if } \|\mathbf{m}_j - \mathbf{m}_c\| \leq \sigma(t). \\ 0, & \text{if } \|\mathbf{m}_j - \mathbf{m}_c\| > \sigma(t). \end{cases}$$

The triangle function refines the bubble function: it also assigns no update to centroids outside the BMU its neighbourhood radius, but appoints linearly increasing update intensities with BMU nearness to those centroids within this radius. The neighbourhood functions presented

here are illustrated in Figure 8. In the R **kohonen** package, only the bubble function and the Gaussian function are available for use.

The shrinkage of the radius through the epochs leads to fewer (bubble function), mitigated (Gaussian function) or both (triangle function) updates through time for non-BMU centroids in every iteration. It also ascertains that from some epoch onwards, only BMUs are updated. The shrinkage of σ furthermore endorses *local* ordering of the centroids in downstream epochs, after an initial *rough* ordering phase with a wide radius in the earlier epochs.

The weight update intensity is also controlled by the learning rate, α, besides by the neighbourhood function. Analogous to the radius, the learning rate decays with t as well and the exact same exponential decay function can be applied, replacing σ with α. Again, the **kohonen** package employs a linear decay function. The shrinking learning rate contributes to the convergence of the algorithm through reduction of overall weight update intensities in later epochs.

Taken together, the weight vector $\mathbf{w}_j$ of a given centroid $\mathbf{m}_j$ gets updated as observations from $\mathbf{X}$ are presented to the grid through the epochs $t \in T$ by the following formula:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + h_{jc}(t)\alpha(t)(\mathbf{x}_i - \mathbf{w}_j(t)),$$

where i indexes the observations. A much simplified schematic illustration hereof is shown in Figure 7. The formula shows that each centroid weight is equal to its weight in the previous iteration, plus a fraction of its distance to the observation it is presented with, moving it a little in that direction. The size of that fraction corresponds to the extent to which the centroid weight is pulled towards the observation. It has both spatial and temporal influences: the former, because the aforementioned fraction decreases with distance on the grid of the given centroid to the BMU of that observation; the latter, because both the neighbourhood radius and the learning rate shrink as the epochs go by.

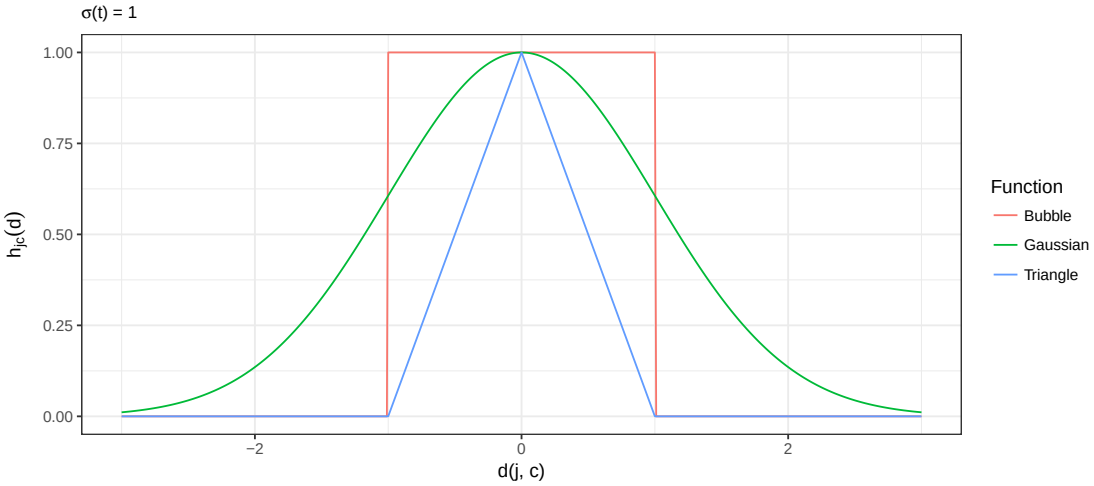


Figure 8: **The described neighbourhood functions that can be used in the Self-Organizing Map (SOM) algorithm.** The function of choice determines in part the weight update of any centroid when presented with the i^{th} observation of the data in a SOM training process. The distance on the grid between the j^{th} centroid and the one closest to $\mathbf{x}_i$ in feature space (the Best Matching Unit or BMU, indexed with c) is denoted by $d(j, c)$. When the centroid in question is the BMU itself, $d = 0$ and the weight update fraction accounted for by the neighbourhood function $h_{jc}(d)$ is maximum: 1. The radius σ, which shrinks through the epochs $t \in T$, is in this illustration set to 1.

Unless stated otherwise, we used the default settings of the **kohonen** package in this work. The form of the grid may play a role in the SOM training process: it can determine the distances between centroids on the grid. This in turn determines to what extent centroids other than the BMU get updated. Imagine the grid in Figure 7 to be 8-by-2, for instance, instead of 4-by-4: the depicted BMU would have less neighbours within the shown radius. Here, we do not want to make assumptions on grid topology, as we do not want to make assumptions on the number or nature of the clusters. Therefore, we aimed to keep the grid as square as possible, given a certain k. To that end, we created a simple function `grid.dim()`, which gives the two integers that are closest to $\sqrt{k}$ whose product is k.

2.2.2.1 Comparison of k-means and SOM Clustering

The recursive, step-wise SOM algorithm described above is strikingly similar to an online k-means clustering algorithm. Indeed, if we take the bubble function and set σ for all epochs to 0, and if we additionally define α to be equal to 1 for all epochs, then the centroid weights will move one-by-one through the feature space without any spatial correlation to each other. The here described k-means algorithm operates in batch mode, however, presenting all data to the centroids at once, for every epoch. A batch version of the SOM algorithm exists [48], also both preserves ordering and converges [49], and has even been claimed to be both faster and safer than the original procedure [44]. In this regard, this *Batch Map* could be viewed as a spatially constrained form of k-means clustering. Nonetheless, the spatial constraint bolsters the global ordering of the centroids on a two-dimensional grid, which is not possible in k-means clustering.

2.3 Clustering Stabilization

The discussed clustering algorithms have one trait in common: they converge typically to local optima [e.g. for k-means 27, 50]. Since they are usually initialized with random starting points, this in turn means that they may yield different results every time; even with the exact same settings for the exact same data. Here, two means of stabilizing clustering algorithms are considered. They both rely on iteratively running the algorithms.

2.3.1 Picking One of Iterated Runs

The standard k-means implementation in R includes a stabilization option called *nstart*. The argument can be any integer and will match the number of starting point sets the function will randomly pick when invoked. For every one of these sets of initial centroids, the selected k-means procedure is run. Subsequently, only the "best" of the *nstart* solutions is chosen and returned. Here, the "best" is measured in terms of the smallest ultimate overall within-cluster sum of squared errors (WCSS), which is the loss function that the k-means algorithm attempts to minimize.

As no such stabilization scheme existed for the SOM yet, a wrapper function `nstart.som()` was created to implement the same procedure as for k-means. The rationale was to enhance simultaneously stability of the clustering algorithms and their comparability, as this stability would be achieved in the same manner for both (by picking the solution with the smallest WCSS).

2.3.2 Centralization of Iterated Runs

It is expected that if k is chosen correctly, the centroids found by multiple, equal runs of the same clustering algorithm may still differ, but will be similar at least. That is, in such a case these iteratively found centroids will be grouped together in regions in feature space, as independent runs should give approximately the same results. Then, it may be desirable to not pick one set of ("best") centroids and discard the others, but rather to average the centroids per region. This would involve the identification of clusters of centroids: which centroids are likely to represent the same solution in separate iterations? Since centroid names are completely arbitrary, it is not

possible to classify groups of iterated centroids as such. Instead, a clustering algorithm would be required to group them, after which they could be centered.

Assuming that k is indeed defined as the true value and that iteratively found centroids are distributed normally per cluster, one could centralize these clusters of centroids by means of the Expectation-Maximization (EM) algorithm [51]. EM can be used to identify mixtures of Gaussians in a given data set. For a slightly more detailed explanation of its workings, see Appendix A. Its implementation in R is facilitated by e.g. the **mclust** software package [52, 53], which was used for this thesis. EM can cluster centroids together and return their means (centers) and variance-covariance (vcv) matrices. Consecutively, each data point could be assigned to its nearest EM-centered (EMc) centroid.

Which centroid is nearest can be measured in Euclidean distances. However, note that the EM algorithm also estimates the vcv matrix for each EMc centroid. This additional information can be used if nearness of centroids is measured in Mahalanobis distances. In fact, the Mahalanobis distance normalizes Euclidean distances with respect to covariance [54]:

$$D_M = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})},$$

where $\mathbf{x}$ is an observation, $\boldsymbol{\mu}$ a centroid mean and $\boldsymbol{\Sigma}$ the variance-covariance matrix of the centroid.

Altogether, this clustering stabilization technique would come down to EM-centered Mahalanobis clustering (EMcMc). The concept is briefly explored in practice at the end of this thesis. The method requires the iterated clusterings to be more or less stable, since otherwise clustering the centroids will not yield informative results. Hence, an accurate k has to be chosen that generates such stable results, but for other values of k, and thus for optimization of that parameter, EMcMc stabilization can not be used. How to pick an adequate value for k is the topic of the next section.

2.4 Cluster Number Validation

Arguably the most challenging issue in cluster analysis is the estimation of the "true" number of clusters $\hat{k}$ in the data. The difficulty obviously has its roots in the unsupervised nature of the task: there is no way of telling what is factually the correct class for even a part of the data. Evaluating clustering quality is not trivial and a multitude of measures have been proposed throughout the years to address the question of assessing whether one clustering can be objectively considered "better" than another [28, 55].

Such an examination requires validation measures that can value the goodness-of-fit of a clustering given a certain k. Canonical measures follow the intuitive reasoning that data points close to each other in feature space should be placed in the same cluster. They focus on concepts as cohesion / compactness (within-cluster distances) and separation / isolation (between-cluster distances), where smaller and larger values indicate a better fit, respectively. Another concept that has gained popularity in the field of cluster analysis is that of clustering stability. Surely, if for a given k the same clusters appear in different subsets of the data, this suggests that those clusters are meaningful and structurally present. Both the use of stability and distance-similarity indices for cluster number estimation are briefly explored here, as of both approaches a variant is put to the test in this work. Other validation techniques for finding $\hat{k}$ are e.g. the use of the Bayesian Information Criterion (BIC) [56] in model-based clustering through EM [57] and countless options for fuzzy clustering solutions [reviewed in 58].

2.4.1 Cohesion and Separation as Validity Criteria

Classic strategies to tackle the issue of true cluster number estimation concentrate mainly on the WCSS (sometimes called W_k, denoting the dispersion or distortion within cluster k). The overall WCSS always decreases with increasing k, because the more clusters you allow, the smaller the average distance of any data instance to its nearest centroid will be. Nonetheless, the WCSS

decrease typically flattens considerably from some k onwards. The value of k at this *elbow* is often taken to be the optimal number of clusters distinguishably present in the data. As this is a rather subjective solution, a number of methods have elaborated on this *elbow phenomenon*, formalizing the approach with some statistic.

The Gap statistic basically compares the observed elbow curve with an expected curve: a null reference [59]. The objective then is to maximize the Gap statistic, which is the difference between the observed and expected WCSS. By drawing a sequence of Monte Carlo samples from the reference distribution, a mean Gap value can be computed for each candidate k, along with a standard error s. The authors proposed to then define $\hat{k}$ as the smallest k such that $Gap(k) \geq Gap(k+1) - s_{k+1}$. This virtually implies moving sequentially from smaller to larger k, accepting the larger value in each step only if its Gap value is higher than or lies within the range of one standard error of that of the smaller k. Notice that in a sense, this appears to advocate a form of successive hypothesis testing where in each step the null hypothesis may be rejected in favour of an alternative if some test statistic exceeds a certain critical value.

The elbow curve was reported earlier under the name *distortion curve*, terming the elbow a "kink" in the curve [60]. This concept was later refined by Sugar and James to what became the Jump method [61]. Essentially, $\hat{k}$ is defined as the k that follows the sharpest "jump" in transformed distortion, where the power of the transformation depends on the dimensionality of the data. The underlying motivation is that small jumps mark within-cluster splits, whereas sharp jumps signify between-cluster partitions.

The two measures mentioned thus far focus on the distances within clusters, i.e. they attempt to find the k for which compactness of the clusters is optimized. The Calinksi-Harabasz statistic also takes the between-cluster sum of squares (BCSS) into account [62]. The averaged BCSS and WCSS are calculated for a clustering with a given k and their fraction, $CH(k)$, is computed. Maximization of this value yields $\hat{k}$. Note that, as the Gap statistic, this inherently bolsters an hypothesis testing principle, where the CH statistic has the form of an ANOVA F statistic checking for the presence of disparate clusters.

Other popular measures that include both cohesion and separation are, for instance, Dunn's Index (DI) [63] and the Davies-Bouldin Index (DBI) [64]. DI divides the smallest inter-cluster distance by the greatest intra-cluster distance. Although the former can be defined in multiple ways, the latter is set to be the greatest distance between any two members of the same cluster. This makes the method sensitive to outliers, which brought about some DI generalizations by Bezdek and Pal [65] and Pal and Biswas [66]. As the between-cluster distance should be large and the within-cluster distance small, the chosen $\hat{k}$ is a best worst-case scenario, in accordance with a *Minimax* perspective. The same accounts for DBI, where the separation is quantified between centroids in any metric and the cohesion can be expressed as any within-cluster dispersion function. Similar to CH, DI and DBI, the Silhouette Width described hereafter also incorporates the principle of cluster separation, on top of cluster cohesion.

2.4.1.1 Silhouette Width

The Silhouette Width is a distance-based measure of how well a certain datum fits within its cluster [30]. It compares cohesion with separation, i.e. how well a point suits its current cluster versus other clusters. Let $a(i)$ be the average distance of an observation $\mathbf{x}_i$ to the other data within its cluster and $b(i)$ the average distance of the datum to the data in the nearest other cluster. Then the silhouette s of that observation i is:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.$$

It can be deduced from the equation above that $-1 \leq s(i) \leq 1$. Since we desire low values for a and high values for b, greater silhouettes imply better fits. Averaged over observations per cluster, one could obtain k mean goodness-of-fit values per clustering. An overall silhouette index for complete clusterings could then be computed e.g. by either taking the average or minimum of

these k silhouettes. The former was proposed by the author in the original paper; the latter may grant comparable information, albeit from a worst-case scenario (Minimax) point of view: the cluster that fits least well would determine the goodness-of-fit of the total clustering for that k. The **cluster** package provides an excellent implementation of the Silhouette Width in R [67].

Cohesion and separation can be gauged in several ways. Gap, Jump and CH statistics use protoype-based distance-similarities: WCSS and BCSS are computed with observation-to-centroid distances. However, DI calculates cohesion from one observation to another, for instance. The Silhouette Width assesses cohesion and separation on grounds of all pairwise distances of the data themselves: a graph-based approach. For smaller data sets, this does not pose problems. However, the number of distances that have to be computed grows quadratically with sample size N of the data in a graph-based setting (namely with $\binom{N}{2}$). This will give rise to computational issues for very large N, as the working memory needed to store the resulting distance matrices will become impossibly vast. The POLDER-3 data analyzed in this study has $N = 1,131,324$ observations, which is simply too many for normal Silhouette Width computations. In the next section, a workaround is presented which we call the Gridded Silhouette Width; it allows for an approximation of the actual Silhouette Width for data sets with many observations.

2.4.1.2 Gridded Silhouette Width for Large Data Sets

Recently, adaptations of Dunn's Index and Silhouette Width for big data were reported, designated BD-Dunn and BD-Silhouette, respectively [68, 69]. These new validation indices were inspired by the original ones, but in effect brought back to a prototype-based level as to circumvent the calculation of all pairwise data distances. Basically, only two types of distances are considered: that of a data instance $\mathbf{x}_i$ to its cluster centroid $\mathbf{C}_k$ and that of a centroid to a global centroid $\mathbf{C}_0$ (i.e. the centroid of the centroids). Both measures return one value informing on the quality of the entire clustering as a whole. BD-Dunn divides the minimum $\mathbf{C}_k$-to-$\mathbf{C}_0$ distance by the maximum $\mathbf{x}_i$-to-$\mathbf{C}_k$ distance over all clusters. BD-Silhouette is computed as the fraction of the average $\mathbf{C}_k$-to-$\mathbf{C}_0$ distance (inter-cluster) minus the average $\mathbf{x}_i$-to-$\mathbf{C}_k$ distance over all clusters (intra-cluster) on the one hand, and the maximum of the two on the other hand. Although the similarity with the original methods is evident, the nature of at least the BD-Silhouette procedure is fundamentally different from its predecessor: the graph-based character is fully abolished.

The goal here was to device an adjusted Silhouette Width algorithm that is applicable to a data set such as the one at hand (large N), while maintaining the graph-based character. We reasoned that we needed to reduce the number of distances calculated, but without resorting to the use of centroids as representatives of complete clusters. In brief: we had to summarize data over more local regions within clusters, using such local summaries for the distance computations.

Hence, the first step is to divide the feature space into equally sized sub-regions or boxes, which we call grid cells. The grid cells have the same dimensionality p as the data. So, taking c cells per dimension yields c^p grid cells in total. Generally, not all grid cells will contain data and the higher the dimensionality of the data, the greater the number of empty grid cells will probably be. Empty grid cells are non-informative and therefore discarded. Subsequently, all pairwise distances between the centers of remaining the grid cells $\mathbf{g}$ are calculated. Compared to using all pairwise observation distances, the number of required computations can be severely diminished in this fashion.

Now, a Gridded Silhouette Width $gs_j(i)$ will be calculated for every grid cell center $\mathbf{g}_i$ within every cluster set S_j, where $1 \leq j \leq k$ for a clustering of data in k partitions. Note that data points within one grid cell may be assigned to different clusters. For that reason, the Gridded Silhouette is calculated per cluster S_j. Let i' index every grid cell center other than $\mathbf{g}_i$ in set S_j. Likewise, j' indexes every set S other than S_j. Distances are depicted as $d(x, y)$, giving the distance between x and y. In principle any distance metric could be used; in this work we use Euclidean distances.

The average distance of a grid cell center $\mathbf{g}_i$ to all other grid cell centers $\mathbf{g}_{i'}$ in its own cluster S_j is calculated, weighted to the number of observations of S_j in the target grid cell $n_{\mathbf{g}_{i'}}$. This

value is called $ga_j(i)$, and is the gridded equivalent of $a(i)$ in the real Silhouette Width:

$$ga_j(i) = \frac{1}{(n_j - n_{\mathbf{g}_i})} \sum_{\mathbf{g}_{i'} \neq \mathbf{g}_i \in S_j} \left(n_{\mathbf{g}_{i'}} \cdot d(\mathbf{g}_i, \mathbf{g}_{i'})\right),$$

where n stands for the number of observations of whatever it is indexed by. Observe that $n_j - n_{\mathbf{g}_i}$ here equals the sum of all $n_{\mathbf{g}_{i'}}$ for a specified $\mathbf{g}_i \in S_j$.

In a similar fashion, $gb_j(i)$ corresponds to $b(i)$ of the original Silhouette Width. It thus represents the smallest of average weighted distances of grid cell $\mathbf{g}_i \in S_j$ to the grid cells $\mathbf{g}_{i'}$ of every other cluster $S_{j'}$:

$$gb_j(i) = \min_{\substack{1 \leq j' \leq k \\ j' \neq j}} \left[\frac{1}{n_{j'}} \sum_{\mathbf{g}_{i'} \in S_{j'}} \left(n_{\mathbf{g}_{i'}} \cdot d(\mathbf{g}_i, \mathbf{g}_{i'})\right) \right].$$

Please appreciate that $\mathbf{g}_{i'}$ is defined differently in $gb_j(i)$ than in $ga_j(i)$: it marks grid cells from set $S_{j'}$ and S_j, respectively. Making use of these two quantities, we can define a Gridded Silhouette Width per grid cell per cluster:

$$gs_j(i) = \frac{gb_j(i) - ga_j(i)}{\max\{ga_j(i), gb_j(i)\}}.$$

In order to retrieve the subsequent Silhouette Width approximation per cluster, we average the Gridded Silhouette Widths over grid cells per cluster, weighted to the number of data instances assigned to that cluster in each grid cell:

$$gs_c(j) = \frac{1}{n_j} \sum_{\mathbf{g}_i \in S_j} (n_{\mathbf{g}_i} \cdot gs_j(i)),$$

of which the accuracy naturally depends on the specified number of grid cells per dimension c.

Identically to the real Silhouette Width, a measure of goodness-of-fit of an entire clustering would finally be obtained by either taking the average or minimum over the individual cluster silhouettes. The validity index has also been implemented in R in this work in the function `silh.grid()`. This new function returns an object of the class `list` with length k, of which every element is a data frame containing the $gs_j(i)$ values for one cluster S_j.

One could argue that this approach is a hybrid between a graph- and prototype-oriented method. We do not measure (dis)similarity by means of centroids, but rather through localized regions: the grid cells. In some sense, one could view the centers of these grid cells as local, within-cluster centroids. The index is then based on all weighted pairwise distances of these local "centroids". These weights are based on the data densities per cluster in the grid cells: how well such a local "centroid" represents its region (or cell) for a given cluster.

The Gridded Silhouette Width is intuitively an approximation of the actual Silhouette Width: as the grid casted over the feature space becomes infinitely fine by letting the number of cells per dimension c grow to infinity, every observation gets its own grid cell. Moreover, the grid cell centers will get closer to the observations for finer grids, making the approximation complete. However, this also points towards a limitation of this novel validity index: it works well for large N, but not so much for large dimensionality p of a data set. The reason lies in the exponential growth of the number of grid cells c with dimensionality: c^p. Although only grid cells containing data are considered for distance computations, the algorithm still has to determine for each grid cell whether it holds data or not. In other words: all grid cells nonetheless have to be analyzed in some sense, creating a *curse of dimensionality*-like issue.

As can be seen in the Results section, a simulation study offers experimental evidence that indeed the Gridded Silhouette approximates the original version. Following a clustering in k partitions, gridded and real values are calculated per cluster for a few situations, after which the Mean Squared Error (MSE) is calculated between them for a range of candidate c:

$$MSE_k(c) = \frac{1}{k}\sum_{j=1}^{k}\Big((s(j) - gs_c(j))^2\Big).$$

Plotting the MSE as a function of c could grant insight into another question that arises with Gridded Silhouette application: which value of c is adequate for a proper approximation? The answer will obviously heavily depend on the situation: cluster compactness is clearly a key player, for instance. Essentially, it poses just another parameter to optimize: enlarging c improves the approximation, but is met with computational cost. Thus the question may be reformulated to finding the smallest c that still produces adequate approximations. With real-life data, possibly a pre-train set could be used for the optimization process, using similar MSE plots for a range of potential values of c.

A possibility would be to split this pre-train set into a number of equally sized parts and to cluster them independently with the algorithm(s) of choice for the set of k that is also to be studied on the train set. For each of these clusterings, the real and Gridded Silhouette Widths can be determined per cluster. This process is then repeated for a range of c, producing an MSE plot as described above with mean MSE values and standard errors of the Silhouette Width approximation for every c and every k. One could subsequently adopt a "1-se-rule", defining $\hat{c}$ as the smallest value of c for which the MSE lies within one standard error of the c with the overall minimum MSE.

In this work however, only a proof of concept is given. In depth testing of c optimization was beyond the scope of this work and a practical solution was adopted here: the largest value of c was elected for which the MSE by eye appeared close to 0 and which the working computer these analyses were ran on could handle without crashing. This value was found to be around $c = 10$ for this data set and this computer. The pre-train set used to construct the MSE curve contained 10% of the total train set observations and it was split randomly into 5 parts, stratified by the month the data were observed in. This meant that every one of these pre-train data folds held about $16,000$ data instances; a number for which the computation of the Silhouette Width was still feasable.

2.4.2 Stability as Validity Criterion

The concept of this type of cluster validation revolves around the intuition that a good clustering is one that is reproducable on variants of the data. Such variants can include e.g. subsamples, bootstrap samples or jittered samples. The idea here is of course that the data come from an underlying k number of distributions. Clustering different samples from these distributions should generally yield the same result. Forcing an algorithm to look for too many clusters may lead to arbitrary splits that are specific for the sample, instead of the distribution. Likewise, hunting for too few would lead to arbitrary mixing of clusters. Both settings would be unstable over multiple samples from the same distributions, since different clusters would be split or merged in the different samples.

The principle of clusters having to manifest stably in subsets of the data was used by the resampling method of Levine and Domany [70]. They proposed to perform a clustering on the full data with N observations, using a set of algorithm parameters V (e.g. k in k-means). An N-by-N connectivity matrix $\mathcal{T}$ then represents whether each two data instances reside in the same cluster (1) or not (0). Next, m resamples of the data are taken and each of these subsamples is clustered in the same fashion as the complete data set it is derived from (using the same V). This in turn yields m connectivity matrices, indexed by μ. For each of these matrices $\mathcal{T}^{(\mu)}$, a comparison is made with the original, full data connectivity matrix, only considering the data points in common to both data sets. For each pair of points ij that were originally clustered together and also occur in the subsample, their co-membership in $\mathcal{T}^{(\mu)}$ is checked as well and marked either the same (1) or not (0): $\delta_{\mathcal{T}_{ij},\mathcal{T}^{(\mu)}_{ij}}$. Finally, the ultimate measure is given by averaging over these values for ij and subsequently over all m subsamples: $\mathcal{M}(V) = \langle\langle\delta_{\mathcal{T}_{ij},\mathcal{T}^{(\mu)}_{ij}}\rangle\rangle_m$. If this index lies closer to 1, it

indicates robustness of the clustering to resampling and therewith stability. Hence, the V that produces the highest figure is picked as $\hat{V}$.

A similar albeit slightly different approach was proposed in the Model Explorer Algorithm by Ben-Hur, Elisseeff, and Guyon [71]. Instead of comparing the clustering of a subsample with that of the full data set, pairs of subsamples are compared with each other. Similarity of their clustering solutions are expressed by extraction of the data instances they share and computing e.g. a Jaccard coefficient. The whole chain of operations can then be repeated for a set number of times for a range of k. The estimation of $\hat{k}$ is chosen to be the largest k for which the distribution of similarity values is concentrated close to 1, representing the boundary of a transition from stable to unstable clustering.

The Clest procedure of Dudoit and Fridlyand [72], published shortly after the Model Explorer Algorithm, also splits the original data set in two parts, but operates from a more supervised learning perspective. As in the previous method, both subsamples are clustered in an identical manner. However, one subsample is dubbed the train set and a classifier is constructed that optimally desribes the cluster labels based on the data in that subsample. The performance of this classifier in predicting the cluster labels of the second subsample, the test set, is then assessed in the form of some similarity statistic. This part of the scheme appears comparable to the use of 2-fold cross-validation in supervised learning, regarding the cluster labels of the test set to be the true labels of the response variable $\mathbf{y}$. The rest of the procedure takes on an hypothesis testing-like point of view. From a suitable null distribution, a number of simulated data sets are generated. On each of these sets, the same computations are carried out, generating similarity values for a reference situation. The median similarity statistics of both situations are compared and if the difference d_k exceeds some user-defined threshold, the k used for the clustering is favoured over the null (which states that there is not more than one cluster). The full process is repeated for a range of k, after which $\hat{k}$ is taken to be the k of the significant ones for which d_k is largest. This validation technique requires many parameters to be specified by the user on top of the typical ones, among which the type of classification algorithm to use, the similarity statistic and the reference null distribution to sample from.

Lange et al. [73, 74] start with the same outset: split the data into two parts, cluster both in the same manner, train a classifier on the one and use it to predict the cluster labels of the other. To judge the prediction quality, they wish to apply the normalized Hamming distance, which essentially returns a 1 if the labels match and a 0 otherwise and subsequently takes the average of these zero's and one's. In short, this is the misclassification rate. However, the label names themselves are arbitrary in cluster analysis: two separate clusterings of the same data with $k = 2$ may produce the exact same clustering with exact opposite labelling. That would result in a maximum misclassification error of 1, as all labels are different. Therefore, they take the error to be the minimum normalized Hamming distance for all permutations of the prediction class labels. They continue to define their stability index as the average adjusted misclassification error; a smaller value indicates a higher degree of stability of the clusterings over the two disjoint splits of the data. Lastly, they normalize their stability measure for its dependence on the k used in the clusterings. A simple example illustrates the issue: the chance of guessing the correct label for a data instance for $k = 2$ is 50%, whereas it is 1% for $k = 100$; random guessing gives lower error rates for lower k. The normalization is achieved by dividing the found stability index value by an empirically found one from randomly sampling k labels a number of times and averaging over these simulations. The index can then be calculated for repeated splits and a range of k, defining $\hat{k}$ as the one for which the validation measure is minimal.

Hennig [75, 76] noted that all these indices attempt to estimate the stability of an entire clustering, over all clusters. He argues that some individual clusters within a clustering that appears on average unstable may in fact be robust and meaningful. Hennig therefore proposed cluster-wise examination of a clustering [76]. Stability of each cluster can for example be assessed by means of a comparison with their most similar cluster resulting from a resampling technique. Following that line, he explored multiple variants using bootstrapping, subsetting and jittering

and used the Jaccard coefficient for cluster comparison [75]. This cluster-centered perspective was shared by Bertrand and Mufti [77], who compare cluster solutions of a number of subsamples. The latter method aims to decompose a clustering stability to grant insight into what is stable in a certain data partition and what is less so: separate clusters, their isolation and cohesion. In addition, the authors provide a means to retrieve p-values for each of these components in a hypothesis-like setting.

As described earlier, identical cluster analyses on the same data set generally yield different results, due to the convergence of most clustering algorithms to local optima and the use of random starting points. It has been suggested to make use of this trait to estimate $\hat{k}$ as well. Steinley [78] proposed to analyze the stability of a clustering solution over a number of initiation point sets, for a range of candidate k. He introduced a new validation index for this purpose, analogous to the aforementioned CH statistic. Subsequently, one could do the same for a simulated data set, sampled from a uniform distribution with the same limits for every variable as the original set. The optimal k is then picked to be the one for which the difference between the real and simulated index is maximum. In case the difference is less than zero for all k, this would indicate no structure: $\hat{k}$ is 1.

A more recent paper also proposed to exploit the local optima property of clustering algorithms by averaging over the repeated solutions generated by using separate sets of random starting points [79]. The author sets out to use existing validity indices, as the Silhouette Width and CH statistic, to define stability scores over these indices for individual data instances. The scores depend on the quality of the total data partitioning, expressed by a user-chosen index, and how often two observations are clustered together. Furthermore, the method applies a correction for each two observations to be co-clustered by mere chance, instead of due to underlying structure in the data set. Averaging over data points within clusters and over clusters can produce stability validation indices for clusters or the entire clustering, respectively.

Wang [80] described the distance between two clusterings on the same data as the probability that they do not put the observations in the same cluster. The two clusterings are trained on two independent samples and then both projected on a third sample, thereby generating two partitions on the same, third subset. The instability thereof is subsequently determined by averaging the between-clustering distance: the estimated probabilities that data instances in the third set are clustered together for both clusterings. As can be observed from this description, the Wang method requires three subsets: two to separately train the clusterings and a third to cluster, using the trained models. This is why the author described a cross-validation scheme, partioning the data in three random folds. In every iteration, one of the three folds serves as validation fold. The final instabilities can, for each of a range of k, then be averaged over the validation fold solutions. Clearly, the estimated instability should be minimized: $\hat{k}$ is the k for which that is true. Instead of a cross-validation set-up, the method was extended to a bootstrap setting too [81].

A theoretical examination of stability-based cluster validation led to the surprising implication that there might be situations in which such indices may yield poor results [82, 83]. Due to asymmetries in the "natural" clusters of the data, a too small k may generally merge the same clusters and a too large one may on average split the same clusters over different data variants. The clustering would be stable, even if k was chosen incorrectly. Although caution is needed when using stability indices, Von Luxburg [84] has pointed out that the posed issues are unlikely to play a major role in realistic situations, opposed to idealized ones. The author concludes that stability as validation measure breaks down when the usage of center-based clustering algorithms (like k-means) breaks down, which depends on the data and its underlying distribution. However, it has become clear that the theoretical background of stability-based cluster number validation is in need of further elaboration and elucidation.

An important validation method not discussed thus far is that of Prediction Strength [29, 85]. It is worked out to a larger extent in the next subsection, as it is one of the indices implemented, tested and used in practice here.

2.4.2.1 Prediction Strength

As Clest [72] and the method of Lange et al. [73, 74], the Prediction Strength measure follows a supervised learning-oriented concept: train clusterings on two subsets, and try to predict the clustering of the one using the other [29, 85]. Low prediction errors imply stable clustering patterns over subsamples of the data for the used input parameters. One subset is dubbed the train set $\mathbf{X}_{tr}$ and the other the validation set $\mathbf{X}_{val}$ (in the original articles the latter is called the test set; however, in this thesis we wish to reserve that name for later use).

So, for every candidate k, $\mathbf{X}_{tr}$ and $\mathbf{X}_{val}$ are clustered independently in an identical fashion. Denote these operations by $C(\mathbf{X}_{tr}, k)$ and $C(\mathbf{X}_{val}, k)$. They both partition the feature space into k distinct regions. The Prediction Strength method assesses whether any combination of two data points from the validation set that fall into the same region for $C(\mathbf{X}_{val}, k)$ also do so for $C(\mathbf{X}_{tr}, k)$. Now whether or not two observations i and i' are clustered together in the same region can be marked by a zero ("no") or a one ("yes"), as is done in Levine and Domany's resampling method [70]. Here, the n-by-n matrix containing these co-membership values of all n observations in a clustered data set is designated D. This matrix is fully symmetric and its diagonal always consists exclusively of 1's, since every observation is always in a cluster with itself.

The method measures to what extent the centroids found in the train set can predict co-memberships in the validation set. Thus, Prediction Strength is examined per cluster as generated by $C(\mathbf{X}_{val}, k)$, in a sense in consonance with the cluster-wise validation concept of Hennig [76]. Denote these clusters as A_k, indexed by j, with n_{kj} corresponding observations. For each of these clusters,

$$\sum_{i \neq i' \in A_{kj}} D[C(\mathbf{X}_{tr}, k), \mathbf{X}_{val}]_{ii'}$$

returns twice (as the matrix is symmetric) the number of validation set observation pairs that end up clustered together when these data are assigned to their closest train set centroids. Since we count the off-diagonal elements of D, the actual fraction of correctly predicted co-memberships in the validation set per cluster is given by

$$\frac{1}{n_{kj}(n_{kj}-1)} \sum_{i \neq i' \in A_{kj}} D[C(\mathbf{X}_{tr}, k), \mathbf{X}_{val}]_{ii'}.$$

The Prediction Strength is subsequently defined as the minimum of these fractions over all clusters within a clustering for a certain k:

$$ps(k) = \min_{1 \leq j \leq k} \frac{1}{n_{kj}(n_{kj}-1)} \sum_{i \neq i' \in A_{kj}} D[C(\mathbf{X}_{tr}, k), \mathbf{X}_{val}]_{ii'}.$$

Note, that this approach circumvents the issue of arbitrary cluster labels, without having the need to explore all possible label permutations as in the procedure described by Lange et al. [73, 74]. After all, Prediction Strength just considers whether pairs of data instances stably reside together in a cluster, regardless of the cluster name.

Prediction Strength is considered in a repeated v-fold cross-validation setting. That is, the data are split randomly into v equally sized folds, of which one becomes the validation set and the other $v-1$ folds collectively form the train set. The Prediction Strength is calculated using these settings and this is done in an iterative manner using every fold as validation fold once. The final value is the average over the v folds and optionally a standard error can be calculated as well. An additional option is to perform this whole procedure an M number of times, over several independent random splits, to avoid any potential structure coincidentally introduced in the data folds by the random splitting. Means and standard errors over M can be reported and used to select $\hat{k}$.

When $k = 1$, all data will always by definition be in the same cluster, resulting in a Prediction Strength of 1. So, $ps(1) = 1$. Therefore, $\hat{k}$ is not found by merely maximizing Prediction Strength, as the maximum is generally found at $k = 1$. Instead, $\hat{k}$ is rather estimated to be the largest k

for which the Prediction Strength is reasonably high. Hence, this index gives an indication of the largest number of clusters k that can be reliably predicted in the data set. Although the threshold for what is "reasonably high" is somewhat arbitrary, the authors claimed that a lower boundary of $0.8 - 0.9$ appeared to work well in practice [29, 85].

This validation index is available for use in the R package **fpc** [86]. However, the implementation is limited in a number of ways. Firstly, it works with a selection of clustering algorithms, excluding the Self-Organizing Map. Secondly, the data is always split into two folds, i.e. v is fixed at $v = 2$. Although the authors of the original papers found no evidence that larger v produce better results (and therefore stuck to $v = 2$), excluding the option altogether makes the cross-validation scheme incomplete [29, 85]. In addition, it thereby also prohibits potential further studies of the effect of different v in various situations. Lastly, the data splitting process is invariably set to be completely random. While this is in principle a desired trait, the addition of *stratified* random sampling is in practice valuable. In the POLDER-3 data case here, for instance, all folds preferably contain approximately equal numbers of observations per month. Whereas the data assignment to a fold is in fact still random (as it should be), the stratifying constraint should enhance the folds' representability of the full data set.

These limitations led to the construction of new functions in the R environment in which the mentioned parameters can be specified. We set out to create an overarching cross-validation function, `cv.clust()`, which can take in multiple clustering algorithms, cross-validation options and cluster validation measures as input. It subsequently calls to the corresponding, smaller goodness-of-fit functions, which are thus sort of building blocks that can optionally be specified or not. The Prediction Strength function constructed here, `ps()`, is one of these building blocks. So, its input arguments are restricted: only the labels assigned to the validation set by the train and validation centroids are required and only the Prediction Strength that belongs to those labelings is returned. This working plan is different from that of the **fpc** package, which performs the full cross-validation procedure in its `prediction.strength()` function as well.

Direct implementation of Prediction Strength as presented in the papers will work well for small data sets. However, for larger sets, a tremendous amount of working memory is required. For example, we found that using the approximately $800,000$ observations of the POLDER-3 data would have required over 500 Gb of RAM. The underlying reason can be traced back to two computationally intensive processes:

(1) Calculation of the distance matrices of all validation set data to the train centroids;
(2) Calculation of the co-membership matrices D, making all pairwise observation label comparisons.

The first issue is brought about by the computation of the full distance matrix of all observations to all centroids in order to assign each observation to the centroid for which this distance is smallest. This matrix is overly large for validation set clusters with many observations. Therefore, our `ps()` function refrains from the full matrix computation, but rather performs this labelling step in an observation-wise fashion: for each datum, all distances are determined and only the cluster label for which it is smallest is stored, discarding the distances before moving to the next observation. Naturally, this approach is more time-consuming and thus represents a remedy in the form of a RAM-computation time trade-off.

The problem of the second point in question lies in the fact that the D-matrices become quadratically larger with the number of observations n_{kj} in a given validation set cluster A_{kj}. For instance, in a cluster with $200,000$ data points, which is a very sensible scenario considering the over $800,000$ data instances in our POLDER-3 train set, the total number of just *unique* pairwise label comparisons would be $\binom{200,000}{2} = 19,999,900,000$. Clearly, this is too much to expect from any computer deemed "normal". Now the fundamental idea underpinning the solution to this issue is that all these separate pairwise comparisons – and indeed the sheer computation of D – are redudant. Factually, we are merely interested in the number of 1's in just one triangle of D; either the upper or lower one. To this end, we only need to calculate the number of possible pairwise combinations per unique label within every validation cluster A_{kj}. Recall that these within-validation set cluster labels were assigned by the closest train centroid for each validation

set observation. Taking this consideration into account, the Prediction Strength equation can be re-written to:

$$ps(k) = \min_{1 \leq j \leq k} \frac{2}{n_{kj}(n_{kj}-1)} \sum_{i \in A_{kj}} \binom{n_i}{2},$$

where n_i is the number of observations that have the i^{th} label within validation cluster A_{kj}. The sum is muliplied by 2, since it only yields the total number of 1's in one triangle of D, whereas the denominator $n_{kj}(n_{kj}-1)$ gives the combined number of elements in D of both triangles (minus the diagonal).

Concealed in the **fpc** Prediction Strength function code, we found that a similar solution had been applied. Essentially, the `prediction.strength()` function uses the following formula:

$$ps(k) = \min_{1 \leq j \leq k} \frac{1}{n_{kj}(n_{kj}-1)} \sum_{i \in A_{kj}} n_i^2 - n_i,$$

which boils down to the same solution, as $2\binom{n}{2} = n^2 - n$.

2.4.3 A Cross-Validation Framework for Optimization of k

Abridged, the main statistical challenge comes down to optimization of k, the number of clusters to look for in the data set. This can be viewed as a model selection issue: we wish to estimate the $\hat{k}$ that represents the optimal balance between model complexity and data fitting. Clearly, the model fit would be optimal for $k = N$, as the WCSS would be minimal: 0. This is obviously not what we are after; we wish to fit a more parsimonious model that captures the general trends in the data. For too small k, the model is not complex enough and probably misses patterns in the data by merging clusters that should be separated. In contrast, too large k will cause the algorithm to fit the data too tightly, undesirably splitting clusters. The resulting trade-off is weighed by the validation indices reported above, each in their own described way.

In a supervised learning context, the optimal model is commonly determined by means of empirical risk minimization, choosing the parameter value that minimizes the expected prediction error of the model. However, steps should be taken to ensure the results generalize to another, independent data set. This is done in order to prevent *overfitting*: fitting a model that describes the data it is trained on too well, modelling all its peculiarities that are specific for that set thus rendering it non-generic. Such a model is said to have high variance, since the solutions yielded by an overfitting model on different data sets will vary largely. The opposite is called *underfitting*: an overly simple model will miss the important trends and has high bias. This *bias-variance trade-off* is a well-studied phenomenon, as it is generally impossible to minimize both sources of prediction error simultaneously. Arguably the most elegant and widely used tool to find an optimal compromise, is cross-validation.

The cross-validation scheme is thus meant to find the model (parameter value) that optimizes predictive competence. Yet, prediction is not always the objective in unsupervised learning. Often, methods in this area are employed with the sole purpose of data exploration: no inference, or extension to other, independent data sets is desired in such cases. Bear in mind, however, that in this work we actually do aim to find patterns in the data that generalize to potential other sets. The clusters sought after in the POLDER-3 data are hoped to represent (mixtures of) aerosol types that are common and reside in the atmosphere in general; not just in the year 2006. Therefore, the k we are looking for should be optimized for the generic case, similar to the supervised learning scenario sketched above. To that end, we explored the usage of cross-validation methods for the unsupervised learning case. The goal was to create a framework, both in theory and in practice through code construction, which allows the user to optimize a generic parameter k using cross-validation.

The canonical strategy is to first split the data set $\mathbf{X}$ in a train set $\mathbf{X}_{tr}$ and a test set $\mathbf{X}_{te}$. The test set is put aside and will only be used at the very end, when the model parameter is

optimized. In this way, the data used for parameter optimization, i.e. the train set, are not used to assess the quality of the final model: it provides a way of examining the performance of this final model on an independent data set. Here, we reserved 30% of the data as $\mathbf{X}_{te}$. The data split is performed in a random fashion, but stratified on the months the observations were measured. This generates a train and test set in which the $30:70$ ratio is present per month, to preserve the extent to which the two sets are representative of the full data set. Subsequently, k was optimized on $\mathbf{X}_{tr}$ by means of v-fold cross-validation.

Usually, this approach is called k-fold cross-validation. Obviously, using the symbol k here would cause confusion, as it is already used to indicate the number of clusters. This is why in the field of unsupervised learning the method is oftenwise referred to as v-fold cross-validation. As already briefly explained in the Prediction Strength section, the idea is to split $\mathbf{X}_{tr}$ in v folds with approximately equal numbers of observations. Again, these fold assignments of data are carried out in a random manner, in our application stratifying on month. Now one of the folds becomes the validation fold and the other $v-1$ folds together make up the train fold. A model is trained on the train fold and its predictive capacity is evaluated on the validation fold in the form of a prediction error. This process is repeated v times with the exact same model parameters, each iteration taking another fold as validation fold. The cross-validation estimation of the prediction error for those parameters is then defined to be the average found prediction error over the v iterations. Repeating the whole set of operations for a range of values for a model parameter can then grant insight into what could be a proper value: the one is picked for which the thus computed expected prediction error is minimal.

The question as to what an adequate value for v is, has been subject to discussion in the literature. In the special case where $v = N$, the models are being trained on all but one observation of the train data. This procedure is named leave-one-out cross-validation (LOOCV). It is computationally intensive, as it requires N iterations. Large values of v will reduce the bias of the model: since the train set is bigger, the model will be more likely to fit to the regularities and patterns present in the data. However, this reduction in bias is met by an increase in variance: the more tightly the model fits to the train data, the higher the degree of overfitting. This additional bias-variance trade-off thus leads to a compromise for the choice of v. Typical settings are $v = 5$ or $v = 10$, to balance between higher bias (lower v) and higher variance (higher v), besides taking into account the computational cost for very high v as in LOOCV [87, 88, 89, 90].

Cross-validation techniques have been applied in several stability-based cluster validation indices for unsupervised learning methods, as described earlier in this thesis. Both Dudoit and Fridlyand [72] and Lange et al. [73, 74] essentially apply repeated 2-fold cross-validation and Wang [80] reported an adapted form of 3-fold cross-validation for his approach. Prediction Strength can in principle be performed with v-fold cross-validation [29, 85]. Hence, in a sense these indices attempt to find a k that is generic, uncovering clusters that have meaning beyond the given data set. This supervised learning-like set-up could in theory be extended to validation measures other than those judging stability. The objective then becomes to estimate $\hat{k}$ to cluster in such a fashion that it is meaningful for other, new data as well.

The model selection criterion in our cross-validation is not prediction error, as in the supervised case. Instead, the selection part is handled by the validation indices, e.g. Prediction Strength selects on the basis of stability over the cross-validation folds. Not everyone might always wish to assess cluster quality by stability in every case, however. Moreover, it might be desirable to examine the same clusterings using multiple validation measures, using different criteria. This was a major motivation to also implement a distance-based index, in addition to the stability-based Prediction Strength. The Silhouette Width is a popular validation tool [55, 91] and by implementing it, we could combine it with the development of our new extension of it to large data sets: the Gridded Silhouette Width. In the cross-validation framework, the thus acquired Validated Silhouette variants assess how well a trained model can partition another data set into compact and separable clusters: it is computed on the clustering created by assigning validation fold data to train fold centroids. Taking the average or minimum of these cluster-wise Validated Silhouette Widths gives our validation measures.

For this purpose, we constructed the `cv.clust()` function in R (schematically depicted in Figure 9). It splits data randomly into a specified v number of folds, with the option of stratification by a given variable in the data. For a defined M number of times, a v-fold cross-validation procedure is carried out. Every time, the train fold is clustered by means of a provided algorithm, on top of the validation fold for stability-based validation measures. The method should in principle work with any prototype-based clustering algorithm, possibly with only minor adaptations to the code. The validation fold data are then assigned the label of their closest train centroid in Euclidean distance. Finally, the desired validation measures are computed using the labels of the validation fold data supplied by the train centroids and, in case of a stability-based validation index, those given by the validation centroids too. Averages and standard errors can be calculated over either the v folds (if $M = 1$), or the global cross-validation repeats (if $M > 1$). Setting M to a larger value than 1 could serve to reduce the influence of any structure coincidentally introduced in the folds by random sampling, since the fold assignments are repeated over the cross-validation operations in that case. This is therefore presumably useful for smaller data sets. When this cross-validation framework is employed for k optimization, it can simply be repeated for a range of k. This yields validation measure values with standard errors for all given values of k, of which a plot can be generated that will allow for generic $\hat{k}$ estimation.

Validation measures that are included in the current version of `cv.clust()` are the Prediction Strength (PS), the Average Validated Silhouette (AVS), the Minimum Validated Silhouette (MVS), the Average Validated Gridded Silhouette (AVGS) and the Minimum Validated Gridded Silhouette (MVGS). In case the Gridded Silhouette is applied, the number of grid cells per dimension parameter c needs to be provided as well. Any other argument for clustering algorithm functions will be passed to them by `cv.clust()`, e.g. k, but also choice of neighbourhood function for a Self-Organizing Map.

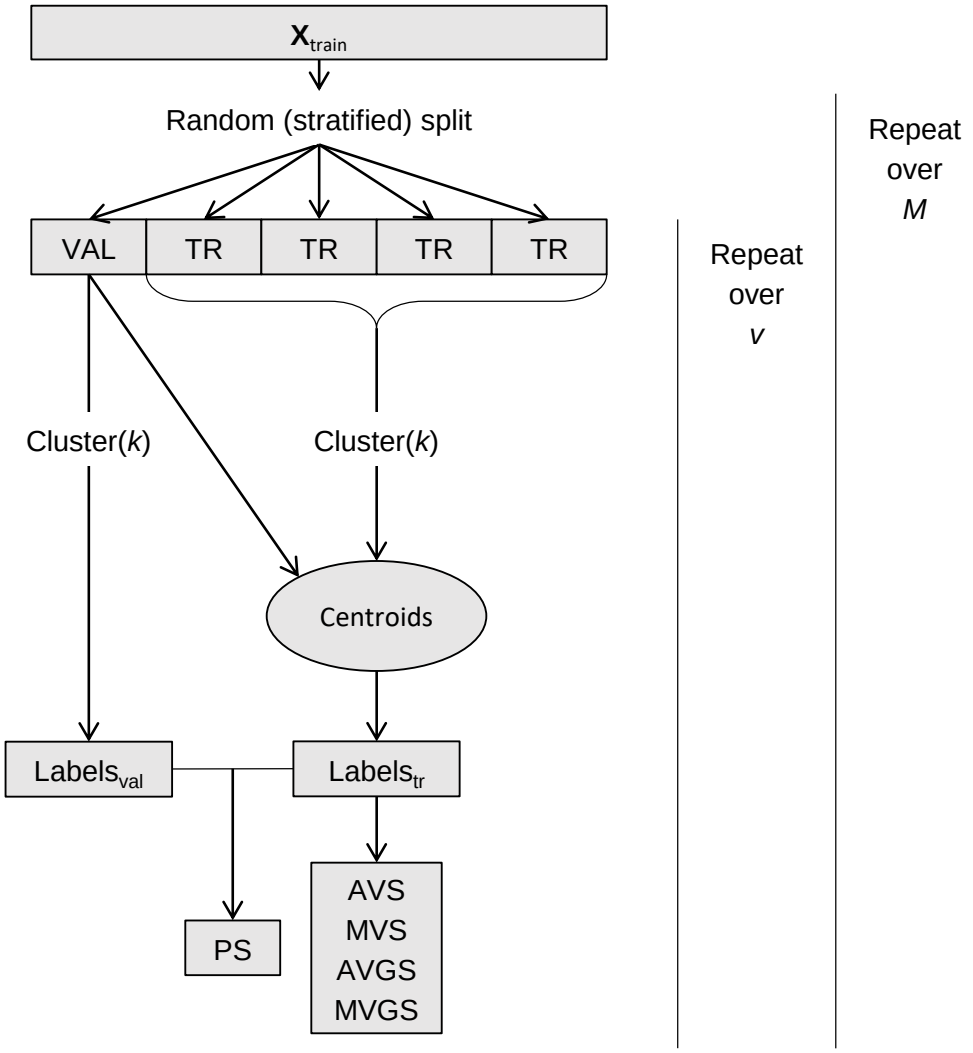


Figure 9: **Schematic illustration of the `cv.clust()` procedure.** The train set $\mathbf{X}_{train}$ is randomly, possibly stratified, split into v folds, here 5. Each of these folds is in one of v iterations the validation fold VAL, the others form together the train fold TR. The train fold data is clustered by a specified prototype-based algorithm and the validation fold data are assigned to their nearest train fold centroid, producing the train labels $Labels_{tr}$ of the validation fold data. Using these labels, Average or Minimum Validated (Gridded) Silhouettes (A/MV(G)S) are computed. In case of Prediction Strength (PS) calculations, the validation fold data are clustered as well, generating the validation fold labels $Labels_{val}$. The two sets of validation fold data labels are subsequently used to compute PS. For a set number of clusters k, the means and standard errors can be calculated over either the v folds (if $M = 1$), or over the total cross-validation repeats M (if $M > 1$). Applying this scheme over a range of candidate k returns the requested validation indices, which can be plotted as a function of k and thereby aid in its optimization.

3 Practical Application and Results

The aforementioned preceding exploratory study of Visser [23] was taken as a starting point for this work. Hence, an attempt to reproduce that research and its results was made at first. This raised a number of questions concerning the methodology, which led to the development of the methods described in the previous section. These methods were practically examined by means of several simulation studies, followed by application to the real-life POLDER-3 data set. The set-up of this section is accordingly, presenting the outcomes of those analyses.

3.1 Reproduction of Previous Work

Visser [23] trained a SOM on half of the POLDER-3 data set: the odd months (January, March, May, July, September, November). Next, the data of the full year were mapped to their nearest final neuron of the model, thus labelling all the data available. The motivation for this approach was to reduce computation time by training on only half of the data. For sake of simplicity and as a proof of concept, only three variables were used in the clustering process, namely SSA, AE and Sphericity. The SOM was set to find $k = 9$ clusters, in line with the assumption that there would not be more than 9 'natural' clusters present and the rationale that similar clusters could always be merged in hindsight. The detected clusters were characterized and, indeed, partly merged, generating 6 final clusters. Labels (aerosol types) were subsequently assigned to the clusters matching these characterizations, after which the labelled data were projected on a world map, per season.

For this reproduction effort, the SOM input parameters were kept the same as in the previous study as well as possible: an initial neighbourhood radius σ_0 of 3, initial learning rate α_0 of 0.1, a decay speed for both of $\lambda = 150$, a number of epochs T of 300 and a square grid of 3 by 3 centroids. Nonetheless, it was not possible to keep all parameters equal. The study of Visser [23] was carried out using the Python software package PyBrain [92], which has different options than **kohonen**. Exponential decay functions are not available in the software used in this project and thus linear decays were applied over T for both σ and α. Moreover, it remains unclear which neighbourhood function was used by Visser [23]. Personal correspondence with SRON, the site of that project, revealed that neither of the neighbourhood functions implemented in **kohonen** (Gaussian and bubble) were employed, but presumably a triangle function instead. Hence, in this comparitive reproduction effort both the Gaussian and bubble neighbourhood functions were applied separately, as it was the best we could do given the software at hand.

Visser [23] documented only one SOM run with the specified parameters. However, a SOM converges to a local optimum and the outcome might thus be unstable. As we studied its reproducability, we ran each analysis repeatedly: 500 and 1000 times for the bubble and Gaussian neighbourhood function settings, respectively. The final centroids of both inquiries are depicted in Figure 10, with the ones of Visser superimposed. The figure is set in feature space, where the dimension of the third variable is colour-coded. Thus, every point in the plots portrays one of nine centroids generated by one of the SOMs; i.e. the data is not shown here. As can be seen, the bubble function returnes strikingly stable results, since the centroids are consistently found in virtually the same regions in feature space. Application of a Gaussian function appears less stable, although groups of centroids can be spotted here as well over iterated runs. There are similarities between centroids found here and those of Visser; however, the abberations are more noticable. Since the same data was used in both studies, this might be due to the differences in algorithm parameter settings. Indeed, the results in Figure 10 indicate that these settings may matter, as the differences between the plots can be attributed to the use of distinct neighbourhood functions only.

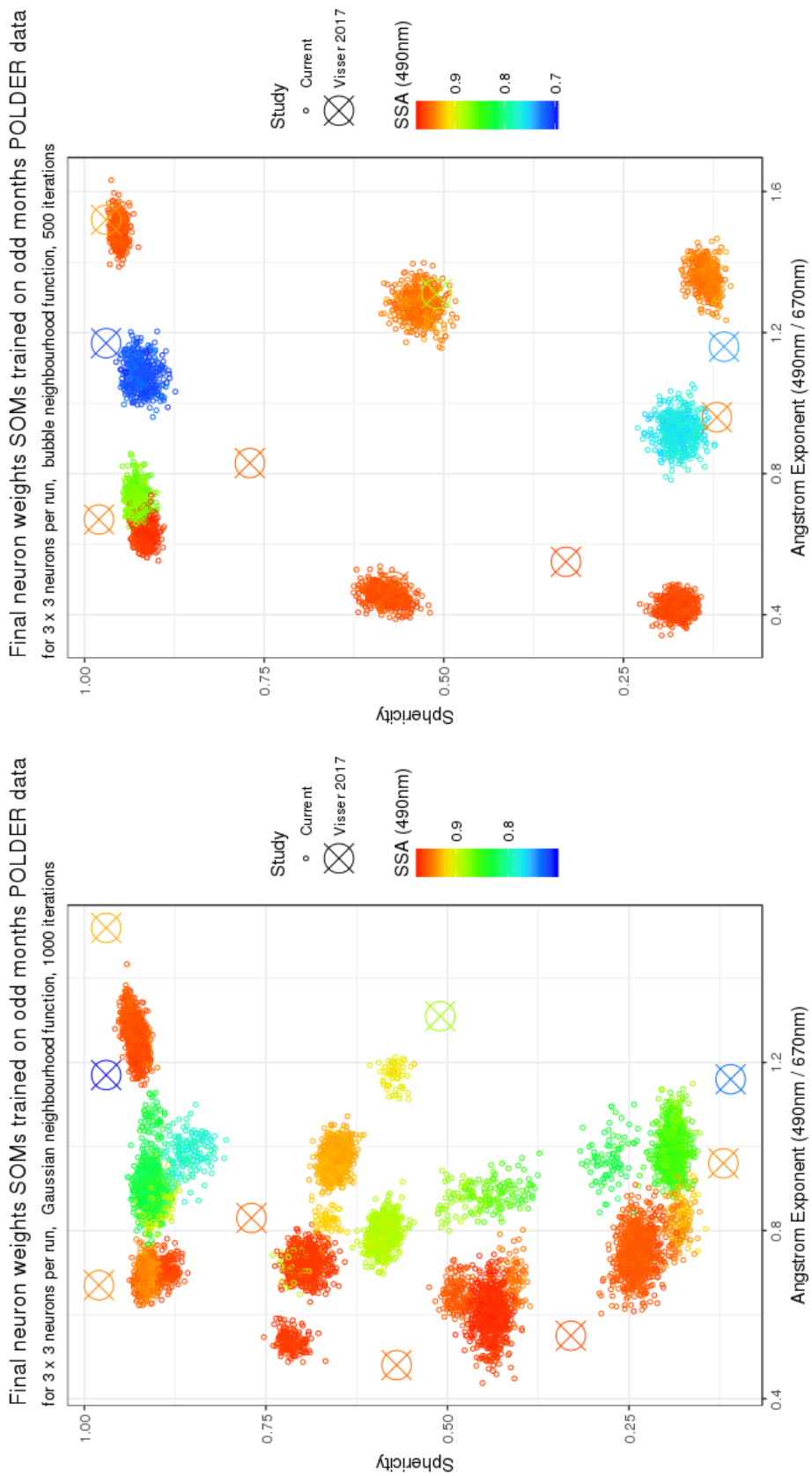


Figure 10: **Found centroids, or neurons / weight vectors, of Self-Organizing Map iterations on the odd months POLDER-3 data for the two available neighbourhood functions, depicted in feature space.** For all clusterings, a 3x3 rectangular grid was used. Centroids as found by Visser [23] are superimposed for comparison. Third dimension shown by colour code. Left panel: Gaussian neighbourhood function, clustering performed 1000 times. Right panel: bubble neighbourhood function, clustering performed 500 times. SOM: Self-Organizing Map; SSA: Single Scattering Albedo.

Both the methods recorded by Visser [23] and the results obtained here gave rise to various questions regarding the clustering methods and the possible interpretation of the outcomes. We wished to validate a set k beforehand, making a conscious choice for the number of clusters to look for in the data. Moreover, we aimed to perform this optimization on a separate part of the data to prevent overfitting and allow for a prediction-orientated study: we wanted the found clusters to have meaning beyond the given data. Data splitting would not be in odd and even months, but with a stratified random sampling scheme to avoid the accidential introduction of structure in the train, validation and test sets. Individual clustering runs would be stabilized to counter the local optima issue and lastly, a comparison was to be made with the much simpler k-means clustering algorithm to assess any added value of the SOM itself.

The methods proposed in this work are meant to meet these issues in the pursuit of erecting a more standardized framework to approach this clustering problem, improving on the existing work. Their performance is presented in the followig sections.

3.2 Simulation Studies to Test Developed Methods

The simulation studies reported here serve a two-fold purpose: (1) demonstrating that the constructed R codes and functions for the methods work and (2) testing whether the theoretically described methods actually work in practice, i.e. can they be used to optimize k for data clustering?

3.2.1 Developed Prediction Strength Function Works

The `ps()` function constructed for this work was invoked on clusterings of three simulated data sets to calculate the Prediction Strength at different values of k. A 5-fold cross-validation scheme was used, which in turn was repeated 5 times as a whole; hence v and M were both set to 5. Means and standard errors were computed over the 5 M-repeats for clusterings performed by both the k-means and the SOM algorithm for k ranging from 1 to 10. Default options were used for the SOM (e.g. bubble function). The three simulation scenarios encompassed data simulated from either 2, 3 or 4 two-dimensional Gaussian distributions, always drawing 1000 random observations from each distribution. Data were scaled using feature scaling.

The computed Prediction Strengths per simulation, algorithm and k are shown in Figure 11. Recall that with Prediction Strength, one looks for the largest k that lies above some threshold taken in the range of $0.8 - 0.9$. The Prediction Strength for $k = 1$ is by definition 1. For the case of these well-separated clusters, the estimated $\hat{k}$ are correct in all cases, implying that the created R function works properly.

3.2.2 SOM Stabilization by *nstart* Wrapper Works

SOMs converge to local optima and thus yield different results over runs with equal settings. In order to stabilize independent SOM clustering runs, a wrapper function `nstart.som()` was produced with the same strategy as the *nstart* argument of `kmeans()`: run the algorithm *nstart* times and return the clustering with the smallest overall WCSS. The *nstart* option was separately set to 1 and 10 for both the k-means and SOM clusterings in Figure 11. The wrapper function works well and the figure demonstrates that for the actual correct k, $\hat{k}$ is mostly estimated properly with higher certainty for the stabilized runs. This is not always the case for incorrect k, however.

This result seems reasonable, since the wrapper only helps in finding the best solution in the data of one fold, given a certain k. If this k is correct, the locations of centroids in feature space found in the different folds will be more similar to each other for stabilized runs, provided the folds are representative of the full data. This in turn means that the produced Prediction Strength will be higher, since more similar centroids yields more similar data labels. If this k is incorrect, however, the solutions within folds may be optimized to some extent, but the locations of the centroids will still differ over the folds due to the arbitrary splitting or joining of clusters. This latter process is more dependent on the random splitting of data into folds: due to mere

chance a cluster in some fold may be a bit more elongated than in another fold, for instance. However, the centroids found in the different folds will still be disparate and not necessarily less so for stabilized within-fold clusterings. The Prediction Strength found for an incorrect k will thus not necessarily be affected by finding the "best" solution within a fold. In brief: the best solution for a fold will still be wrong for incorrect k and will still differ from the best solution of another fold.

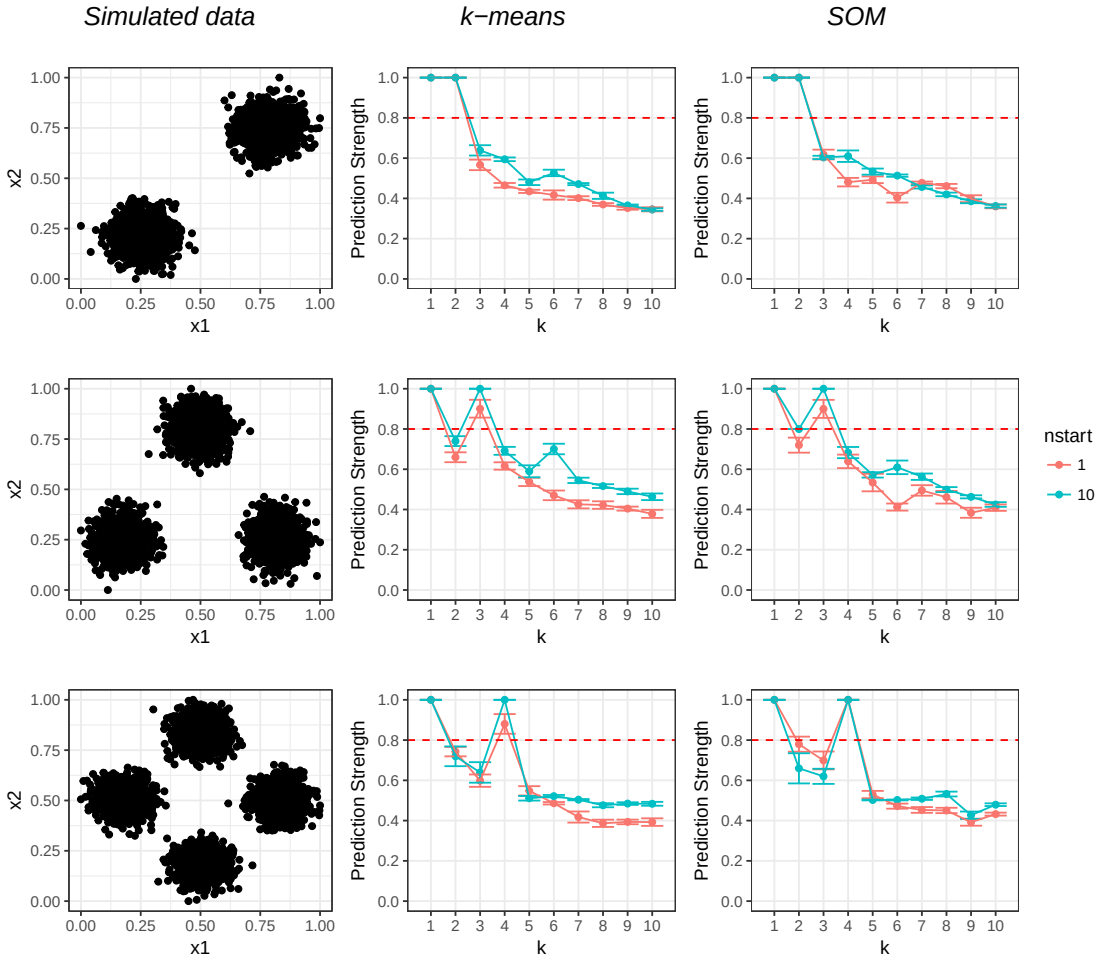


Figure 11: **Prediction Strength for three simulated data sets clustered by both k-means and Self-Organizing Maps with different *nstart* options for a range of *k*.** Simulated sets and corresponding results are depicted per row. Left column shows data sets, middle column outcomes of k-means clustering, right column outcomes of SOM clustering. A 5-times repeated 5-fold cross-validation scheme was used. Error bars are standard errors and points averages, computed over the $M = 5$ overall cross-validation repeats. Data sets were simulated from two-dimensional Gaussian distributions, with $n = 1000$ observations per distribution, over which feature scaling was applied. SOM: Self-Organizing Map; k: number of clusters looked for by the algorithm.

3.2.3 Gridded Silhouette Width Works and Approximates Real Silhouette Width

The Gridded Silhouette Width validation index was tested on three simulated data sets. Random sets of 1000 observations were drawn from 2, 3 and 4 Gaussian distributions in four dimensions, respectively. These data sets are depicted in the left panels of Figure 12, where the third and fourth dimension of the data are colour-coded in the fillings and edges of the points. The data were clustered with the k-means algorithm, using k settings of $2-10$. For each of these clusterings, both the actual Silhouette Width and the Gridded Silhouette Width were computed per found cluster. The Gridded Silhouettes were calculated for a range of grid cells per dimension c, namely $2-50$. The Mean Squared Errors (MSE) of the actual and gridded indices over the clusters per clustering are shown in the right panels of Figure 12.

The overall trend in the plots demonstrate that the Gridded Silhouette indeed appears to approximate its original cousin: the MSE effectively goes down to 0 as c increases. Thus, the `silh.grid()` function constructed here seems to work as it should. Furthermore, the grid cells per dimension required to achieve proper approximation rise with increasing k clusters looked for in the data. This observation makes sense in the light that clusters need to exist in more than one grid cell to yield a sensible result: otherwise the ga_j values of the method will become non-existent (`NA`), as there are no other grid cells in that cluster to compute distances to. As k increases, more clusters will be split and the resulting clusters will thus be smaller. Consequently, the risk of a cluster falling into one grid cell exclusively is greater for larger k. The inverse is true for c: smaller values thereof mean larger grid cells and therewith a greater risk too. Therefore, it is indeed important to choose a large enough c.

Lastly, Figure 12 shows that the MSE convergence is not entirely smooth over the course of increasing c. This may be explained by the fact that the clusters in these simulated examples are well isolated and relatively tightly bound together: for some c, the grid cell net casted over the feature space may again just separate most of the clusters each in their own grid cell. For one c smaller or larger however, the boundaries between the grid cells for that net may just fall right on the found clusters, thereby ensuring they exist in more than one grid cell. The MSE values of some c may therefore be based on the (Gridded) Silhouette Widths of less clusters, because any non-existing (`NA`) values will be omitted. This effect of grid cell-cluster alignment may cause the convergence curve to decline staggeringly for well-separable clusters, especially in combination with large k (as this implies smaller clusters). For this same reason, some of the curves start out at very low MSE values. The most extreme cases are seen for the two-Gaussian simulation: all cluster-wise Gridded Silhouette Widths for $c = 2$ had to be omitted, except for k settings of 2 and 3.

Taken together, it appears that at least three factors decrease the accuracy of the Gridded Silhouette Width approximation: large k, small c and small standard deviation of the underlying Gaussian distribution. These issues can all be resolved if c is taken to be large enough, since this takes on the latent problem of perfect grid cell-cluster alignment. Naturally, larger c is opposed by computational cost. Hence, an optimal c has to be chosen per situation, in which MSE curves might play a useful advisory role.

3.2.4 Cross-Validation Framework and Validation Measures Work

Data sets were simulated as in the previous section to assess the workings and performance of the cross-validation scheme in combination with the cluster number validation measures as described in the Theory and Methodology section. All three simulation settings were generated four times, setting the standard deviation (SD) of the simulation distributions to either 0.5, 1, 2 or 5, followed by feature scaling. The `cv.clust()` framework was tested on each of these 3x4 data sets, using both k-means and SOM clustering for a k ranging from 2 to 10. The cross-validation was not repeated (i.e. $M = 1$) and the fold number was set to $v = 5$; means and standard errors of each validation index were thus computed over the v solutions produced by the validation folds. For the Gridded Silhouette Width calculations, the number of grid cells per dimension c were set to

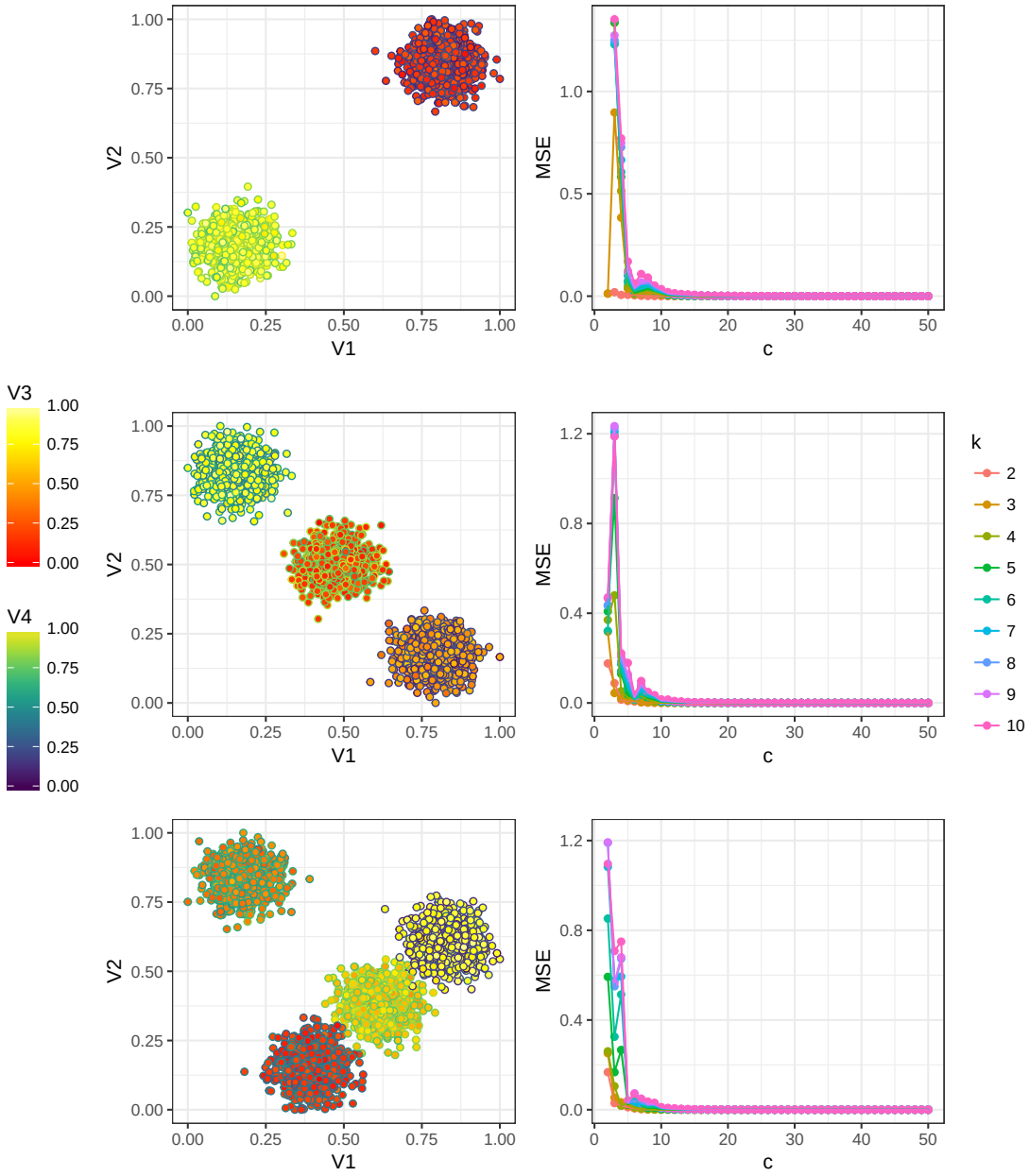


Figure 12: **Silhouette Width approximation accuracy by Gridded Silhouette Width for three simulated data sets for a range of *c* and *k*.** Simulated sets and results are shown per row; left column depicts the sets in feature space, right column the approximation accuracy. The approximation error is calculated as the Mean Squared Error (MSE) between the actual Silhouette Width and Gridded Silhouette Width per found cluster. Clustering is performed by the k-means algorithm for a set of k numbers of clusters to look for. Data were simulated from Gaussian distributions in four dimensions, with $n = 1000$ observations per cluster, scaled by feature scaling. Last two dimensions are shown by colour coding of the data fillings and edges, respectively. The error converges towards 0 for larger numbers of grid cells per dimension c.

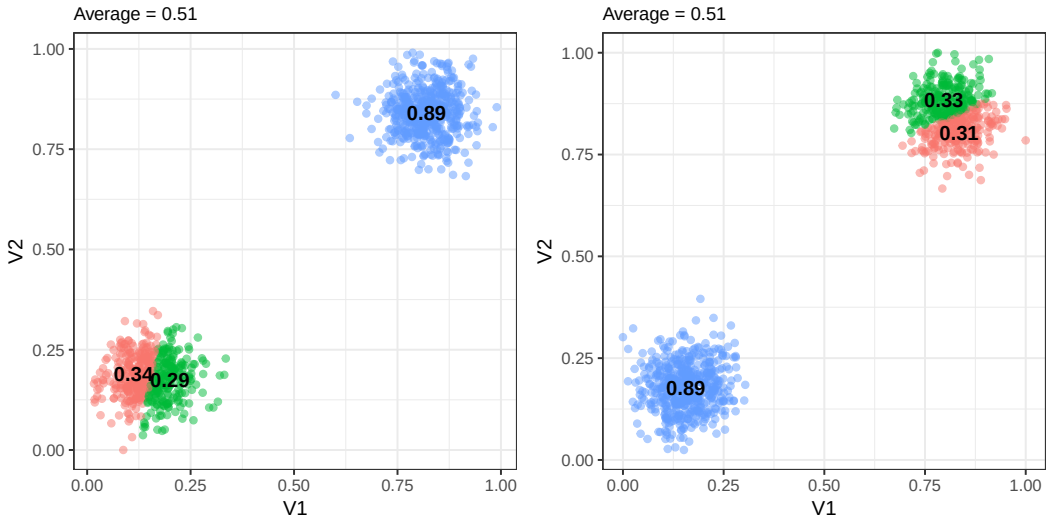


Figure 13: **Illustrative example of variation in Silhouette Widths over cross-validation folds.** Averages and minima over clusterings do not really differ between two folds the data is randomly split in, even though the solutions do. Based on simulated data from two Gaussians in two dimensions, clustered by k-means clustering with $k = 3$ after feature scaling. Number of observations $n = 1000$ per simulation distribution. Colours indicate clusters identified by k-means. Numbers in the plot are average Silhouette Widths per cluster.

20: reasonably high for an adequate approximation of the Silhouette Width, in accordance with Figure 12. Since sample sizes in these simulation studies were not inordinately big (maximum N is 4000), the actual Average and Minimum Validated Silhouettes (AVS & MVS) could be calculated, on top of their gridded alternatives (AVGS & MVGS) and the Prediction Strength (PS). Note that in real-life situations, the Gridded Silhouette Widths become obsolete when the original indices can be used.

The results of the 5-fold cross-validated cluster number validation measures for the two-, three- and four-Gaussian cluster simulation sets are presented in Figures 14, 15 and 16, respectively. As can be expected, the increase in overlap of the true clusters (due to larger SD) makes it more difficult for all indices to estimate $\hat{k}$ correctly. This difficulty seems to grow along with true cluster number: for two distributions, all measures estimate $\hat{k}$ accurately (Figure 14); for three distributions the MVS and MVGS just break down for the largest SD (Figure 15) and for four distributions, none of the indices produce the right $\hat{k}$ for an SD of 5 and the MVS and MVGS also fail for an SD of 2 (Figure 16).

In all three figures, it can again be observed that the Validated Gridded Silhouettes behave analagous to their original versions. An additional peculiarity of all Validated Silhouette measures, is that in all figures the standard errors seem to be missing. In fact, they are not missing, but exceptionally small. This implies that, over the folds, the minimum and average Silhouette Widths of the clusterings remain similar, even for incorrect k. This does not have to come as a surprise, considering that the main difference between the fold clusterings in such a case lies in the cluster locations, not the distances. In other words: strikingly similar Silhouette Widths are found, albeit in different parts of the feature space. This concept is illustrated in a simplified way in Figure 13.

All in all, the constructed `cv.clust()` function in combination with these clustering algorithms and the described cluster validation measures can, at least in some situations, be employed to estimate $\hat{k}$. The minimum Silhouette variants yield more extreme and outspoken estimations, as due to their nature, they are more sensitive to clusterings of which just one cluster is not very compact and separable.

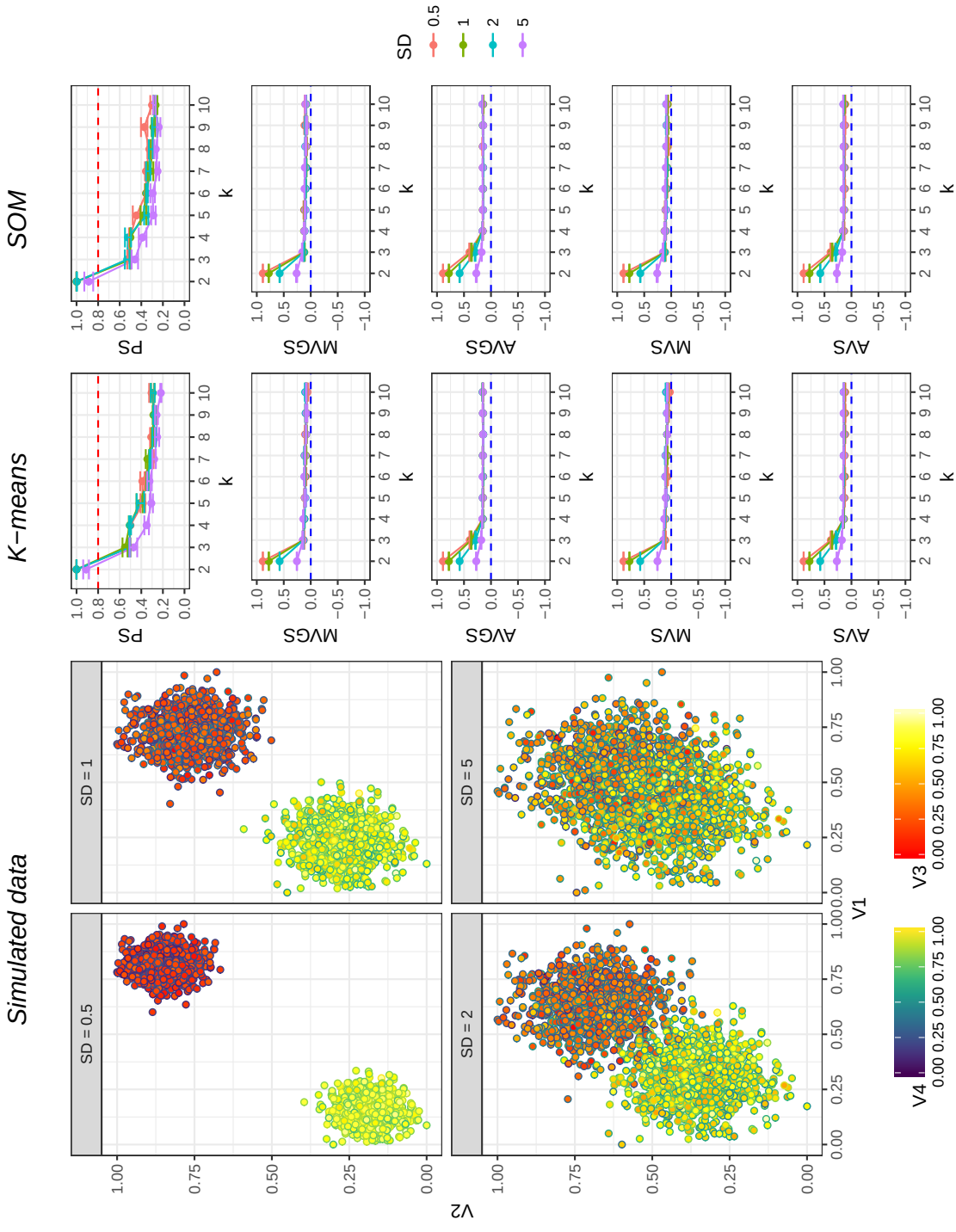


Figure 14: **Performance of validation measures in cross-validation framework on 4-dimensional 2-cluster Gaussian simulated data.** Standard deviations (SD) of sampling distributions varied and feature scaling was applied. Data sets are shown left, colour codes on fillings and edges depict the last two dimensions. Following k-means or SOM clustering, all measures were computed for a range of k centroids using 5-fold cross-validation. Means and standard errors over folds are depicted. SOM: Self-Organizing Map; PS: Prediction Strength; M/AV(G)S: Minimum/Average Validated (Gridded) Silhouette; $n = 1000$ observations per cluster.

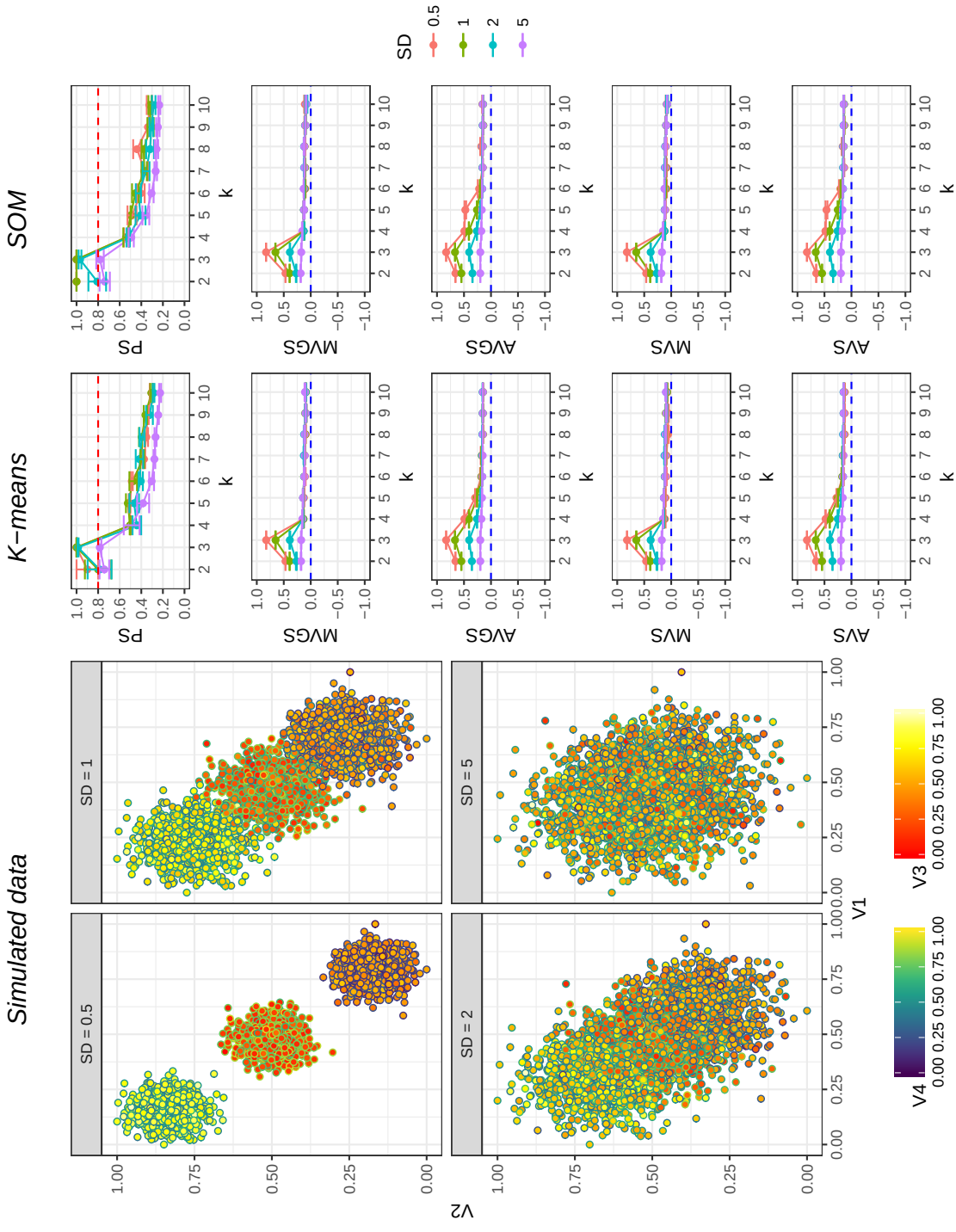


Figure 15: **Performance of validation measures in cross-validation framework on 4-dimensional 3-cluster Gaussian simulated data.** Standard deviations (SD) of sampling distributions varied and feature scaling was applied. Data sets are shown left, colour codes on fillings and edges depict the last two dimensions. Following k-means or SOM clustering, all measures were computed for a range of k centroids using 5-fold cross-validation. Means and standard errors over folds are depicted. SOM: Self-Organizing Map; PS: Prediction Strength; M/AV(G)S: Minimum/Average Validated (Gridded) Silhouette; $n = 1000$ observations per cluster.

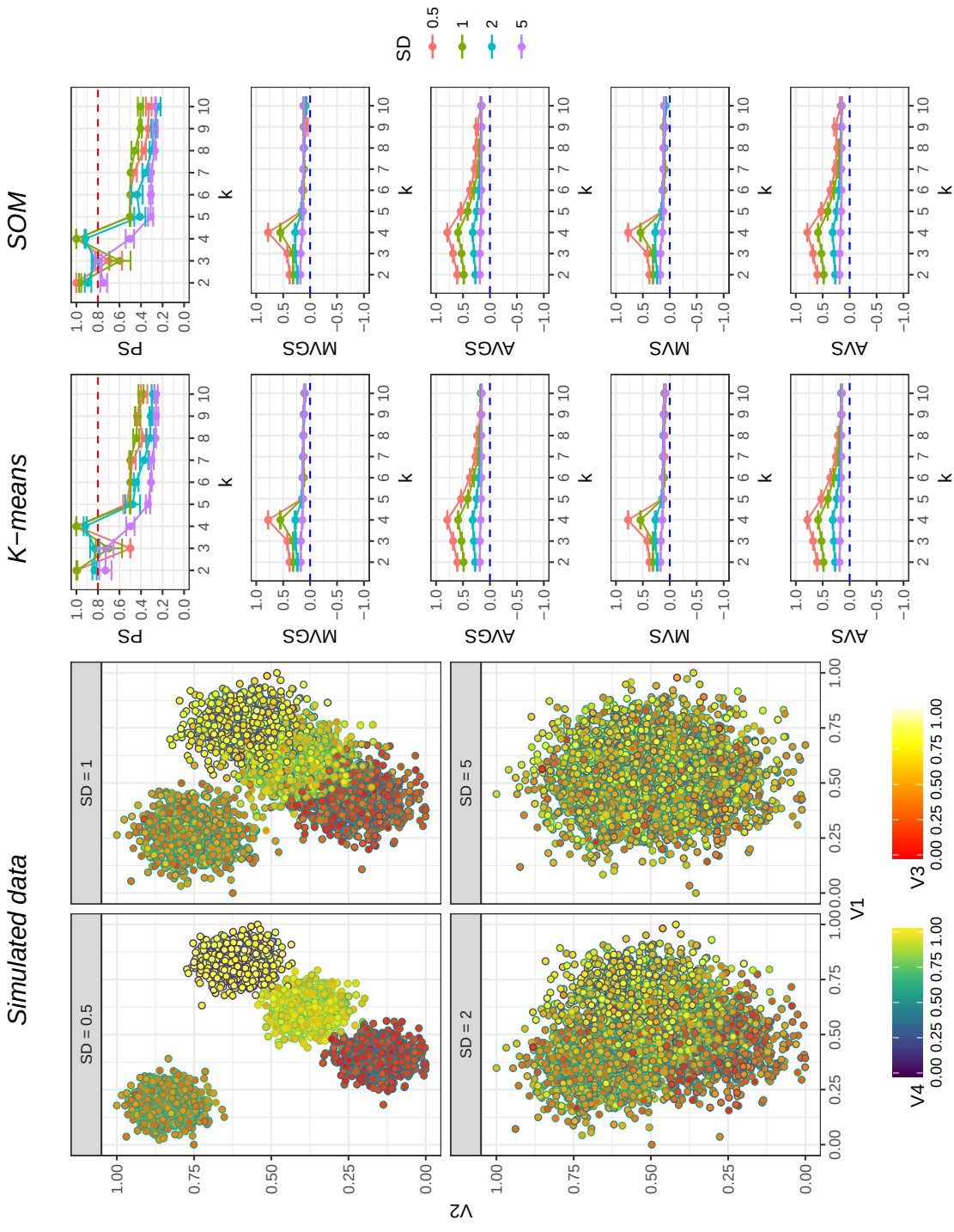


Figure 16: **Performance of validation measures in cross-validation framework on 4-dimensional 4-cluster Gaussian simulated data.** Standard deviations (SD) of sampling distributions varied and feature scaling was applied. Data sets are shown left, colour codes on fillings and edges depict the last two dimensions. Following k-means or SOM clustering, all measures were computed for a range of k centroids using 5-fold cross-validation. Means and standard errors over folds are depicted. SOM: Self-Organizing Map; PS: Prediction Strength; M/AV(G)S: Minimum/Average Validated (Gridded) Silhouette; $n = 1000$ observations per cluster.

3.3 POLDER-3 Train Set: Optimization of k

The train set contained 70% of the total data, which equals approximately 800,000 observations. This data split was achieved randomly, stratified on month of observation. The train set was used to independently find an optimal value of k clusters to estimate in the data.

3.3.1 Find Adequate c for Gridded Silhouette Width

MSE curves were constructed for five pre-train sets of the POLDER-3 data, each containing 2% ($n \approx 16,000$) of the train set data, in the same manner as the ones depicted in Figure 12. The data was sampled in a random fashion, stratified on the month of observation and each set was clustered with the *nstart* option set to 10 for both k-means and SOM clustering. The objective was to get an indication of which value of c for the Gridded Silhouette would yield adequate approximations of the actual Silhouette Widths. Averages and standard errors over the five pre-train set solutions are shown in Figure 17.

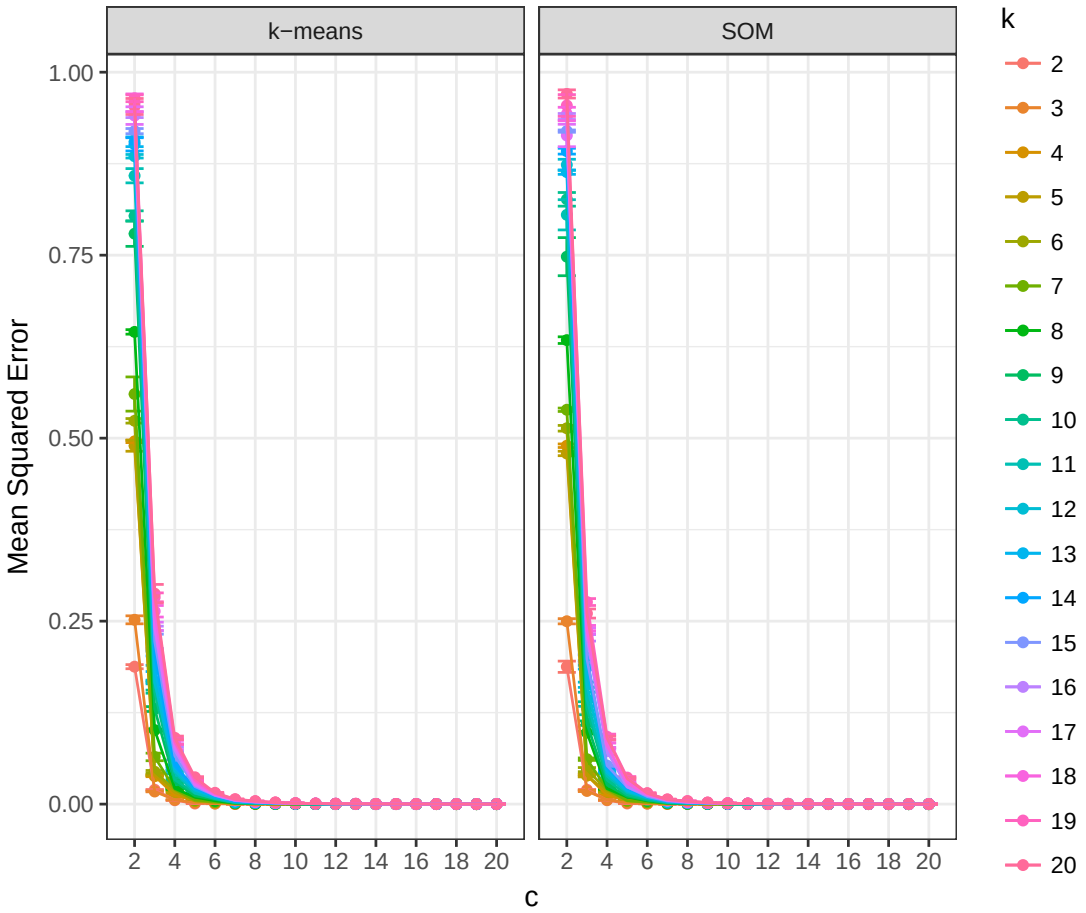


Figure 17: **Approximation accuracy of Silhouette Width by Gridded Silhouette Width on 5 pre-train sets of POLDER-3 data.** Validation indices for a range of c were computed per cluster found for a range of k using either k-means or SOM clustering. Averages and standard errors of the cluster-wise Mean Squared Errors between the measures over the 5 data subsets are depicted. SOM: Self-Organizing Map; k: number of clusters the algorithm searched for; c: number of grid cells per dimension for Gridded Silhouette Width; $n \approx 16,000$ per pre-train set.

The plots suggest that for both the k-means and the SOM clustering of these data, a c of about 10 appears to be sufficient. The MSE curve seems to have reached an average approximation error of virtually 0 at that c, with only very small improvement after.

3.3.2 Optimization of k with Cross-Validated Indices

Due to the large number of observations in the POLDER-3 train set, only the PS, MVGS and AVGS validation measures could be employed. The number of folds v for cross-validation of these measures was set to 5 and the full procedure, including fold splits, was repeated $M = 5$ times. The assignment of data to the folds was again done at random, stratified on the month of observation. For the Gridded Silhouette Width calculations, the number of grid cells per dimension c was set to 10, as explained in the preceding subsection.

Taking a cut-off threshold of 0.8 for the Prediction Strength measure as before, it advises for both of the clustering algorithms a $\hat{k}$ of 8 (Figure 18). The Gridded Silhouette Width indices however show a surprising pattern with all optima at $k = 2$. The minimum variant produces this answer more decisively as well, as seen before, demonstrating a bigger gap between its optimum and the other options. As in the simulated examples above, the standard errors for these validation measures are again negligible.

The results presented here suggest that although $k = 8$ clusters can reliably be found in the train data, the average pairwise distances of observations to others within their own cluster are for this k only marginally larger than those to data in the nearest other cluster. This balance between within- and between-cluster pairwise distances is considerably better for $k = 2$. Nevertheless, if more clusters can stably be identified, this data clustering has our preference, despite these distances being more equal between and within clusters. We expect that the reproducable clusters may still represent different (mixtures of) aerosol types in the end, making the distinction between the clusters meaningful for the objective of the clustering. Even if some of these clusters may in terms of Silhouette Widths be close to each other, separating them might still be informative, as long as this can be reproduced and is not just an artefact. Since the Prediction Strength results show that this latter risk is unlikely to be the case, we continued to cluster the test set with k set to 8. This decision was futhermore supported by reasoning that it is highly unlikely that there are only 2 well-distinguishable aerosol mixture types in the Earth's atmosphere.

3.4 POLDER-3 Test Set: Clustering and Aerosol Typing

The test set comprised the remaining 30% of the data, so $n \approx 340,000$. It was used to assess the final clustering model performance and to interpret the found clusters. As for the train set (Figure 18), the cross-validated indices were also computed for the test set using the optimized $\hat{k} = 8$ and otherwise equal settings. The found values were indeed comparable, suggesting no problems regarding the generalizability of the train set estimated $\hat{k}$ (Table 1).

Table 1: **Cluster validation measures on POLDER-3 test set, clustered with optimized *k*.** Rounded averages and standard errors, retrieved by the cross-validation framework with the same settings as for the train set (M and v set to 5). Number of clusters k set to 8, as optimized on the train set. SOM: Self-Organizing Map; PS: Prediction Strength; M/AVGS: Average/Minimum Validated Gridded Silhouette.

	PS	MVGS	AVGS
k-means	0.84 +/- 0.03	0.12 +/- 0.00	0.20 +/- 0.00
SOM	0.77 +/- 0.01	0.12 +/- 0.00	0.20 +/- 0.00

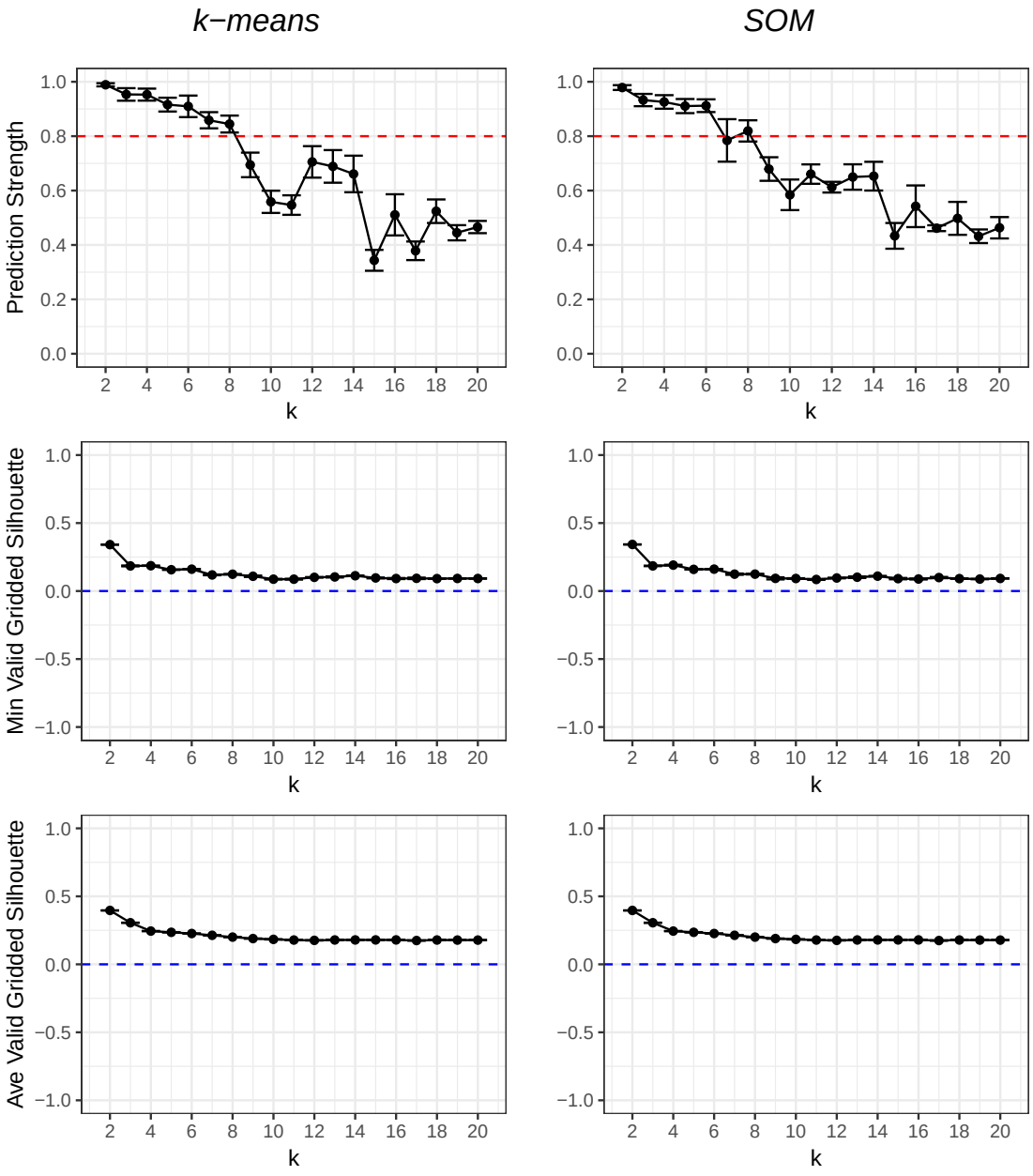


Figure 18: **Optimization of k on POLDER-3 train set using the cross-validation framework.** Clusterings by k-means (left column) and SOM (right column) assessed for range of k centroids. Means and standard errors over $M = 5$ repeats of 5-fold cross-validation are depicted. SOM: Self-Organizing Map; Min: Minimum; Ave: Average; Valid: Validated; $n \approx 800,000$ observations in train set.

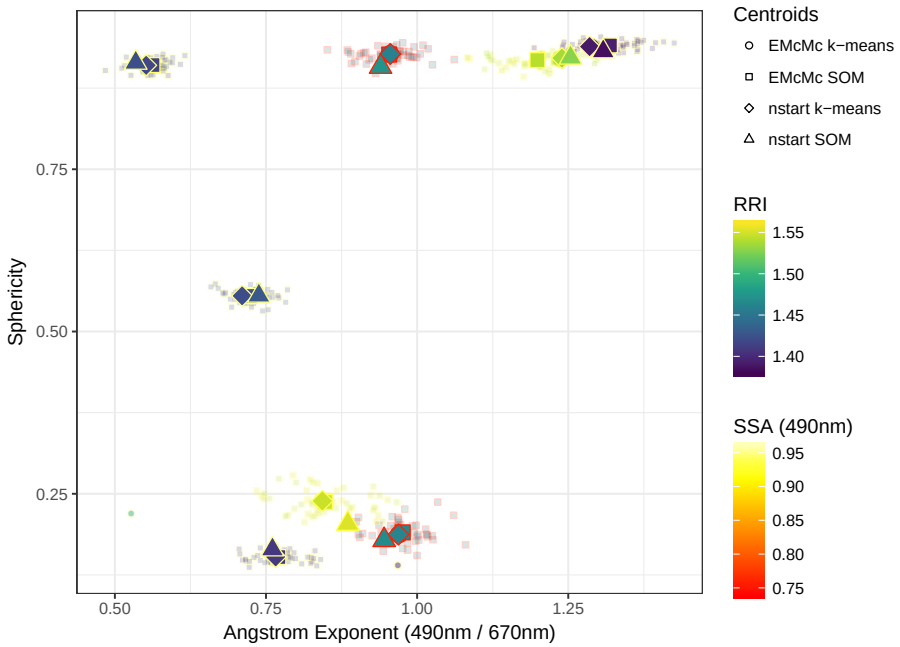


Figure 19: **Final centroids in feature space found by all four clustering algorithm - stabilization method combinations in the POLDER-3 test set.** Two of the four dimensions are depicted with colour codes on the fillings and edges of the shown points. Clustering was performed on (feature) scaled data, centroid coordinates were transformed back to original scale prior to plotting. More transparent, smaller points represent (50) iteratively found centroids by the EMcMc methods. Large, solid points represent final returned centroids. EMcMc: Expectation-Maximization-centered Mahalanobis clustering; SOM: Self-Organizing Map; RRI: Real Refractive Index; SSA: Single Scattering Albedo; $n \approx 340,000$ observations in the test set.

3.4.1 Comparison of Clustering Solutions

The test set was clustered into $k = 8$ partitions using both clustering algorithms and for both stabilization methods: with *nstart* set to 10 and by means of centering iteratively found centroids with EMc(Mc). In the latter approach, each clustering run was repeated 50 times.

The thus found centroids are plotted in Figure 19. The separate iterations of the EMcMc procedure are depicted semi-transparent and the symbols of the final centroids are for all methods depicted with larger symbols. Two of the four dimensions are colour-coded, allowing the 4D feature space to be depicted in a 2D plot. As can be observed, all clustering algorithm - stabilization method combinations generate highly comparable final centroids: they are always found in the same regions. The EMcMc iterations for the SOM are spread more or less symmetrically, whereas those of the k-means runs reside more stable in almost the exact same place with very few exceptions, as the one centroid found in the lower left of the feature space. These few exeptions do however influence the vcv matrix estimated by the EM algorithm. Since subsequent data-cluster assignment is based on Mahalanobis distances, this can actually affect the final labels appointed to the data. Indeed, especially the low sphericity data are split a bit differently for this method, compared to the others (not shown). The data that would otherwise reside in the two clusters with slightly higher average Angstrom Exponent were for a major part drawn into the cluster represented by the centroids in the lower left region of Figure 19, presumably because of this reason. The more smooth SOM clustering produced more evenly spread out centroids over the EMc(Mc) iterations and resembles the partitions made by the *nstart* approaches. Notwithstanding, it appears that clusterings of the test data into $k = 8$ parts can yield stable clusters.

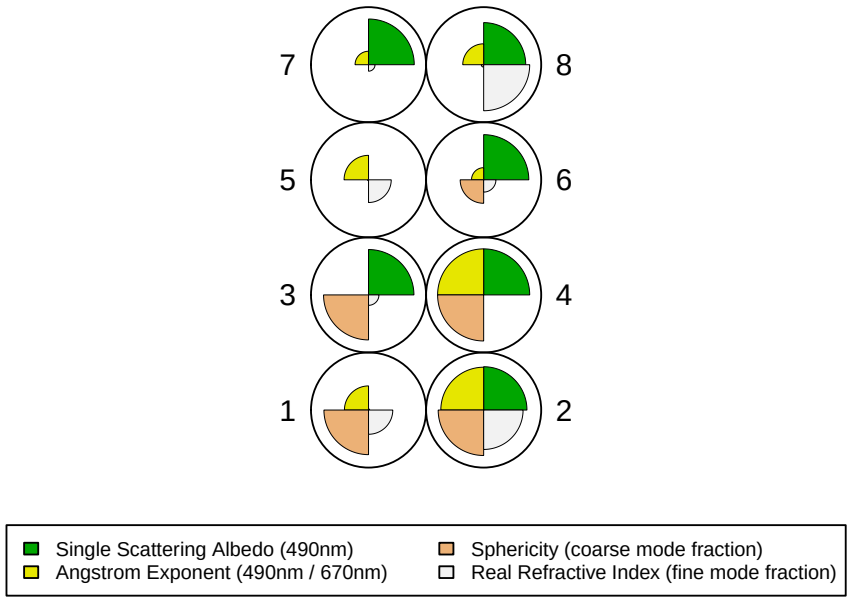


Figure 20: **Grid plot of Self-Organizing Map that clustered the POLDER-3 test set.** Clustering was stabilized by setting *nstart* to 10. Per found cluster, indicated by the cluster numbers next to each centroid, the weight vectors are depicted for each variable. Cluster numbers correspond to the ones in Table 2, other figures and text.

3.4.2 Cluster Characterization

Since the four combinations of methods shown in Figure 19 produce comparable ultimate centroids, here we will only interpret and characterize the clusters of one of them for the sake of clarity and brievety. The *nstart* stabilization method is simpler and more established and the Self-Organizing Map allows for outcome illustration by means of what we call here a grid plot. Therefore, all following results are derived from that clustering.

The grid plot is shown in Figure 20. It demonstrates the final feature space weights of all centroids on the rectangular 2-by-4 grid. For larger grids with more centroids, it will be easier to see that those close on the grid resemble each other more closely in their weight vectors. Some trends can be spotted here too: the centroids in the lower part of the grid have higher sphericities and higher values of Angstrom Exponent can be found in the lower right part. Two clusters have low Single Scattering Albedo; these clusters thus contain absorbing particles, presumably smoke-like aerosols. However, for proper characterization, it is more useful to also plot the labelled test set data on world maps. In that way, their geographical occurance can also be taken into account for the aerosol typing process. In Table 2 the proposed aerosol types are listed per cluster, based on the characterization and discussion described below.

One of the mentioned absorbing clusters can be found at the very lower left of the grid (Figure 20). The particles in this cluster 1 are furthermore spherical, which would be in agreement with biomass burning-originating smoke [19]. This matches the geographical occurence of that cluster well (Figure 21). That is, wildfires typically ravage parts of Mexico and California (spring and

summer), the Iberian peninsula in Southern Europe (summer), parts of Central Asia (summer), Australia (notorious bushfires in autumn and winter), the Sahel region and Sudanian Savanna in Africa (autumn and winter) and Southern Africa (late autumn and winter) [33, 34]. Note that these seasons mark those of the nothern hemisphere, i.e. these fires happen in the warmest time of year for each region.

Table 2: **Manually estimated dominant aerosol types per cluster.** Cluster numbers coincide with those in other figures and text. n: number of observations.

Cluster	Proposed aerosol type(s)	n
1	Smoke	25,328
2	Mixed Smoke	31,212
3	Marine	76,391
4	Urban-Industrial	56,527
5	Dusty Smoke	15,105
6	Marine Dust	63,879
7	Dust	52,711
8	Poluted Dust	18,249

However, this cluster 1 is not spotted over other areas where wildfires are known to occur routinely. These zones are however covered by cluster 2 (Figure 22) and include central South America and Middle Africa [33, 34]. Additionally, atmospheric air columns over Siberia are assigned to this cluster during summer. This might be explained by reported wildfires in the boreal forests, or taiga, in summer [33, 34, 93]. The higher SSA values might be explained by the difference in biomass burning smoke type: dark or white smoke, as distinguished by e.g. Russell et al. [19]. Another explanation can be offered by this cluster representing a mixture of more than one dominant aerosol type, like inorganic, industrial aerosols. These are generally finer and less absorbing, possibly clarifying the shifts of SSA (lower) and AE (higher), compared with cluster 1 (Figure 20). However, the regions mentioned are not known for their density in population and none of these options elucidate the high RRI, still.

Strikingly, the centroid directly adjacent to number 2 on the grid, namely cluster 4, does appear to embody industrial polution as dominant aerosol sort, causing centroid 2 to lie more or less in between smoke and urban-industrial representing centroids (Figure 20). The particles in there are generally highly spherical, non-absorbing, fine and lower in RRI (Figure 20), which all corresponds to the urban-industrial aerosol kind as described by Russell et al. [19]. Furthermore, this cluster is found over densily populated areas such as India, Bangladesh, Western Africa, East China and Indonesia (Figure 24). Taking the microphysical profile and geographical locations into account, the urban-industrial aerosol type for developing economies of Russell et al. [19] appears to be a good match with the cluster identified here. This could be further supported by the observed remarkable accumulation of data assigned to this cluster just off-coast Peru, particularly in spring. Elevated atmospheric sulfur levels have been reported, assumed to originate from industrial areas in northern Chile [94]. Although the absence of this cluster over India in the summer may seem extraordinary, this can easily be clarified by the fact that there was almost no complete data available for this location during this season (see also e.g. Figure 3).

The cluster mostly spotted over the oceans is number 3 (Figure 23). Indeed, its microphysical characterization matches that of the marine aerosol cluster of Russell et al. [19]: highly non-absorbing, coarse, spherical and a relatively low RRI, suggesting high water content (Figure 20). The dominant aerosol type in this cluster is thus expected to be sea salt.

As mentioned, the clusters in the upper part of the grid have lower sphericity values (Figure 20); this indicates the presence of dust, which is highly non-spherical. Moreover, dust particles are coarse and non-absorbing [19]. This corresponds to cluster 7, which is indeed mostly located over large deserts (Figure 27): the Sahara Desert, Arabian Desert, Thar Desert, Taklamakan Desert

and Gobi Desert are all covered during multiple seasons, as well as a region in between Australia's Great Victoria, Simpson and Strzelecki Deserts. These are all known major dust sources [95]. In addition, transport of desert dust from the Sahara Desert over the Atlantic Ocean to the Caribbean and Central and Northern America can be seen during spring and, especially, summer [95, 96]. In winter, it can be seen that the Saharan dust spreads a little more south-west and away from the Mediterranean Sea, in line with previous studies, although the reported transport over the Gulf of Guinea and the ocean towards South America cannot be observed [95, 97, 98, 99].

The remaining three clusters 5, 6 and 8 are assumed to be mixtures of multiple dominant aerosol types: they have lower sphericity values, pointing to dust in the cluster, but the rest of the microphysical properties of the clusters do not match this one type (Figure 20). Cluster 6 is likely to comprise dust particles, as its contents are non-absorbing and coarse and it is located over deserts, like cluster 7. The main difference with cluster 7 is the higher sphericity in this cluster. This thus implies a mixture of dust with another type of aerosol. The most apt candidate is the marine type: cluster 6 is also found over oceans and marine aerosols are spherical, on top of non-absorbing, coarse and relatively low in RRI (Figure 26). Some transport of dust over the Atlantic Ocean, mixed with marine aerosols, may be seen: during autumn and winter southward to Brazil, in spring to the Carribean and in summer even a little more northward to the northern USA east-coast. During summer a cloud lies over the Gulf of Guinea as well, however, which is not in line with current dust transport models; rather with biomass burning ones [95, 100].

Probably the most noticable characteristic of cluster 5 is its low SSA: the particles are predominantly absorbing in nature (Figure 20). Accordingly, its geographical spread overlaps largely with that of cluster 1, which we assigned smoke aerosols from biomass burning (Figure 25). On the other hand, the current cluster also contains non-spherical matter, suggesting we are dealing with a mixture of aerosols including dust too. The presence of the cluster above the Atacama Desert and arguably the northern Patagonian Desert, the Australian Deserts and Namib Desert in autumn and winter and over the North American Deserts, as the Mojave and Sonoran Desert in spring and summer, support the inclusion of dust. Note that this cluster occurs in multiple locations that border between wildfire areas (as described earlier) and deserts: the Sahel region and Sudanian Savanna, Southern Africa, Australia, Central Asia and the Middle East and California (Figure 25).

Finally, cluster 8 holds a major dust part as well, as its sphericity is very low (Figure 20). It appears clearly over the isolated Taklamakan Desert in Central Asia during winter, spring and summer (Figure 28) and to a lesser extent above other dust sources, including inland Australia, the Sahara Desert, Arabian Desert and Thar Desert in summer [95]. In terms of optical parameters, it mainly differs from cluster 7 due to its high RRI, implying less water content. Moreover, a glance at the histograms in Figure 28 reveals slightly higher sphericity than one would expect for dust and a bimodal distribution for AE. This in turn implies that the cluster may consist mainly of dry dust, but in addition to another, fine aerosol type. Admitting it is difficult to identify this potentially second large component of the mixture, the most likely type is probably of the urban-industrial kind: these particles are commonly fine, spherical and non-absorbing [19]. What is more, accumulations of cluster 8 appear to take place off-shore India in autumn and along Indonesia in winter, where they might represent mixtures of polution from those countries and dust from the Arabian or Thar Desert and the Australian Deserts, respectively. This is, however, speculative.

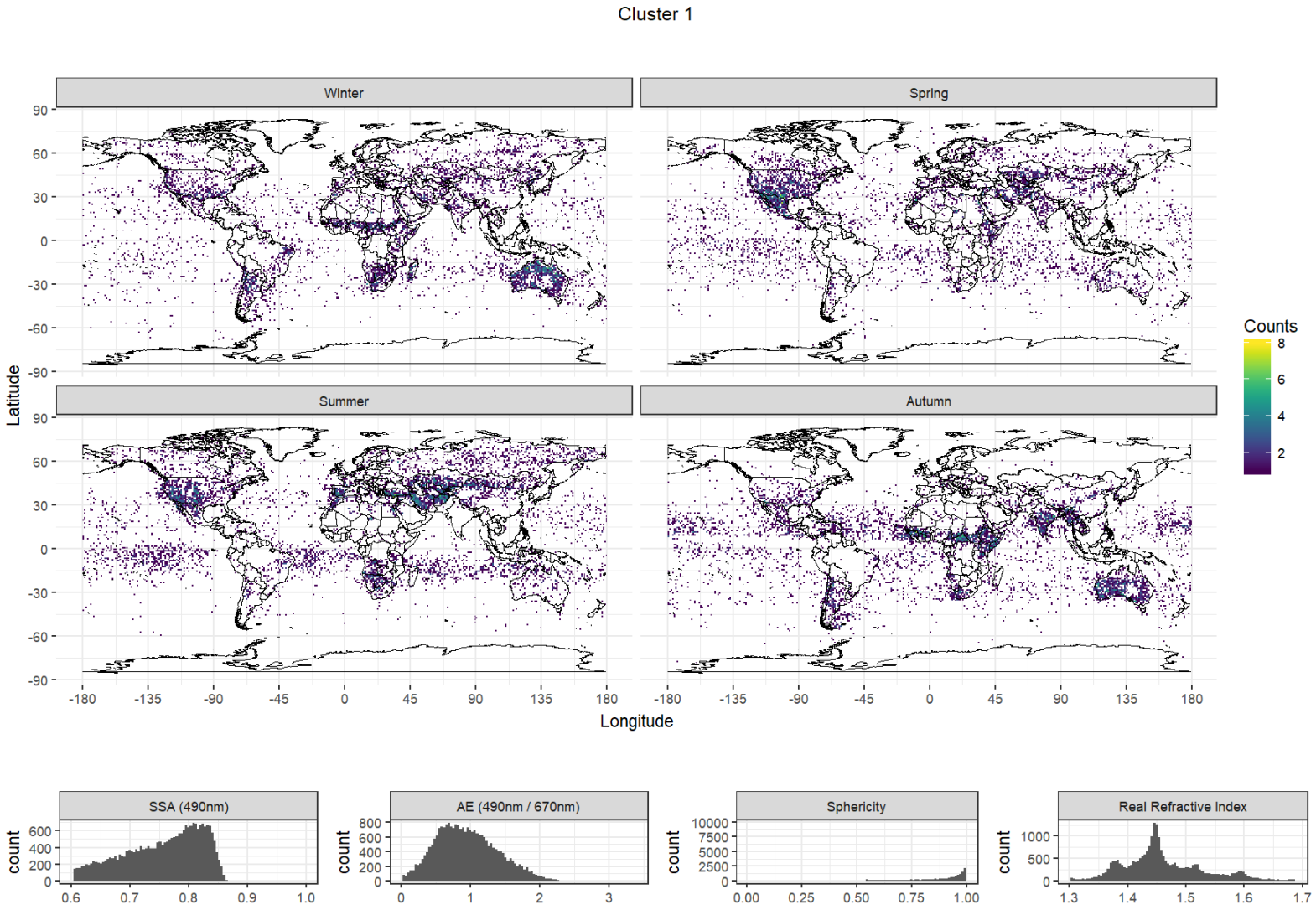


Figure 21: **Seasonal (northern hemisphere) geographical prevalence of and value distributions per variable within cluster 1.** Colour indicates the number of times a grid cell was assigned to this cluster within that season. Clustering results of the Self-Organizing Map with *nstart* = 10 on the POLDER-3 test set. SSA: Single Scattering Albedo; AE: Angstrom Exponent.

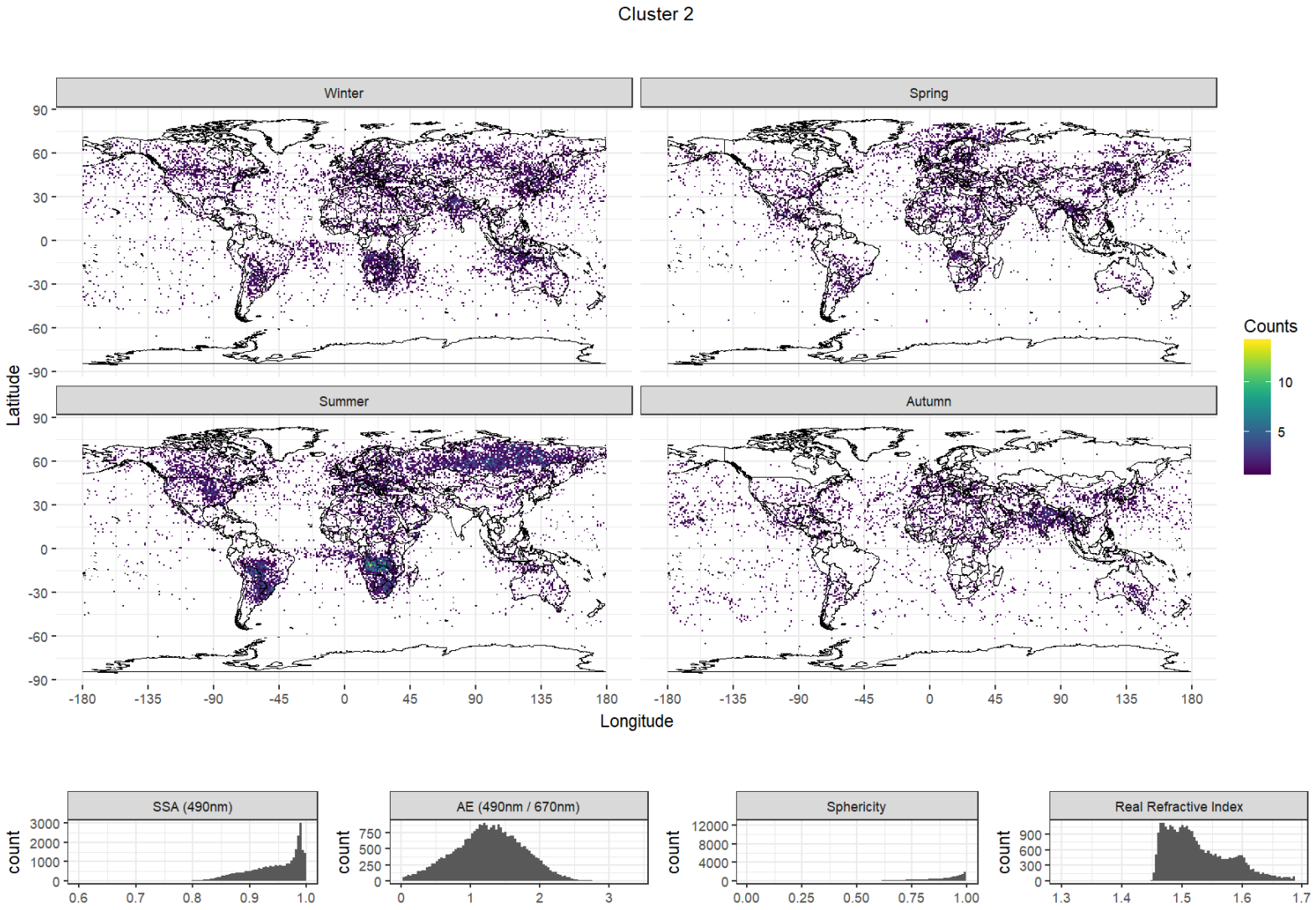


Figure 22: **Seasonal (northern hemisphere) geographical prevalence of and value distributions per variable within cluster 2.** Colour indicates the number of times a grid cell was assigned to this cluster within that season. Clustering results of the Self-Organizing Map with *nstart* = 10 on the POLDER-3 test set. SSA: Single Scattering Albedo; AE: Angstrom Exponent.

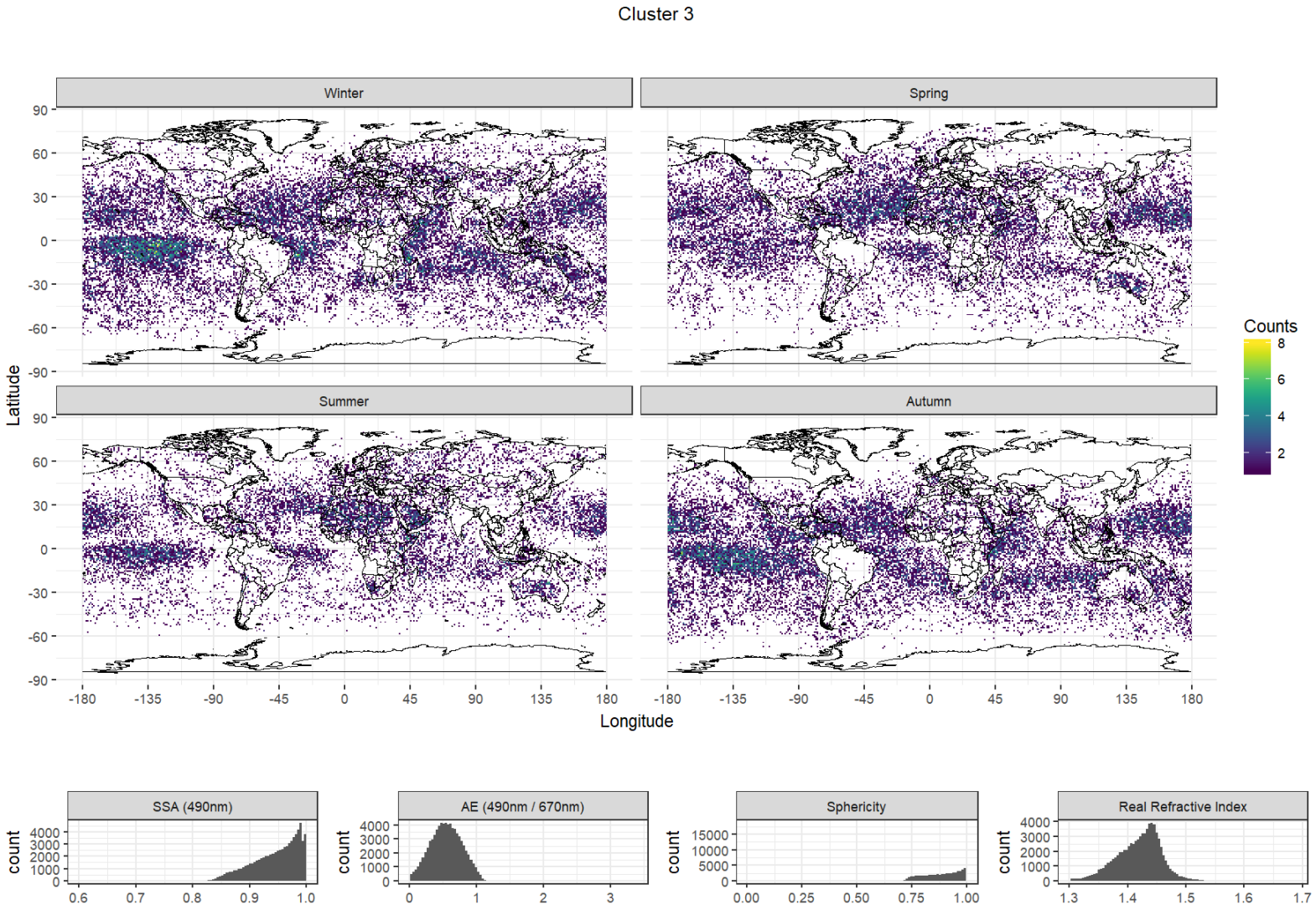


Figure 23: **Seasonal (northern hemisphere) geographical prevalence of and value distributions per variable within cluster 3.** Colour indicates the number of times a grid cell was assigned to this cluster within that season. Clustering results of the Self-Organizing Map with *nstart* = 10 on the POLDER-3 test set. SSA: Single Scattering Albedo; AE: Angstrom Exponent.

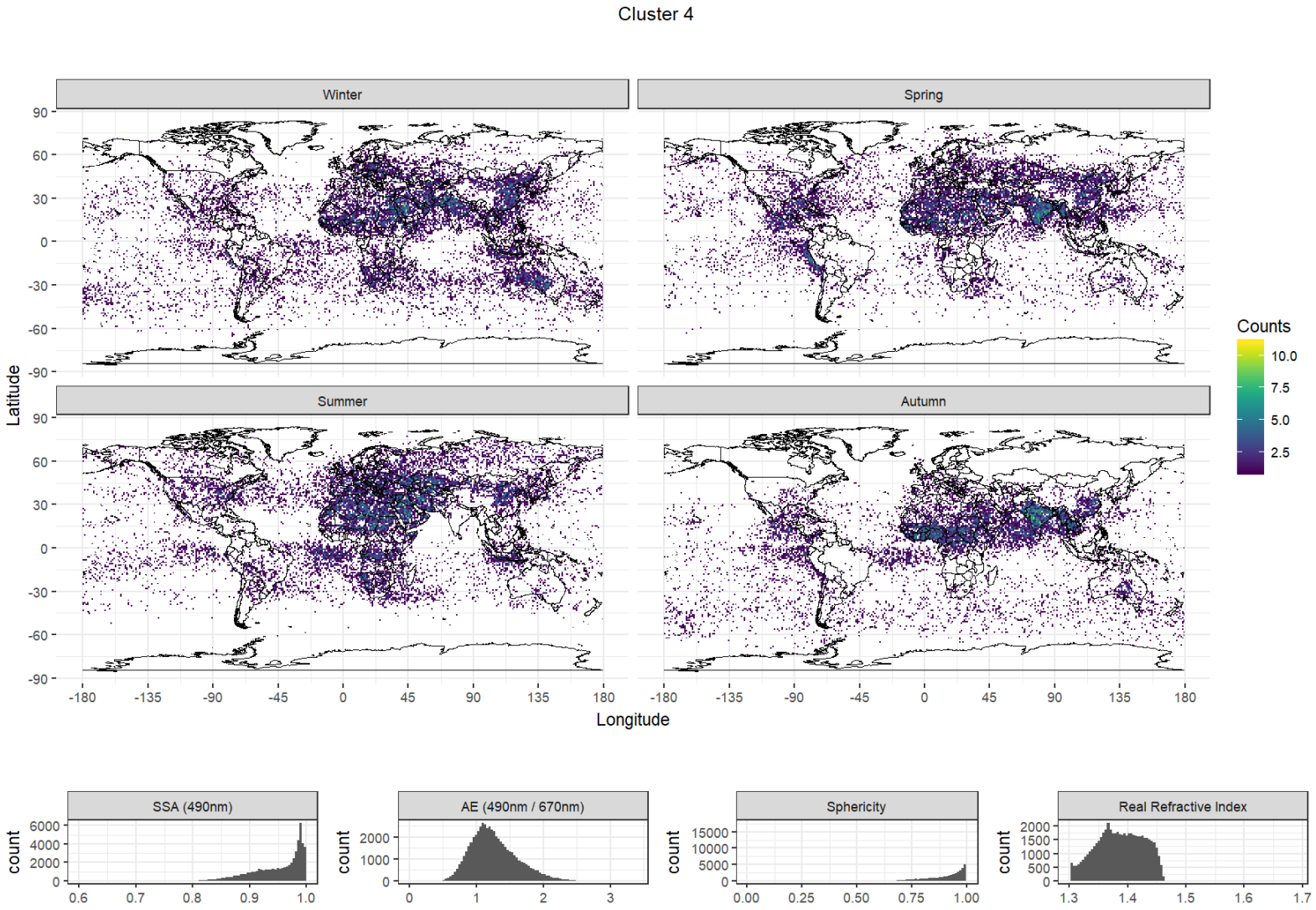


Figure 24: **Seasonal (northern hemisphere) geographical prevalence of and value distributions per variable within cluster 4.** Colour indicates the number of times a grid cell was assigned to this cluster within that season. Clustering results of the Self-Organizing Map with *nstart* = 10 on the POLDER-3 test set. SSA: Single Scattering Albedo; AE: Angstrom Exponent.

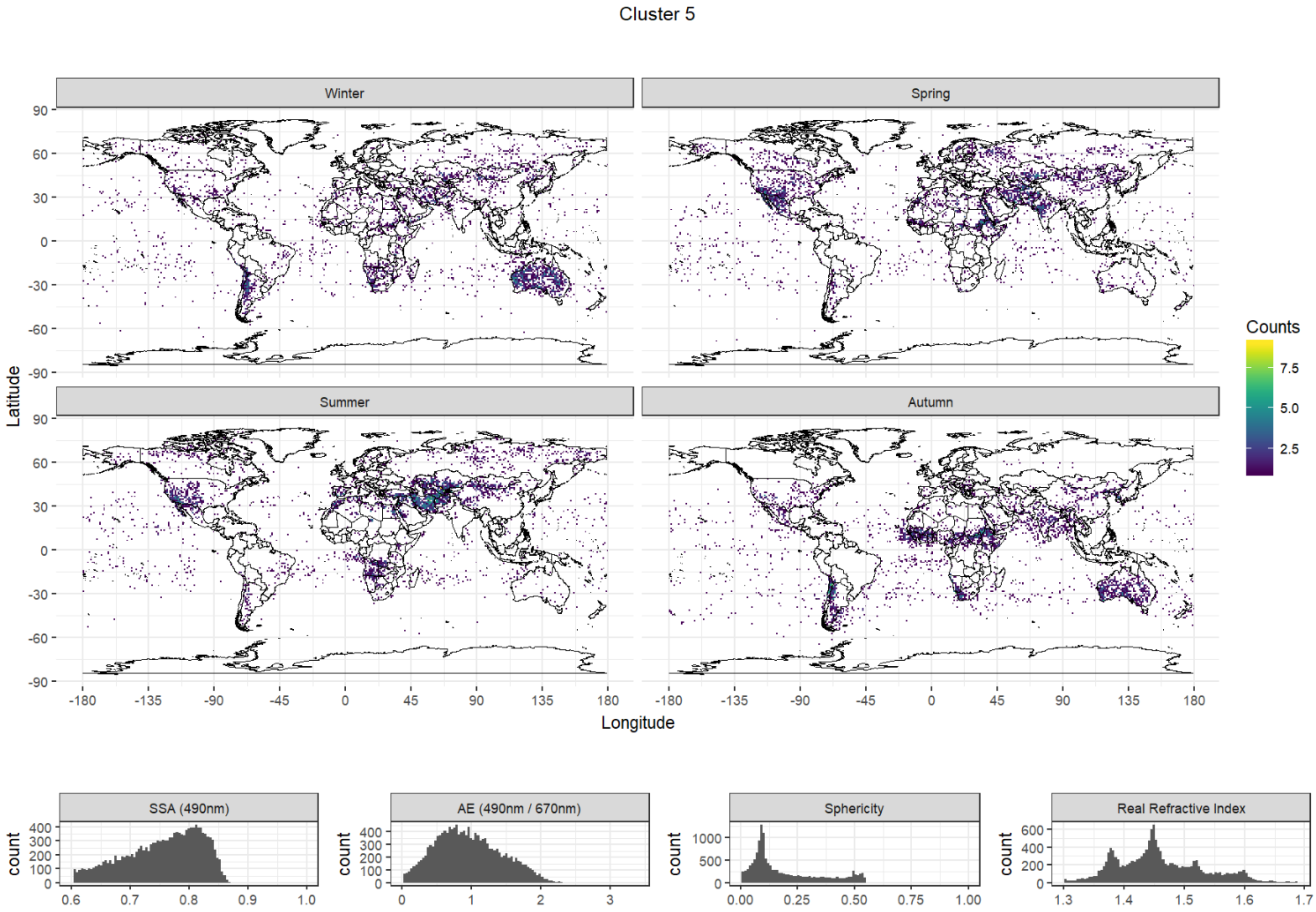


Figure 25: **Seasonal (northern hemisphere) geographical prevalence of and value distributions per variable within cluster 5.** Colour indicates the number of times a grid cell was assigned to this cluster within that season. Clustering results of the Self-Organizing Map with *nstart* = 10 on the POLDER-3 test set. SSA: Single Scattering Albedo; AE: Angstrom Exponent.

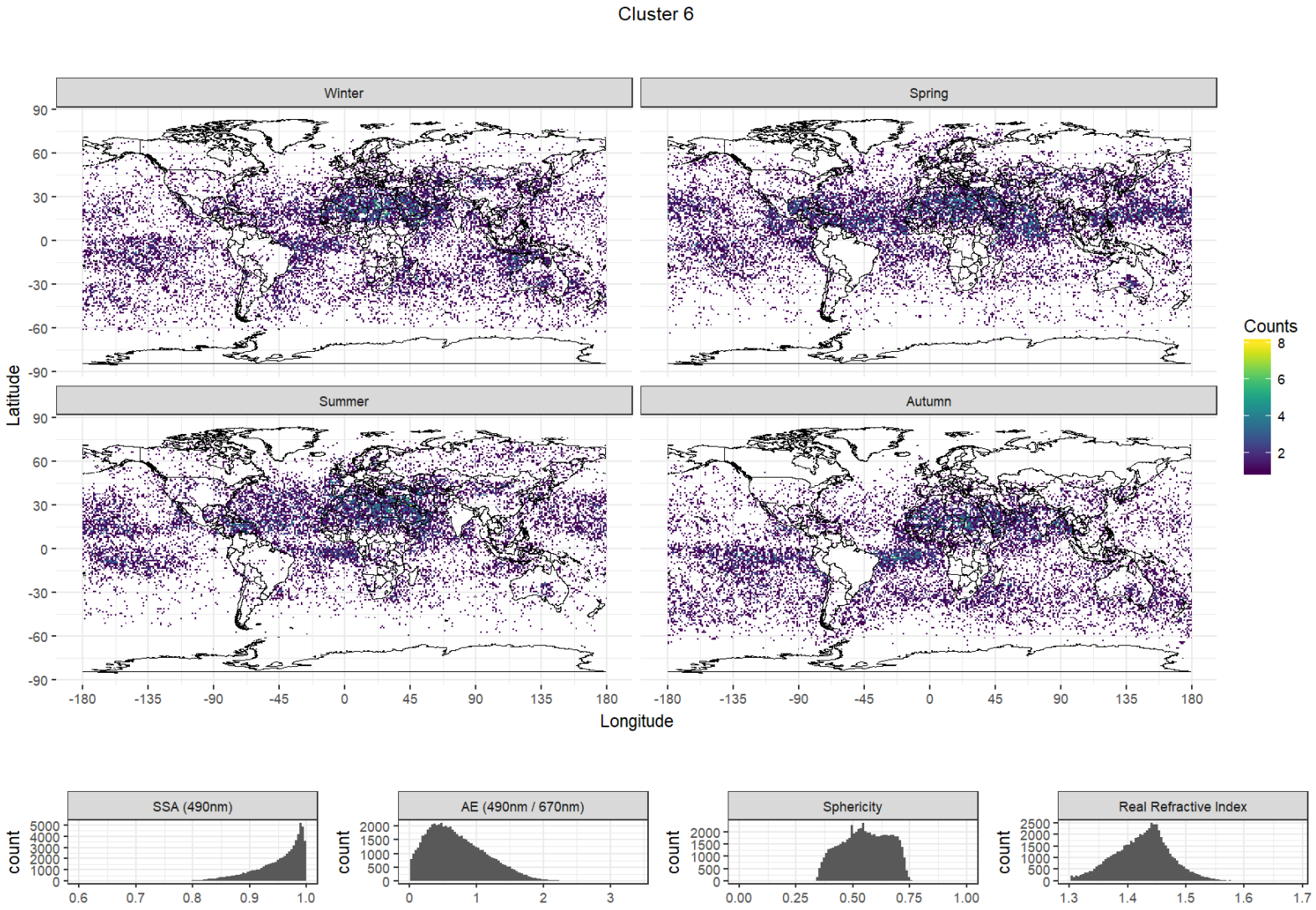


Figure 26: **Seasonal (northern hemisphere) geographical prevalence of and value distributions per variable within cluster 6.** Colour indicates the number of times a grid cell was assigned to this cluster within that season. Clustering results of the Self-Organizing Map with *nstart* = 10 on the POLDER-3 test set. SSA: Single Scattering Albedo; AE: Angstrom Exponent.

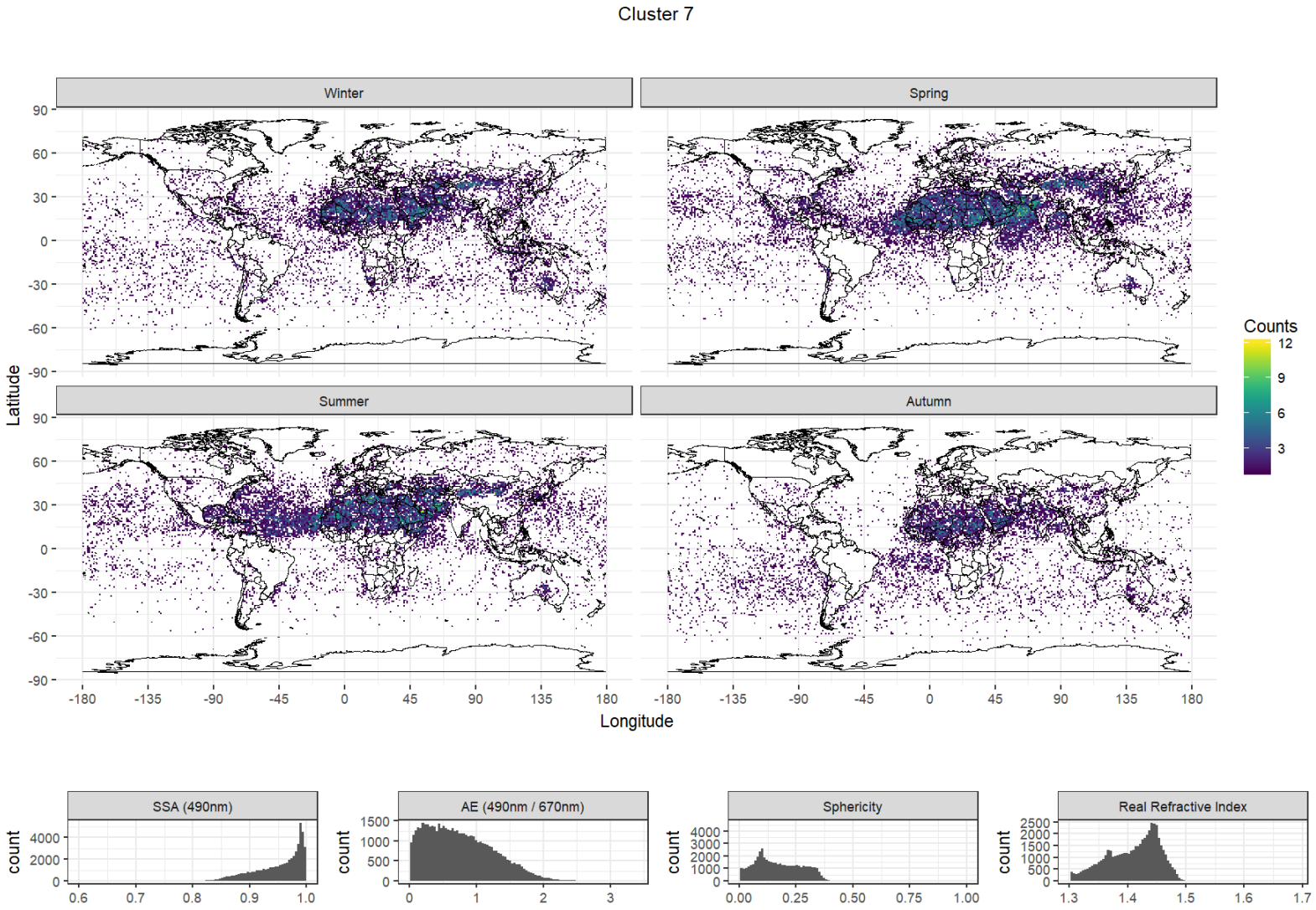


Figure 27: **Seasonal (northern hemisphere) geographical prevalence of and value distributions per variable within cluster 7.** Colour indicates the number of times a grid cell was assigned to this cluster within that season. Clustering results of the Self-Organizing Map with *nstart* = 10 on the POLDER-3 test set. SSA: Single Scattering Albedo; AE: Angstrom Exponent.

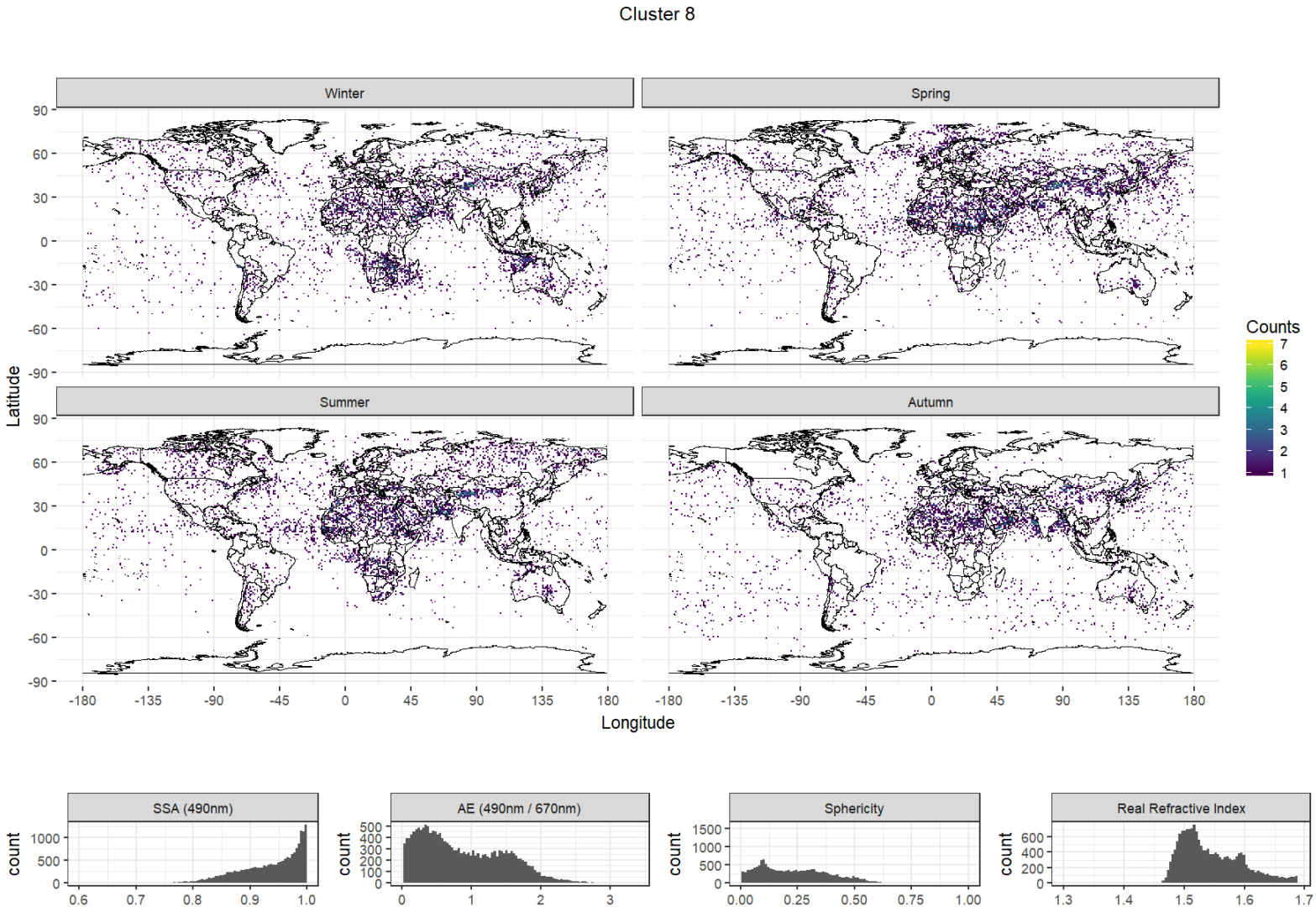


Figure 28: **Seasonal (northern hemisphere) geographical prevalence of and value distributions per variable within cluster 8.** Colour indicates the number of times a grid cell was assigned to this cluster within that season. Clustering results of the Self-Organizing Map with *nstart* = 10 on the POLDER-3 test set. SSA: Single Scattering Albedo; AE: Angstrom Exponent.

4 Conclusion and Discussion

In this work, we presented a method to identify clusters of aerosol types in the atmosphere based on their optical properties by making use of microphysical parameters derived from the 2006 POLDER-3 data. No *a priori* assumptions on aerosol types, number or nature are required, making use of unsupervised learning techniques. The proposed approach comprises a framework in which prototype-based clustering algorithms can be used and which allows for optimization of the number of clusters k in the data by means of (repeated) cross-validation of several cluster validation indices, thereby countering overfitting. The methods have been practically implemented in the statistical software R and examined in this thesis. Simulation studies have shown that all created software functions work properly and the methods can indeed be used for k optimization; at least in the basic case of scaled Gaussian distributed clusters. Subsequent application to a subset of the POLDER-3 data has demonstrated that meaningful clusters can be found in practice in this way as well, extending its potential to use on real-life data. A more elaborate discussion on the gathered results, the proposed methodology, its usefulness and limitations and suggestions for future work to study these aspects to a greater extent is presented below.

4.1 On the POLDER-3 Data Results

In order to find generic clusters of aerosol contents in the POLDER-3 data, the described cross-validation framework was used on a separate (stratified) train set, followed by a clustering of the test set with the thus optimized model parameter k. We used both the Validated Gridded Silhouette Width and Prediction Strength measures to determine the optimal value of k. A marked outcome was that the Silhouette indices suggested the presence of only $\hat{k} = 2$ clusters, whereas the Prediction Strength implied $\hat{k} = 8$. From a practical point of view, we know that there should almost certainly be more than 2 types of aerosol mixtures in the data: in a full year there should be more distinct mixtures of particles in the atmosphere around the complete globe. However, this does not explain the obtained result.

Recall, that the Silhouette Width is basically defined by two distances: the average within-cluster pairwise distance a (marking cohesion) and the average pairwise distance to the nearest neighbouring cluster b (marking separation). In our data set, a is mostly relatively large and b relatively small, due to the large spread of the data throughout the feature space (in essence, every data point is a mixture of aerosol types). Although b is still always larger than a (no Silhouette Width became less than 0 in Figure 18), this means that the data density at the cluster center has to be increasingly large for a cluster to get a high Silhouette Width. After all, only such a high density could compensate for the small b by reducing a. Ostensibly, sufficiently dense data masses are only achieved for $k = 2$ clusters. The Silhouette Width has furthermore been reported to be influenced by non-normality of clusters and high variance (high degree of overlap), in which case the indices become lower due to the distance-based makeup as the authors presumed [91]. Nevertheless, the Prediction Strength measure demonstrated that for higher k, clusters could still reliably, repeatedly be found in the data.

Indeed, we showed that the $k = 8$ clusters could stably be recovered and, importantly, appeared to bear meaningful information on reality: by characterizing the typical data patterns within clusters and studying their geographical dispersion, we could propose probable aerosol contents for each cluster. Bear in mind, that every datum in the data set is in principle a weighted average of a mixture of aerosols in the corresponding air column; i.e. presumably no observation belongs to a truly pure cluster. This is reflected by the fact that all clusters can be observed over geographical locations where they do not necessarily belong. Even clusters we expect to be relatively pure, as desert dust (cluster 7), can be seen to be spread out a little over the world maps (Figure 27).

The aerosol typing step following the clustering was performed in a manual fashion for this project. That is, by comparing the outcomes with available knowledge on aerosol optical properties in existing literature, we managed to generate probable aerosol labels per cluster.

Ideally, the aerosol typing procedure would be automated or standardized in some way for future studies. It seems that this step will inevitably involve comparisons to be made with results in other publications. One could use the centroids of reference clusters from other studies and assign each found cluster center to the closest centroid label. However, such an approach would disregard the potentially mixed clusters and, moreover, the fact that such a reference centroid is "closest" to a newly found centroid does not necessarily imply it is "close". It would already seem more sensible to compare the obtained centroids with only a few, major reference clusters (e.g. dust, marine, biomass burning and polution) from, for example, AERONET data: one could for each found centroid compute its relative probability to belong to each of those reference clusters, representing the extent to which a cluster may hold a certain type of aerosol. Nonetheless, this issue requires a great deal of attention first if it is to be challenged and this was beyond the scope of the current project. The method applied here has produced interpretable results, in light of which the methodology described and worked out here appears to be a reliable one to gain insight in aerosol distribution in the atmosphere.

Another use of the SOM is in a mere exploratory fashion, i.e. not for prediction purposes. This could prove very helpful in gaining insight in prevalence of aerosol types in the atmosphere as well, albeit only in the data at hand. One could opt for choosing a very large k, for instance 100x100 centroids and display the clustering in the form of the grid plot. On the grid plot, it may be possible to identify more or less pure clusters in the same manner as we did in our results section. Because the SOM preserves global topological ordering on the grid, the centroids in between roughly pure cluster centroids should represent mixtures thereof. The distance to such a "clean" centroid would indicate to what extent the corresponding aerosol type can be found in the mixed cluster. Additionally, the grid topology could be defined as toroidal, connecting the edges on either side. Naturally, clusters found in this way could again be depicted on actual world maps, which could verify any assigned aerosol labels. Using colour coding, one could potentially even display degrees of mixtures on the world maps, similar to Taylor et al. [20].

A logical next step in a practical sense could be to carry out the same analyses using more variables, or microphysical parameters. Many more can be included from the provided data and adding more dimensions to the feature space should never result in worse separation of the data. Such a study would shift the analysis towards a more severe machine learning approach. One could for example start with the same variables as used in Russell et al. [19] and compare the outcomes. However, bear in mind that this data set has many observations (large N), which brought us to create and use the Gridded Silhouette. This measure will, however, pose problems for higher-dimensional data (larger p), due to exponential growth of the total number of grid cells (c^p).

In addition, the POLDER-3 data set contains uncertainties on the microphysical parameters, resulting from the retrieval algorithm used to derive these variables from the raw data [101]. These could potentially be included in an extension of the methodology reported in this work. By illustration, Russell et al. [19] do this by treating every observation as an extended data point, taking into account their uncertainties; one could view every extended observation as a "pseudo-cluster" in its own respect. These uncertainties can then be incorporated in e.g. Mahalanobis distance calculations.

4.2 On the Methodology

When using our reported cross-validation framework, one should always consider the added value of cross-validating on a separate train set; i.e. is preventing overfitting an issue for the task at hand? The answer depends on the research objective: data exploration or prediction. In this work, we aimed to do the latter, in which it is important to set aside a test set and assess model quality on validation sets in order to find a generic solution. However, cluster analyses are often carried out with the former objective, in which k may be found by applying a(n) (cross-validation) optimization scheme over the full data at once, followed by a clustering of the same data set with the optimized k. The `cv.clust()` function constructed here allows for this possibility as well.

Additionally, when a test set is used like here, the relative size of this set may be varied: for large data sets as in our case, it may be worthwhile to enlarge the test set, since the train set might nonetheless very well remain sufficiently large for proper training. A study of the effects of different train:test ratios, as well as varying values of both the number of cross-validation repeats M and the number of folds v therein, was not within the scope of this work.

A second consideration involves the way in which the clustering is performed. Here, we demonstrated the use of two clustering algorithms: k-means and the Self-Organizing Map (SOM). Since these algorithms are both prototype-based, the `cv.clust()` software can treat the objects made by both clusterings nearly in the same way: it takes the names of the elements within the objects that contain the centroids and cluster labels as input. In turn, this means that the implementation of other such clustering algorithms would only require minor code adaptations, if any. Besides for instance k-medoids methods as PAM (Partitioning Around Medoids) [102, 103] or others [104], or for exceedingly large data sets CLARA (Clustering LARge Applications) [103], we have contemplated using density-based clustering algorithms. A prime candidate would be the implementation of the popular DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [105]. This algorithm does not require k to be specified, but rather two other parameters, for whose optimization the cross-validation scheme could be used too. Validation fold data could then be assigned to train clusters by matching each of them to the label of their nearest train fold datum. For DBSCAN usage, the clusters to identify do not need to be (approximately) Gaussian distributed; validation of clustering quality by Silhouette Width may therefore also not be advisable [91].

Prototype-based clusterings can be stabilized by means of *nstart* or EMcMc (EM-centered Mahalanobis clustering) procedures as we report. Both are de facto *nstart* mechanisms: in both cases, the exact same clustering run has to be invoked a set number of times. The main difference is that the former picks the "best" of these runs, whereas the latter averages the centroids in feature space (and assigns the data anew based on Mahalanobis distances). Hence, the code facilitating these methods could be made more generic, allowing the user to specify the desired stabilization mode, if any. An additional option could be to let the user define the metric as well (e.g. Mahalanobis or Euclidean). We wish to emphasize that EMcMc is a – to our knowledge – new concept, of which the applicability and performance have yet to be examined to a considerably larger extent, also in simulation studies. In addition, it should be noted that EMcMc will only produce meaningful results for a k that yields stable results over the iterations. As multiple authors have suggested, inverting this principle can be used as foundation for a cluster validation index: stable results over the same runs with different sets of initiation points can be an indication for proper k specification [78, 79].

Furthermore, the uses of the SOM algorithm extend way beyond our application in this work. Similar to common k-means clustering, a batch version exists for the SOM and is even available in the **kohonen** package. Since the creator of the SOM himself recommends the use of the batch version (claiming it is faster and obsolesces the need to set the learning rate parameter), it would be wise to at least investigate its use for this application as well [44]. Other SOM input parameter settings could be explored too, on top of the sheer default options we used in our studies. Straightforward options are to examine the influence of alternative neighbourhood functions, grid topology, number of epochs and both starting values and decay functions of the learning rate and neighbourhood radius. It should be noted that the possibilities for some of these options are limited in the current **kohonen** package (e.g. no exponential decay functions).

A few selected cluster validation measures were implemented for this thesis. Naturally, other indices can be added to the framework in the future; the R code is composed in such a manner that it should not be complicated to do so. In concert with the `cv.clust()` function, the `ps()` function works comparably to the **fpc** package to calculate Prediction Strength. Whereas our implementation allows for specification of v in addition to M, which is not possible in the **fpc** version, the latter has a few advantages of its own for now as well: more clustering algorithms are already integrated for usage, as well as more ways to assign the data of the validation fold a second label. However, our combination of creating `ps()` as a building block to include it in a

larger cross-validation function could in time be more flexible: other measures can be included in addition.

The other cluster validation measures currently built into `cv.clust()` are variants of the Silhouette Width. The Validated Silhouettes are computed over the clusters found by training centroids in the validation folds and are thus informative on how well a trained clustering can partition a validation set of data into k regions. As illustrated in Figure 13, found differences over folds may be very small even for incorrect k, rising the question whether v-fold cross-validation has any added value for this index. It appears that the index will only vary over folds if the samples in the different folds are not comparable. This would suggest that the use of $v = 2$ folds is sufficient for both the Average and the Minimum Validated Silhouette Widths presented here.

The addition of a minimum validated index was originally inspired by the Prediction Strength calculation and denotes a measure of the Minimax decision format: minimize the maximum loss. In other words: consider all worst-case scenarios and pick the best of those. Hence, a clustering is only as good as its worst fitting cluster: as soon as just one cluster has a bad fit according to the index, the whole clustering index goes down. This approach is thus more sensitive for outlying clusters, which might make its usage more desirable in case of many "true" clusters: the bad fits within a clustering cannot be averaged out by many good fits. Nevertheless, a thorough examination would be needed to state in which cases the use of the minimum may be prefered over the average.

For data sets with many observations (large N), the Silhouette Width cannot be computed due to computational cost. Here, a solution is presented in the form of the Gridded Silhouette Width, which approximates the original measure. We showed that the new index has been successfully implemented in the software, indeed approximates the real Silhouette Width and can aid in k optimization through the use of the AVGS and MVGS as well. On the downside, it requires the choice of an extra parameter value: the number of grid cells per dimension c. The optimization of c is a balance between the approximation accuracy and the computational cost, which both increase with c. Suggestions for c optimization have been presented, but an inquiry into the matter remains to be carried out. The accuracy of the Gridded Silhouette Width is furthermore influenced by the parameter k of the clustering: larger k requires a finer grid to cover all clusters sufficiently.

Taken together, the framework has been set up and works for k optimization and clustering purposes. However, a wealth of experiments is yet to be performed to test the performance of the proposed methods in a wide range of situations. Simulation studies can be carried out to assess for example the effects of other underlying distributions (non-Gaussian, in which a potential implementation of DBSCAN would pose an advantage) and their dimensionality, different degrees of cohesion and separation, and varying sample sizes over clusters. Moreover, the influence of various values for M, v and c can be examined, as well as the use of alternative validation measures, clustering algorithms and their settings.

Since all of these aspects of the methodology remain to be studied, caution should be taken in its application. After all, its efficacy has only been demonstrated here for a few selected, relatively simple situations. Additionaly, it is important to consider the objective of the cluster analysis to be performed ahead of `cv.clust()` usage. The choice of cluster algorithm and validation tools depend heavily on the type of clusters one expects (distribution, overlap, etc.) and the clustering purpose (exploration or prediction) [106]. So, the first question to be answered should always be: what are the "true" (or "natural") clusters that we wish to model? The answer may oftenwise be strikingly devious: one could argue that the "true" clusters for our POLDER-3 analysis are the ones that ideally represent pure aerosol types. In practice, however, these could not be modelled due to the abundance of data that embody mixtures of such types. Hence, for our analysis, the stably occuring aerosol mixtures could be seen as the "true" clusters. Even though we did thus not model the underlying reality (pure types), but rather the patterns caused by it (mixtures), we demonstrated that our approach can still be informative on reality. Nonetheless, one should always bear in mind what the clusters resulting from a cluster analysis represent and how they are meaningful in the interpretation of nature's workings, in the real world.

In summary, we showed a means to retrieve clusters of atmospheric columns that are comparable in aerosol contents, studied the performance of the proposed clustering and validation methods and discussed their workings, interpreted the generated clusters and demonstrated how and to what extent they may describe the aerosol distribution in the atmosphere of the Earth. Such information – potentially based on more locally acquired data with enhanced resolution in the forseeable future – could support (local) governments in shaping their policies regarding air quality for improvement of public health, in which atmospheric aerosols play a key role in this day and age.

Bibliography

[1] Boucher, O. et al. "Clouds and Aerosols". In: *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change* 7 (2013), pp. 571–657. DOI: 10.1017/CBO9781107415324.016. URL: http://www.ipcc.ch/report/ar5/wg1/.

[2] World Health Organization, WHO. *Air Quality Guidelines Update 2005*. 2006. ISBN: 9289021926.

[3] Valavanidis, Athanasios, Fiotakis, Konstantinos, and Vlachogianni, Thomais. "Airborne particulate matter and human health: toxicological assessment and importance of size and composition of particles for oxidative damage and carcinogenic mechanisms". In: *Journal of Environmental Science and Health, Part C* 26.4 (2008), pp. 339–362.

[4] Anderson, Jonathan O, Thundiyil, Josef G, and Stolbach, Andrew. "Clearing the air: a review of the effects of particulate matter air pollution on human health". In: *Journal of Medical Toxicology* 8.2 (2012), pp. 166–175.

[5] Kim, Ki-Hyun, Kabir, Ehsanul, and Kabir, Shamin. "A review on the human health impact of airborne particulate matter". In: *Environment international* 74 (2015), pp. 136–143.

[6] Kahn, Ralph A. and Gaitley, Barbara J. "An analysis of global aerosol type as retrieved by MISR". In: *Journal of Geophysical Research: Atmospheres* 120.9 (2015). 2015JD023322, pp. 4248–4281. ISSN: 2169-8996. DOI: 10.1002/2015JD023322. URL: http://dx.doi.org/10.1002/2015JD023322.

[7] Omar, Ali H et al. "The CALIPSO automated aerosol classification and lidar ratio selection algorithm". In: *Journal of Atmospheric and Oceanic Technology* 26.10 (2009), pp. 1994–2014.

[8] Remer, Lorraine A et al. "Global aerosol climatology from the MODIS satellite sensors". In: *Journal of Geophysical Research: Atmospheres* 113.D14 (2008).

[9] Kacenelenbogen, M et al. "Characterization of aerosol pollution events in France using ground-based and POLDER-2 satellite data". In: *Atmospheric Chemistry and Physics* 6.12 (2006), pp. 4843–4849.

[10] Su, X et al. "Aerosol variability over East Asia as seen by POLDER space-borne sensors". In: *Journal of Geophysical Research: Atmospheres* 115.D24 (2010).

[11] Harrison, Roy M and Yin, Jianxin. "Particulate matter in the atmosphere: which particle properties are important for its effects on health?" In: *Science of the total environment* 249.1-3 (2000), pp. 85–101.

[12] Kelly, Frank J and Fussell, Julia C. "Size, source and chemical composition as determinants of toxicity attributable to ambient particulate matter". In: *Atmospheric Environment* 60 (2012), pp. 504–526.

[13] Eck, TF et al. "Wavelength dependence of the optical depth of biomass burning, urban, and desert dust aerosols". In: *Journal of Geophysical Research: Atmospheres* 104.D24 (1999), pp. 31333–31349.

[14] Dubovik, Oleg et al. "Variability of absorption and optical properties of key aerosol types observed in worldwide locations". In: *Journal of the atmospheric sciences* 59.3 (2002), pp. 590–608.

[15] Omar, Ali H et al. "Development of global aerosol models using cluster analysis of Aerosol Robotic Network (AERONET) measurements". In: *Journal of Geophysical Research: Atmospheres* 110.D10 (2005).

[16] Kalapureddy, MCR et al. "Identification of aerosol type over the Arabian Sea in the premonsoon season during the Integrated Campaign for Aerosols, Gases and Radiation Budget (ICARB)". In: *Journal of Geophysical Research: Atmospheres* 114.D17 (2009).

[17] Lee, Jaehwa et al. “Characteristics of aerosol types from AERONET sunphotometer measurements”. In: *Atmospheric Environment* 44.26 (2010), pp. 3110–3117.

[18] Giles, David M et al. “An analysis of AERONET aerosol absorption properties and classifications representative of aerosol source regions”. In: *Journal of Geophysical Research: Atmospheres* 117.D17 (2012).

[19] Russell, Philip B. et al. “A multiparameter aerosol classification method and its application to retrievals from spaceborne polarimetry”. In: *Journal of Geophysical Research: Atmospheres* 119.16 (2014). 2013JD021411, pp. 9838–9863. ISSN: 2169-8996. DOI: 10.1002/2013JD021411. URL: http://dx.doi.org/10.1002/2013JD021411.

[20] Taylor, M et al. “Global aerosol mixtures and their multiyear and seasonal characteristics”. In: *Atmospheric Environment* 116 (2015), pp. 112–129.

[21] Chin, Mian et al. “Tropospheric aerosol optical thickness from the GOCART model and comparisons with satellite and Sun photometer measurements”. In: *Journal of the atmospheric sciences* 59.3 (2002), pp. 461–483.

[22] Kohonen, Teuvo. “Self-organized formation of topologically correct feature maps”. In: *Biological cybernetics* 43.1 (1982), pp. 59–69.

[23] Visser, Guido. “Unsupervised aerosol classification from POLDER-3 data using self-organizing maps”. Aug. 2017.

[24] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2017. URL: https://www.R-project.org/.

[25] Steinley, Douglas. “K-means clustering: a half-century synthesis”. In: *British Journal of Mathematical and Statistical Psychology* 59.1 (2006), pp. 1–34.

[26] Jain, Anil K. “Data clustering: 50 years beyond K-means”. In: *Pattern recognition letters* 31.8 (2010), pp. 651–666.

[27] Steinley, Douglas. “Local optima in K-means clustering: what you don’t know may hurt you.” In: *Psychological methods* 8.3 (2003), p. 294.

[28] Hancer, Emrah and Karaboga, Dervis. “A comprehensive survey of traditional, merge-split and evolutionary approaches proposed for determination of cluster number”. In: *Swarm and Evolutionary Computation* 32 (2017), pp. 49–67.

[29] Tibshirani, Robert and Walther, Guenther. “Cluster validation by prediction strength”. In: *Journal of Computational and Graphical Statistics* 14.3 (2005), pp. 511–528.

[30] Rousseeuw, Peter J. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[31] Pierce, David. *ncdf4: Interface to Unidata netCDF (Version 4 or Earlier) Format Data Files*. R package version 1.16. 2017. URL: https://CRAN.R-project.org/package=ncdf4.

[32] Hamill, Patrick et al. “An AERONET-based aerosol classification using the Mahalanobis distance”. In: *Atmospheric Environment* 140 (2016), pp. 213–233.

[33] Giglio, Louis, Csiszar, Ivan, and Justice, Christopher O. “Global distribution and seasonality of active fires as observed with the Terra and Aqua Moderate Resolution Imaging Spectroradiometer (MODIS) sensors”. In: *Journal of Geophysical Research: Biogeosciences (2005–2012)* 111.G2 (June 2006). ISSN: 2156-2202. DOI: 10.1029/2005JG000142. URL: http:https://doi.org/10.1029/2005JG000142.

[34] Krawchuk, Meg A et al. “Global pyrogeography: the current and future distribution of wildfire”. In: *PloS one* 4.4 (2009), e5102.

[35] Hale, George M and Querry, Marvin R. “Optical constants of water in the 200-nm to 200-μm wavelength region”. In: *Applied optics* 12.3 (1973), pp. 555–563.

[36] Schuster, Gregory L et al. “Inferring black carbon content and specific absorption from Aerosol Robotic Network (AERONET) aerosol retrievals”. In: *Journal of Geophysical Research: Atmospheres* 110.D10 (2005).

[37] Steinhaus, Hugo. “Sur la division des corp materiels en parties”. In: *Bull. Acad. Polon. Sci* 1.804 (1956), p. 801.

[38] Lloyd, Stuart. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982). Originally as an unpublished Bell laboratories Technical Note (1957), pp. 129–137.

[39] Ball, Geoffrey H and Hall, David J. *ISODATA, a novel method of data analysis and pattern classification*. Tech. rep. Stanford research inst Menlo Park CA, 1965.

[40] Forgey, Edward. “Cluster analysis of multivariate data: Efficiency vs. interpretability of classification”. In: *Biometrics* 21.3 (1965), pp. 768–769.

[41] MacQueen, James et al. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[42] Hartigan, John A and Wong, Manchek A. “Algorithm AS 136: A k-means clustering algorithm”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108.

[43] Fort, Jean-Claude. “SOM’s mathematics”. In: *Neural Networks* 19.6-7 (2006), pp. 812–816.

[44] Kohonen, Teuvo. “Essentials of the self-organizing map”. In: *Neural networks* 37 (2013), pp. 52–65.

[45] Yan, Jun. *som: Self-Organizing Map*. R package version 0.3-5.1. 2016. URL: https://CRAN.R-project.org/package=som.

[46] Wehrens, R. and Buydens, L.M.C. “Self- and Super-organising Maps in R: the kohonen package”. In: *J. Stat. Softw.* 21.5 (2007). URL: http://www.jstatsoft.org/v21/i05.

[47] López-Rubio, Ezequiel and Ramos, Antonio Díaz. “Grid topologies for the self-organizing map”. In: *Neural Networks* 56 (2014), pp. 35–48.

[48] Kohonen, Teuvo. “The self-organizing map”. In: *Neurocomputing* 21.1-3 (1998), pp. 1–6.

[49] Cheng, Yizong. “Convergence and ordering of Kohonen’s batch map”. In: *Neural Computation* 9.8 (1997), pp. 1667–1676.

[50] Steinley, Douglas. “Profiling local optima in K-means clustering: Developing a diagnostic technique.” In: *Psychological methods* 11.2 (2006), p. 178.

[51] Dempster, Arthur P, Laird, Nan M, and Rubin, Donald B. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[52] Fraley, Chris and Raftery, Adrian E. “Model-based Clustering, Discriminant Analysis and Density Estimation”. In: *Journal of the American Statistical Association* 97 (2002), pp. 611–631.

[53] Fraley, Chris et al. *mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation*. 597. Technical Report. 2012.

[54] Mahalanobis, Prasanta Chandra. “On the generalized distance in statistics”. In: National Institute of Science of India. 1936.

[55] Arbelaitz, Olatz et al. “An extensive comparative study of cluster validity indices”. In: *Pattern Recognition* 46.1 (2013), pp. 243–256.

[56] Schwarz, Gideon et al. “Estimating the dimension of a model”. In: *The annals of statistics* 6.2 (1978), pp. 461–464.

[57] Fraley, Chris and Raftery, Adrian E. "Model-based clustering, discriminant analysis, and density estimation". In: *Journal of the American statistical Association* 97.458 (2002), pp. 611–631.

[58] Wang, Weina and Zhang, Yunjie. "On fuzzy cluster validity indices". In: *Fuzzy sets and systems* 158.19 (2007), pp. 2095–2117.

[59] Tibshirani, Robert, Walther, Guenther, and Hastie, Trevor. "Estimating the number of clusters in a data set via the gap statistic". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423.

[60] Sugar, Catherine A., Lenert, Leslie A., and Olshen, Richard A. *An Application of Cluster Analysis to Health Services Research: Empirically Defined Health States for Depression from the SF-12*. Tech. rep. 1999.

[61] Sugar, Catherine A and James, Gareth M. "Finding the number of clusters in a dataset: An information-theoretic approach". In: *Journal of the American Statistical Association* 98.463 (2003), pp. 750–763.

[62] Caliński, Tadeusz and Harabasz, Jerzy. "A dendrite method for cluster analysis". In: *Communications in Statistics-theory and Methods* 3.1 (1974), pp. 1–27.

[63] Dunn, Joseph C. "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters". In: (1973).

[64] Davies, David L and Bouldin, Donald W. "A cluster separation measure". In: *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), pp. 224–227.

[65] Bezdek, James C and Pal, Nikhil R. "Cluster validation with generalized Dunn's indices". In: *Artificial Neural Networks and Expert Systems, 1995. Proceedings., Second New Zealand International Two-Stream Conference on*. IEEE. 1995, pp. 190–193.

[66] Pal, Nikhil R and Biswas, J. "Cluster validation using graph theoretic concepts". In: *Pattern Recognition* 30.6 (1997), pp. 847–857.

[67] Maechler, Martin et al. *cluster: Cluster Analysis Basics and Extensions*. R package version 2.0.6. 2017.

[68] Luna-Romera, José María et al. "An approach to silhouette and dunn clustering indices applied to big data in spark". In: *Conference of the Spanish Association for Artificial Intelligence*. Springer. 2016, pp. 160–169.

[69] Luna-Romera, José María et al. "An approach to validity indices for clustering techniques in Big Data". In: *Progress in Artificial Intelligence* (2017), pp. 1–14.

[70] Levine, Erel and Domany, Eytan. "Resampling method for unsupervised estimation of cluster validity". In: *Neural computation* 13.11 (2001), pp. 2573–2593.

[71] Ben-Hur, Asa, Elisseeff, Andre, and Guyon, Isabelle. "A stability based method for discovering structure in clustered data". In: *Biocomputing 2002*. World Scientific, 2001, pp. 6–17.

[72] Dudoit, Sandrine and Fridlyand, Jane. "A prediction-based resampling method for estimating the number of clusters in a dataset". In: *Genome biology* 3.7 (2002), research0036–1.

[73] Lange, Tilman et al. "Stability-based model selection". In: *Advances in neural information processing systems*. 2003, pp. 633–642.

[74] Lange, Tilman et al. "Stability-based validation of clustering solutions". In: *Neural computation* 16.6 (2004), pp. 1299–1323.

[75] Hennig, Christian. "Cluster-wise assessment of cluster stability". In: *Computational Statistics & Data Analysis* 52.1 (2007), pp. 258–271.

[76] Hennig, Christian. "Dissolution point and isolation robustness: robustness criteria for general cluster analysis methods". In: *Journal of multivariate analysis* 99.6 (2008), pp. 1154–1176.

[77] Bertrand, Patrice and Mufti, G Bel. "Loevinger's measures of rule quality for assessing cluster stability". In: *Computational Statistics & Data Analysis* 50.4 (2006), pp. 992–1015.

[78] Steinley, Douglas. "Stability analysis in K-means clustering". In: *British Journal of Mathematical and Statistical Psychology* 61.2 (2008), pp. 255–273.

[79] Lord, Etienne et al. "Using the stability of objects to determine the number of clusters in datasets". In: *Information Sciences* 393 (2017), pp. 29–46.

[80] Wang, Junhui. "Consistent selection of the number of clusters via crossvalidation". In: *Biometrika* 97.4 (2010), pp. 893–904.

[81] Fang, Yixin and Wang, Junhui. "Selection of the number of clusters via the bootstrap method". In: *Computational Statistics & Data Analysis* 56.3 (2012), pp. 468–477.

[82] Ben-David, S, Von Luxburg, U, and Pal, D. "A sober look at stability of clustering". In: *Proceedings of the Annual Conference on Computational Learning Theory*. 2006.

[83] Ben-David, Shai, Pál, Dávid, and Simon, Hans Ulrich. "Stability of k-means clustering". In: *International Conference on Computational Learning Theory*. Springer. 2007, pp. 20–34.

[84] Von Luxburg, Ulrike. "Clustering stability: An overview". In: *Foundations and Trends® in Machine Learning* 2.3 (2010), pp. 235–274.

[85] Tibshirani, Robert et al. *Cluster validation by prediction strength, Department of Statistics*. 2001.

[86] Hennig, Christian. *fpc: Flexible Procedures for Clustering*. R package version 2.1-10. 2015. URL: https://CRAN.R-project.org/package=fpc.

[87] Breiman, Leo and Spector, Philip. "Submodel selection and evaluation in regression. The X-random case". In: *International statistical review/revue internationale de Statistique* (1992), pp. 291–319.

[88] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1995, pp. 1137–1143.

[89] Rodriguez, Juan D, Perez, Aritz, and Lozano, Jose A. "Sensitivity analysis of k-fold cross validation in prediction error estimation". In: *IEEE transactions on pattern analysis and machine intelligence* 32.3 (2010), pp. 569–575.

[90] Borra, Simone and Di Ciaccio, Agostino. "Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods". In: *Computational statistics & data analysis* 54.12 (2010), pp. 2976–2989.

[91] Brun, Marcel et al. "Model-based evaluation of clustering validation measures". In: *Pattern recognition* 40.3 (2007), pp. 807–824.

[92] Schaul, Tom et al. "PyBrain". In: *Journal of Machine Learning Research* 11 (2010), pp. 743–746.

[93] Conard, Susan G and Ivanova, Galina A. "Wildfire in Russian boreal forests—Potential impacts of fire regime characteristics on emissions and global carbon balance estimates". In: *Environmental Pollution* 98.3 (1997), pp. 305–313.

[94] Saltzman, ES et al. "Elevated atmospheric sulfur levels off the Peruvian coast". In: *Journal of Geophysical Research: Atmospheres* 91.D7 (1986), pp. 7913–7918.

[95] Prospero, Joseph M et al. "Environmental characterization of global sources of atmospheric soil dust identified with the Nimbus 7 Total Ozone Mapping Spectrometer (TOMS) absorbing aerosol product". In: *Reviews of geophysics* 40.1 (2002).

[96] Prospero, JM. "Saharan dust transport over the North Atlantic Ocean and Mediterranean: an overview". In: *The impact of desert dust across the Mediterranean*. Springer, 1996, pp. 133–151.

[97] Kalu, AE. "The African dust plume: its characteristics and propagation across West Africa in winter". In: *Scope* 14 (1979), pp. 95–118.

[98] Moulin, Cyril et al. "Control of atmospheric export of dust from North Africa by the North Atlantic Oscillation". In: *Nature* 387.6634 (1997), p. 691.

[99] Schepanski, Kerstin, Tegen, I, and Macke, Andreas. "Saharan dust transport and deposition towards the tropical northern Atlantic." In: *Atmospheric Chemistry & Physics* 9.4 (2009).

[100] Chiapello, Isabelle et al. "Aerosol detection by TOMS and POLDER over oceanic regions". In: *Journal of Geophysical Research: Atmospheres* 105.D6 (2000), pp. 7133–7142.

[101] Hasekamp, Otto P, Litvinov, Pavel, and Butz, André. "Aerosol properties over the ocean from PARASOL multiangle photopolarimetric measurements". In: *Journal of Geophysical Research: Atmospheres* 116.D14 (2011).

[102] Kaufman, Leonard and Rousseeuw, Peter. "Clustering by Means of Medoids". In: (Jan. 1987), pp. 405–416.

[103] Kaufman, Leonard and Rousseeuw, Peter J. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons, 1990.

[104] Park, Hae-Sang and Jun, Chi-Hyuck. "A simple and fast algorithm for K-medoids clustering". In: *Expert systems with applications* 36.2 (2009), pp. 3336–3341.

[105] Ester, Martin et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[106] Hennig, Christian. "What are the true clusters?" In: *Pattern Recognition Letters* 64 (2015), pp. 53–62.

[107] Zadeh, L.A. "Fuzzy sets". In: *Information and Control* 8.3 (1965), pp. 338–353. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(65)90241-X. URL: http://www.sciencedirect.com/science/article/pii/S001999586590241X.

[108] Bezdek, James C, Ehrlich, Robert, and Full, William. "FCM: The fuzzy c-means clustering algorithm". In: *Computers & Geosciences* 10.2-3 (1984), pp. 191–203.

[109] Peters, Georg et al. "Soft clustering–fuzzy and rough approaches and their extensions and derivatives". In: *International Journal of Approximate Reasoning* 54.2 (2013), pp. 307–322.

Appendices

A Soft Clustering in Aerosol Typing

The aim of both k-means and SOM clustering in this work is to assign each datum from a data set $\mathbf{X}$ to one of k clusters. This type of clustering is called **hard clustering**, as each datum belongs to only one specific cluster *and none other*. Even when an observation is right in between two dense masses of data, it will be assigned to either one of the two clusters. Opposingly, in **soft clustering** methods, each datum is assigned to multiple or all clusters with a certain degree of "belongness". This concept corresponds to the idea of fuzzy sets, as introduced by Zadeh [107].

Considering the ultimate goal of clustering atmospheric columns on grounds of aerosol content, this perspective might make sense here. Every observation in the data at hand corresponds to such a column and every column represents a massive volume of air. So, these columns are unlikely to contain merely one type of aerosol each. Hence, the observations in our data set embody mixtures of aerosol types and therefore weighted averages of the microphysical parameters. Ideally, our cluster centroids represent pure aerosol types, instead of mixtures. Mixtures would emerge naturally in soft clustering for data points that are, to a certain degree, assigned to multiple clusters: observations would belong to various "pure" aerosol types (clusters) with different weights. In other words: hard clustering methods would model the patterns (mixtures), whereas soft clustering approaches might model the underlying reality (pure types), after which the observed patterns would emerge.

Fuzzy c-means

A basic soft clustering method is that of *Fuzzy c-means* (FCM) [108]. Originally proposed as fuzzy variant of Ball and Hall's ISODATA algorithm [39] by Dunn [63], FCM basically extends k-means clustering to fuzzy sets. In essence, FCM works the same as k-means as described above, with an extra membership parameter to weigh the WCSS for each datum in each cluster. This membership in turn depends on a 'fuzziness' parameter. If the latter is set to 1, each membership weight converges to either 0 or 1, yielding classic *hard* k-means results [109].

Expectation-Maximization

Another approach to identify mixtures of clusters, is by means of the *Expectation-Maximization* (EM) algorithm [51]. EM is a density-based method: given a data set $\mathbf{X}$, it attempts to fit a defined k number of specific distributions over the data, allowing for the computation of relative probabilities of an observation to 'belong' to each cluster. A common application is to model mixtures of Gaussians, thereby estimating the latent mixing parameter $\boldsymbol{\tau}$ which specifies the aforementioned relative mixing probabilities. This algorithm also works iteratively until it converges:

(1) In the E-step, each datum is assigned a relative probability estimate for all distributions;
(2) In the M-step, the distribution parameter estimations are updated based on the new soft cluster assignments.

As starting parameters for $\{\hat{\boldsymbol{\mu}}_1, ..., \hat{\boldsymbol{\mu}}_k\}$, usually random data points are picked. The starting variances $\left\{\hat{\boldsymbol{\sigma}}_1^2, ..., \hat{\boldsymbol{\sigma}}_k^2\right\}$ could be set to the overall variance in the data and the mixture parameter $\boldsymbol{\tau}$ is often initialized equally over the clusters: $\frac{1}{k}$.

From this display of the two-step core EM workings, the similarity of its usage in data clustering to that of k-means is clear. One could imagine that in EM clustering, the distributions – instead of just the centroids as in k-means clustering – move iteratively through the feature space. In a sense, the described application of EM for cluster analysis could be viewed as a probabilistic version of k-means clustering. When the variance parameters σ^2 of the fitting densities are restricted to 0, the approach becomes a k-means algorithm.

Soft Clustering not Suitable for this Work

Soft clustering techniques appear optimal for modelling mixtures of clusters, which are without doubt present in the current data. However, any fuzzy clustering algorithm requires sufficient data representing the "pure" clusters, and not too much data in between, in order to recognize them. This is a vital component of the approach and unfortunately, it cannot hold for the data at hand. As mentioned before, here we deal with observations that are most probably all mixtures in themselves. When a vast portion of data lies in between what are actually "pure" clusters, any clustering algorithm will recognize such a mixture as a cluster. A potential solution would be granted by the usage of reference clusters, as mentioned in the Introduction. However, this again requires assumptions on microphysical profile and number of existing aerosol types, as well as their prevalence. This was something we attempted to circumvent in this work, as described before. Thus, soft clustering would not be of any additional value in this case. It is for this reason that soft clustering methods were not considered further for this work, but the focus remained on hard clustering by k-means and SOM clustering instead.

B Full Data Set

Short name	Long name	Wavelengths / Modes / Variants
AOT	Aerosol Optical Thickness	440, 490, 563, 670, 865, 1020 nm
SSA	Single Scattering Albedo	440, 490, 563, 670, 865, 1020 nm
reff	Effective Radius	Fine, Coarse Mode Fraction
veff	Effective Variance	Fine, Coarse Mode Fraction
m_r	Real Refractive Index	Fine, Coarse Mode Fraction
m_i	Imaginary Refractive Index	Fine, Coarse Mode Fraction
sphere_frac	Sphericity	Fine, Coarse Mode Fraction
lat	Latitude	Coordinates of Centers and Corners
long	Longitude	Coordinates of Centers and Corners
psurf	Surface Pressure	
N	Aerosol Column Number Density	Fine, Coarse Mode Fraction
number_of_points	Number of Data Points	Over Ocean, Land
error	Parameter Retrieval Uncertainty / Error	AOT, SSA, reff, veff, m_r, m_i, N, sphere_frac

C List of Main Created R Functions

`cv.clust`

DESCRIPTION
Carries out a (repeated) v-fold cross-validation procedure for cluster analysis given cluster algorithm parameters.

USAGE

```
cv.clust(data, v = 5, M = 1, cellsperdim = 10, split = c("stratified", "standard"),
split_by = NULL, clustering_method = kmeans, labels = "cluster", centroids =
"centers", val_measure = c("ps", "mvgs", "avgs", "mvs", "avs"), ...)
```

INPUT

`data`	an R data frame with numeric data.
`v`	integer indicating number of folds.
`M`	integer indicating number of global cross-validation repeats.
`cellsperdim`	integer specifying the c parameter for the gridded silhouette: number of grid cells per dimension.
`split`	fashion of data splitting into folds. Either standard (totally random), or stratified by a variable in the data frame.
`split_by`	if split is set to stratified, character string matching the colname of the variable in the data frame to stratify on.
`clustering_method`	function name of the clustering algorithm. NOTE: not as character string, merely the function name without parentheses (e.g. kmeans, not "kmeans" or kmeans()).
`labels`	name of the element containing the cluster labels in the object returned by the clustering algorithm, as character string.
`centroids`	name of the element containing the cluster centroids in the object returned by the clustering algorithm, as character string.
`val_measure`	(vector of) character string(s) indicating cluster validation index or indices to be computed. Current options are: Prediction Strength ("ps"), Minimum / Average Validated (Gridded) Silhouette ("m/av(g)s").
`...`	other arguments passed to the clustering algorithm (e.g. centers, grid etc.).

OUTPUT

A list of `M` matrices with `v` columns containing the found values of the specfied cluster validation indices per fold.

`extract.nc`

DESCRIPTION

Extraction of data from netCDF files, loading them into R. Data has to be provided in the format `filebasenameyyyymm.nc`, e.g. `SRON_parasol_gridded200601.nc`.

USAGE

```
extract.nc(filebasename, path, variables = c("SSA_490nm", "AE", "sphere_frac_coarse",
"m_r_fine", "m_r_coarse"), year = 2006, months = 1:12, format = c("array",
"dataframe"), AE_lambda1 = 490, AE_lambda2 = 670, AOT490nm_lowerlim = 0.1,
AE_lowerlim = 0, SSA_lowerlim = 0.6, fill_treshold = 9e36)
```

INPUT

`filebasename`	string with the base of the file names (the common part).
`path`	string with path to the local data files.
`variables`	names of variables to extract from the data file.
`year`	year of which to extract data (integer).
`months`	numbers of months of which to extract data (integer or vector of integers).
`format`	whether to output an object of class data.frame or array.
`AE_lambda1`	lower AOT wavelength to be used in AE extraction.
`AE_lambda2`	upper AOT wavelength to be used in AE extraction.
`AOT490nm_lowerlim`	lower bound of AOT (490 nm) values to be used in AE computation.
`AE_lowerlim`	lower bound of AE values to be included in the data.
`SSA_lowerlim`	lower bound of AE values to be included in the data.
`fill_treshold`	threshold value to distinguish fill values from measurements.

OUTPUT
A single data object of the specified class (default data.frame) and contents.

`feature.scaling`

DESCRIPTION
Perform feature scaling and backscaling to the original scale on numeric vectors.

USAGE
```
feature.scaling(x)
feature.backscaling(xnorm, minorig, maxorig)
```

INPUT

`x`	numeric vector to be scaled.
`xnorm`	scaled numeric vector to be scaled back.
`minorig`	minimum value of numeric vector on original scale.
`maxorig`	maximum value of numeric vector on original scale.

OUTPUT
Numeric vector containing the (back)scaled values.

`gg.heatmap.nc`

DESCRIPTION
Creates a ggplot-based heatmap, displaying the averages per grid box of a given variable on a world map per season.

USAGE
```
gg.heatmap.nc(data, seasons = 1:4, variable = "SSA_490nm", legend_name = "SSA",
title_name = "Single Scattering Albedo (490nm)")
```

INPUT

`data`	an R data frame with numeric data. NOTE: NA values should NOT be removed prior to invoking this function.
`seasons`	(vector of) integer(s) indicating of which seasons maps should be produced (1:4 = winter:autumn, chronologically).
`variable`	character string: name of the variable to be plotted as it appears in the data frame.
`legend_name`	character string indicating title of the legend.
`title_name`	character string indicating title of the total grid of world maps.

OUTPUT
Plot: grid of world heatmaps depicting seasonal means of the variable, one map per season.

`grid.dim`

DESCRIPTION
Finds the two integers closest to the square root of the input, whose product is the input. Thus

gives optimal grid dimensions for a given number of elements to create 2D grid as square as possible.

USAGE
`grid.dim(x)`

INPUT

`x`	integer to be split in factors.

OUTPUT
Two integers: the factors as similar as possible that still produce the input.

`mah.clust`

DESCRIPTION
Mahalanobis clustering: calculates Mahalanobis distances of data to cluster centers with respect to their variance-covariances and assigns labels to the data matching their closest centers in this metric.

USAGE
`mah.clust(data, center, sigma, verbose = TRUE)`

INPUT

`data`	an R data frame with numeric data.
`center`	matrix with cluster center coordinates in same dimensions as data.
`sigma`	array containing variance-covariance matrices; third dimension represents different cluster centers.
`verbose`	logical operator: show progress during computations (TRUE) or not (FALSE).

OUTPUT
List with two elements: `Cluster` is a vector with Mahalanobis clustering label assignments for the data; `Dist_mat` is the calculated Mahalanobis distance matrix.

`nstart.som`

DESCRIPTION
Wrapper function for the `som()` function of the R **kohonen** package. Runs the specified Self-Organizing Map *nstart* times and returnes the solution with the smallest overall within-cluster sum of squared error (WCSS).

USAGE
`nstart.som(data, nstart, verbose)`

INPUT

`data`	an R data frame or matrix with numeric data.
`nstart`	integer: the number of random initialization sets to be chosen to run the SOM with.
`verbose`	logical operator: show progress during computations (TRUE) or not (FALSE).

`...`	arguments to be passed to the `som()` function.

OUTPUT
An "enhanced.kohonen" object with added computed total WCSS `tot.wss` value of final solution. Otherwise same output as `som()` from the R **kohonen** package.

`ps`

DESCRIPTION
Computes the Prediction Strength of two clusterings on the same data.

USAGE
```
ps(labels_train, labels_valid)
```

INPUT

`labels_train`	Vector with cluster labels assigned to the validation fold data by the clustering on the train fold.
`labels_valid`	Vector with cluster labels assigned to the validation fold data by the clustering on the validation fold.

OUTPUT
Integer: Prediction Strength of given clusterings. Issues a warning if at least one cluster of size 1 occured in the direct clustering of the validation fold data. This produces NA values, which are omitted.

`silh.grid`

DESCRIPTION
Computes the Gridded Silhouette Width. Divides feature space of input data into grid cells. Calculates Silhouette Widths per cluster label between grid cells. Distances are weighted relative to data counts in the target grid cell. Approximates Silhouette Widths of data per grid cell.

USAGE
```
silh.grid(data, labels, cellsperdim)
```

INPUT

`data`	An R data frame with (scaled) numeric data.
`labels`	clustering labels of the data.
`cellsperdim`	number of desired grid cells per dimension c.

OUTPUT
List of same length as number of unique clustering labels. Each element is a matrix containing: `Weighted_gridcell_silhouette`: the silhouette of that grid cell for that label, weighted relative to the number of observations of that label in that grid cell; `Gridded_silhouette`: the silhoutte width per grid cell; `Weight`: the weight corresponding to that grid cell for that label; `CellID`: an ID index for the grid cell number in the feature space; `Count`: the number of observations of that label in that grid cell.

`SOM.onecluster.heatmap`

DESCRIPTION
Produces a heatmap superimposed on a world map depicting the prevalence of a specified cluster per season.

USAGE
```
SOM.onecluster.heatmap(data, clustvarname, cluster)
```

INPUT

`data`	an R data frame containing the clustered data, including one variable holding the assigned cluster labels. Data should also include variables "Latitude", "Longitude" and "Season".
`clustvarname`	character string of the name of the variable containing the cluster labels.
`cluster`	cluster label of which the maps are to be created.

OUTPUT
Seasonal world heatmaps displaying the number of times (counts) every grid box on the surface of the Earth was assigned to the specified cluster.

`strat.split`

DESCRIPTION
Splits given data randomly into a specified number of equally sized folds, stratified on a factor.

USAGE
```
strat.split(data, k, by, train_fraction = 0.5)
```

INPUT

`data`	an R data frame with numeric data, including the variable to be stratified on.
`k`	an integer indicating the number of folds the data should be split in.
`by`	a character string of the name of the variable to be stratified on.
`train_fraction`	if 2 folds are specified, a fraction indicating the size of the train fold relative to the total data size.

OUTPUT
Returns a list with two elements, both containing the random stratified fold division: `folds_df` and `folds_ls`.