Data Science for Biodiversity Scientists

An introduction to concepts and practices

Timothée Poisot

2023-10-23

Table of contents

Preface

Data science is now an established methodology to study biodiversity, and this is a problem.

This may be an opportunity when it comes to advancing our knowledge of biodiversity, and in particular when it comes to translating this knowledge into action (Tuia *et al.* 2022); but make no mistake, this is a problem for us, biodiversity scientists, as we suddenly need to develop competences in an entirely new field. And as luck would have it, there are easier fields to master than data science. The point of this book, therefore, is to provide an introduction to fundamental concepts in data science, from the perspective of a biodiversity scientist, by using examples corresponding to real-world use-cases of these techniques.

But what do we mean by *data science*? Most science, after all, relies on data in some capacity. What falls under the umbrella of data science is, in short, embracing in equal measure quantitative skills (mathematics, machine learning, statistics), programming, and domain expertise, in order to solve well-defined problems. A core tenet of data science is that, when using it, we seek to "deliver actionable insights", which is MBA-speak for "figuring out what to do next". One of the ways in which this occurs is by letting the data speak, after they have been, of course, properly cleaned and transformed and engineered beyond recognition. This entire process is driven by (or subject to, even) domain knowledge. There is no such thing as data science, at least not in a vacuum: there is data science as a methodology applied to a specific domain.

Before we embark into a journey of discovery on the applications of data science to biodiversity, allow me to let you in on a little secret: data *science* is a little bit of a misnomer.

To understand why, it helps to think of science (the application of the scientific method, that is) as cooking. There are general techniques one must master, and specific steps and cultural specifics, and there

is a final product. When writing this preface, I turned to my shelf of cookbooks, and picked my two favorites: Robuchon's *The Complete Robuchon* (a no-nonsense list of hundreds of recipes with no place for improvisation), and Bianco's *Pizza, Pasta, and Other Food I Like* (a short volume with very few pizza and pasta, and wonderful discussions about the importance of humility, creativity, and generosity). Data science, if it were cooking, would feel a lot like the second. Deviation from the rules (they are mostly recommendations, in fact) is often justifiable if you feel like it. But this improvisation requires good skills, a clear mental map of the problem, and a library of patterns that you can draw from.

This book will not get you here. But it will speed up the process, by framing the practice of data science as a natural way to conduct research on biodiversity.

1 Introduction

This book started as a collection of notes from several classes I gave in the Department of Biological Sciences at the Université de Montréal, as well as a few workshops I ran for the Québec Centre for Biodiversity Sciences. In teaching data synthesis, data science, and machine learning to biology students, I realized that the field was missing a stepping stone to proficiency. There are excellent manuals covering the mathematics of data science and machine learning; there are many good papers giving overviews of some applications of data science to biological problems; and there are, of course, thousands of tutorials about how to write code (some of them are good!).

But one thing that students commonly called for was an attempt to tie concepts together, and to explain when and how human decisions were required in ML approaches (Sulmont *et al.* 2019). This is this attempt.

There are, broadly speaking, two situations in which reading this book is useful. The first is when you are done reading some general books about machine learning, and want to see how it can be applied to problems that are more specific to biodiversity research; the second is when you have a working understanding of biodiversity research, and want a stepping stone into the machine learning literature. Note that there is no scenario where you *stop* after reading this book – this is by design. The purpose of this book is to give a practical overview of "how data science for biodiversity happens", and this needs to be done in parallel to even more fundamental readings.

These are examples of books I like. I found them comprehensive and engaging. They may not work for you.

A wonderful introduction to the mathematics behind machine learning can be found in Deisenroth *et al.* (2020), which provides stunning visualization of mathematical concepts. Yau (2015) is a particularly useful book about the ways to visualize data in a meaningful way. **TK**

When reading this book, I encourage you to read the chapters in order. They have been designed to be read in order, because each chapter introduces the least possible quantity of new concepts, but often requires to build on the previous chapters. This is particularly true of the second half of this book.

note on the meaning of colors

1.1 Core concepts in data science

1.1.1 EDA

1.1.2 Clustering and regression

1.1.3 Supervised and unsupervised

1.1.4 Training, testing, and validation

1.1.5 Transformations and feature engineering

1.2 An overview of the content

In Chapter **??**, we introduce some fundamental questions in data science, by working on the clustering of pixels in Landsat data. The point of this chapter is to question the way we think about data, and to start a discussion about an "optimal" model, hyper-parameters, and what a "good" model is.

In Chapter **??**, we revisit well-trodden statistical ground, by fitting a linear model to linear data, but uisng gradient descent. This provides us with an opportunity to think about what a "fitted" model is, whether it is possible to learn too much from data, and why being able to think about predictions in the unit of our problem helps.

In Chapter **??**, we start introducing one of the most important bit element of data science practice, in the form of cross-validation. We apply this technique to the prediction of plant phenology over a millenia, and think about the central question of "what kind of decision-making can we justify with a model".

In Section **??**, we discuss data leakage, where it comes from, and how to prevent it. This leads us to introducing the concept of data transformations as a model, which will establish some best practices we will keep on using throughout this book.

In Chapter **??**, we introduce the task of classification, and spend a lot of time thinking about biases in predictions, which are acceptable, and which are not. We start building a model for the distribution of the Reindeer, which we will improve over a few chapters.

In **?@sec-variable-selection**, we explore ways to perform variable selection, think of this task as being part of the training process, and introduce ideas related to dimensionality reduction. We further improve our distribution model.

In Chapter **??**, we conclude story arcs that had been initiated in a few previous chapters, and explore training curves, the tuning of hyperparameters, and moving-threshold classification. We provide the final refinements to out model of the Reindeer distribution.

In Chapter **??**, we will shift our attention from prediction to understanding, and explore techniques to quantify the importance of variables, as well as ways to visualize their contribution to the predictions. In doing so, we will introduce concepts of model interpretation and explainability.

1.3 Some rules about this book

When I started aggregating these notes, I decided on a series of four rules. No code, no simulated data, no long list of model, and above all, no `iris` dataset. In this section, I will go through *why* I decided to adopt these rules, and how it should change the way you interact with the book.

1.3.1 No code

This is, maybe, the most surprising rule, because data science *is* programming (in a sense). But sometimes there is so much focus on programming that we lose track of the other, important aspects of

the practice of data science: abstractions, relationship with data, and domain knowledge.

This book *did* involve a lot of code. Specifically, this book was written using *Julia* (Bezanson *et al.* 2017), and every figure is generated by a notebook, and they are part of the material I use when teaching from this content in the classroom. But code is *not* a universal language, and unless you are really familiar with the language, code can obfuscate. I had no intention to write a *Julia* book (or an *R* book, or a *Python* book). The point is to think about data science applied to ecological research, and I felt like it would be more inclusive to do this in a language agnostic way.

And finally, code rots. Code with more dependencies rots faster. It take a single change in the API of a package to break the examples, and then you are left with a very expensive monitor stand. With a few exceptions, the examples in this book do not use complicated packages either.

1.3.2 No simulated data

I have nothing against simulated data. I have, in fact, generated simulated data in many different contexts, for training or for research. But the limit of simulated is that we almost inevitably fail to include what makes real data challenging: noise, incomplete or uneven sampling, data representation artifacts. And so when it is time to work on real data, everything seems suddenly more difficult.

Simulated data have *immense* training value; but it is also important to engage with the imperfect actual data, as we will overwhelmingly apply the concepts from this book to them. For this reason, there are no simulated data in this book. Everything that is presented correspond to an actual use case that proceeds from a question we could reasonably ask in the context, paired with a dataset that could be used to answer this question.

1.3.3 No model zoo

My favorite machine learning package is *MLJ* (Blaom *et al.* 2020). When given a table of labels and a table of features, it will give back a series of models that match with these data. It speeds up the discovery

of models considerably, and is generally a lot more informative than trying to read from a list of possible techniques. If I have questions about an algorithm from this list, I can start reading more documentation about how it works.

Reading a long enumeration of things is boring; unless it's sung by Yakko Warner, I'm not interested, and I refuse to inflict it on people. But more importantly, these enumerations of models often distract from thinking about the problem we want to solve in more abstract terms. I rarely wake up in the morning and think "oh boy I can't wait to train a SVM today"; chances are, my thought process will be closer to "I need to tell the mushroom people where I think the next good foraging locations will be". The rest, is implementation details.

In fact, 90% of this book uses only two models: linear regression, and the Naïve Bayes Classifier. Some other models are involved in a few chapters, but these two models are breathtakingly simple, work surprisingly well, run fast, and can be tweaked to allow us to build deep intuitions about how machines learn. They are perfect for the classroom, and give us the freedom to spent most of our time thinking about how we interact with models, and why, and how we make methodological decisions.

1.3.4 No `iris` dataset

From a teaching point of view, the `iris` dataset is like hearing Smash Mouth in a movie trailer, in that it tells you two things with absolute certainty. First, that you are indeed watching a movie trailer. Second, that you could be watching Shrek instead. There are datasets out there that are *infinitely more* exciting to use than `iris`.

But there is a far more important reason not to use `iris`: eugenics.

Listen, we made it several hundred words in a text about quantitative techniques in life sciences without encountering a sad little man with racist ideas that academia decided to ignore because "he just contributed so much to the field, and these were different times, maybe we shouldn't be so quick to judge?". Ronald Aylmer Fisher, statistics' most racist nerd, was such a man; and there are, of course, those who want to consider the possibility that you can be outrageously racist as long as you are an outstanding scientist (Bodmer *et al.* 2021).

The `iris` dataset was first published by Fisher (1936) in the *Annals of Eugenics* (so, there's a bit of a red flag there already), and draws from several publications by Edgar Anderson, starting with Anderson (1928); Unwin & Kleinman (2021) have an interesting historiographic deep-dive into the correspondence between the two. Judging by the dates, you may think that Fisher was a product of his time. But this could not be further from the truth. Fisher was dissatisfied with his time, to the point where his contributions to statistics were done in service of his views, in order to provide the appearance of scientific rigor to his bigotry.

Fisher advocated for forced sterilization for the "defectives" (which he estimated at, oh, roughly 10% of the population), argued that not all races had equal capacity for intellectual and emotional development, and held a host of related opinions. There is no amount of contribution to science that pardon these views. Coming up with the idea of the null hypothesis does not even out lending "scientific" credibility to ideas whose logical (and historical) conclusion is genocide. That Ronald Fisher is still described as a polymath and a genius is infuriating, and we should use every alternative to his work that we have.

Thankfully, there are alternatives!

The most broadly known alternative to the `iris` dataset is `penguins`, which was collected by ecologists (Gorman *et al.* 2014), and published as a standard dataset (Horst *et al.* 2020) so that we can train students without engaging with the "legacy" of eugenicists. The `penguins` dataset is also genuinely good! The classes are not so obviously separable, there are some missing data that reflect the reality of field work, and the data about sex and spatial location have been preserved, which increases the diversity of questions we can ask. We won't use `penguins` either. It's a fine dataset, but at this point there is little that we can write around it that would be new, or exciting. But if you want to apply some of the techniques in this book? Go `penguins`.

2 Clustering

As we mentioned in the introduction, a core idea of data science is that things that look the same (in that, when described with data, they resemble one another) are likely to be the same. Although this sounds like a simplifying assumption, this can provide the basis for approaches in which we *create* groups in data that have no labels. This task is called clustering: we seek to add a *label* to each observation, in order to form groups, and the data we work from do *not* have a label that we can use to train a model. In this chapter, we will explore the k-means algorithm for clustering, and illustrate how it can be used in practice.

2.1 A digression: which birds are red?

Before diving in, it is a good idea to ponder a simple case. We can divide everything in just two categories: things with red feathers, and things without red feathers. An example of a thing with red feathers is the Northern Cardinal (*Cardinalis cardinalis*), and things without red feathers are the iMac G3, Haydn's string quartets, and of course the Northern Cardinal (*Cardinalis cardinalis*).

See, biodiversity data science is complicated, because it tends to rely on the assumption that we can categorize the natural world, and the natural world (mostly in response to natural selection) comes up with ways to be, well, diverse and hard to categorize. In the Northern Cardinal, this is shown in males having red feathers, and females having mostly brown feathers. Before moving forward, we need to consider ways to solve this issue, as this issue will come up *all the time.*

The first mistake we have made is that the scope of objects we want to classify, which we will describe as the "domain" of our classification, is much too broad: there are few legitimate applications where we will have a dataset with Northern Cardinals, iMac G3s, and Haydn's

string quartets. Picking a reasonable universe of classes would have solved our problem a little. For example, among the things that do not have red feathers are the Mourning Dove, the Kentucky Warbler, and the House Sparrow.

The second mistake that we have made is improperly defining our classes; bird species exhibit sexual dimorphism (not in an interesting way, like wrasses, but let's give them some credit for trying). Assuming that there is such a thing as *a* Northern Cardinal is not necessarily a reasonable assumption! And yet, the assumption that a single label is a valid representation of non-monomorphic populations is a surprisingly common one, with actual consequences for the performance of image classification algorithms (Luccioni & Rolnick 2023). This assumption reveals a lot about our biases: male specimens are over-represented in museum collections, for example (Cooper *et al.* 2019). In a lot of species, we would need to split the taxonomic unit into multiple groups in order to adequately describe them.

The third mistake we have made is using predictors that are too vague. The "presence of red feathers" is not a predictor that can easily discriminate between the Northen Cardinal (yes for males, sometimes for females), the House Finch (a little for males, no for females), and the Red-Winged Black Bird (a little for males, no for females). In fact, it cannot really capture the difference between red feathers for the male House Finch (head and breast) and the male Red Winged Black Bird (wings, as the name suggests).

The final mistake we have made is in assuming that "red" is relevant as a predictor. In a wonderful paper, Cooney *et al.* (2022) have converted the color of birds into a bird-relevant colorimetric space, revealing a clear latitudinal trend in the ways bird colors, as perceived by other birds, are distributed. This analysis, incidentally, splits all species into males and females. The use of a color space that accounts for the way colors are perceived is a fantastic example of why data science puts domain knowledge front and center.

Deciding which variables are going to be accounted for, how the labels will be defined, and what is considered to be within or outside the scope of the classification problem is *difficult*. It requires domain knowledge (you must know a few things about birds in order to establish criteria to classify birds), and knowledge of how the classification methods operate (in order to have just the right amount of

overlap between features in order to provide meaningful estimates of distance).

2.2 The problem: classifying pixels from an image

Throughout this chapter, we will work on a single image – we may initially balk at the idea that an image is data, but it is! Specifically, an image is a series of instances (the pixels), each described by their position in a multidimensional colorimetric space. Greyscale images have one dimension, and images in color will have three: their red, green, and blue channels. Not only are images data, this specific dataset is going to be far larger than many of the datasets we will work on in practice: the number of pixels we work with is given by the product of the width, height, and depth of the image!

In fact, we are going to use an image with many dimensions: the data in this chapter are coming from a Landsat 9 scene (Vermote *et al.* 2016), for which we have access to 9 different bands.

Table 2.1: Overview of the bands in a Landsat 9 scene. The data from this chapter were downloaded from LandsatLook.

Band	Measure	Notes
1	Aerosol	Good proxy for Chl. in oceans
2	Visible blue	
3	Visible green	
4	Visible red	
5	Near-infrared (NIR)	Reflected by healthy plants
6, 7	Short wavelength IR (SWIR 1)	Good at differentiating wet earth and dry earth
8	Panchromatic	High-resolution monochrome
9	Cirrus band	Can pick up high and thin clouds
10, 11	Thermal infrared	

By using the data present in the channels, we can reconstruct an approximation of what the landscape looked like (by using the red,

Figure 2.1: The Landsat 9 data are combined into the "Natural Color" image, in which the red, green, and blue bands are mapped to their respective channels (left). The other composites is a 6-5-4 image meant to show differences between urban areas, vegetations, and crops. Note that the true-color composite is slightly distorted compared to the colors of the landscape we expect; this is because natural colors are difficult to reproduce accurately.

green, and blue channels).

Or can we?

If we were to invent a time machine, and go stand directly under Landsat 9 at the exact center of this scene, and look around, what would we see? We would see colors, and they would admit a representation as a three-dimensional vector of red, green, and blue. But we would see so much more than that! And even if we were to stand within a pixel, we would see a *lot* of colors. And texture. And depth. We would see something entirely different from this map; and we would be able to draw a lot more inferences about our surroundings than what is possible by knowing the average color of a 30x30 meters pixel. But just like we can get more information that Landsat 9, so too can Landsat 9 out-sense us when it comes to getting information. In the same way that we can extract a natural color composite out of the different channels, we can extract a fake color one to highlight differences in the landscape.

In Figure **??**, we compare the natural color reconstruction (top) to a

false color composite. All of the panels in Figure **??** represent the same physical place at the same moment in time; but through them, we are looking at this place with very different purposes. This is not an idle observation, but a core notion in data science: *what we measure defines what we can see*. In order to tell something ecologically meaningful about this place, we need to look at it in the "right" way. Of course, although remote sensing offers a promising way to collect data for biodiversity monitoring at scale (Gonzalez *et al.* 2023), there is no guarantee that it will be the right approach for all problems. More (fancier) data is not necessarily right for all problems.

So far, we have looked at this area by combining the raw data. Depending on the question we have in mind, they may not be the *right* data. In fact, they may not hold information that is relevant to our question *at all*; or worse, they can hold more noise than signal. The area we will work on in this chapter is a very small crop of a Landsat 9 scene, taken on path 14 and row 28, early in late June 2023. It shows the western tip of the island of Montréal, as well as Lake Saint-Louis to the south (not actually a lake), Lake Deux-Montages to the north (not actually a lake either), and a small part of Oka national park. This is an interesting area because it has a high variety of environments: large bodies of water, forested areas (bright green in the composite), densely urbanized places (bright purple and white in the composite), less densely urbanized (green-brown), and cropland to the western tip of the island.

We will revisit the issue of variable selection and feature engineering in **?@sec-variable-selection**.

But can we classify these different environments starting in an ecologically relevant way? Based on our knowledge of plants, we can start thinking about this question in a different way. Specifically, "can we guess that a pixel contains plants?", and "can we guess at how much water there is in a pixel?". Thankfully, ecologists, whose hobbies include (i) guesswork and (ii) plants, have ways to answer these questions rather accurately.

One way to do this is to calculate the normalized difference vegetation index, or NDVI (Kennedy & Burbach 2020). NDVI is derived from the band data (NIR - Red), and there is an adequate heuristic using it to make a difference between vegetation, barren soil, and water. Because plants are immediately tied to water, we can also consider the NDWI (water; Green - NIR) and NDMI (moisture; NIR - SWIR1) dimensions: taken together, these information will represent every pixel in a three-dimensional space, telling us whether there are plants

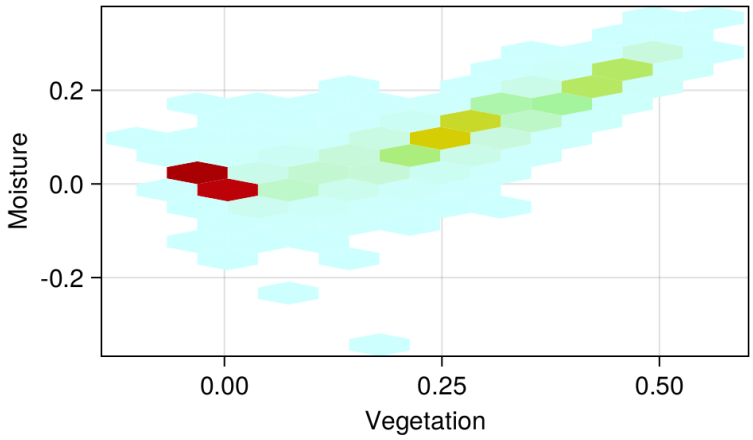


Figure 2.2: The pixels acquired from Landsat 9 exist in a space with many different dimensions (one for each band). Because we are interested in a landscape classification based on water and vegetation data, we use the NDVI, NDMI, and NDWI combinations of bands. These are *derived* data, and represent the creation of new features from the raw data. Darker colors indicate more pixels in this bin.

(NDVI), whether they are stressed (NDMI), and whether this pixel is a water body (NDWI). Other commonly used indices based on Landsat 9 data include the NBR (Normalized Burned Ratio), for which high values are suggestive of a history of intense fire (Roy *et al.* 2006 have challenged the idea that this measure is relevant immediately post-fire), and the NDBI (Normalized Difference Built-up Index) for urban areas.

We can look at the relationship between the NDVI and NDMI data Figure **??**. For example, NDMI values around -0.1 are low-canopy cover with low water stress; NDVI values from 0.2 to 0.5 are good candidates for moderately dense crops. Notice that there is a strong (linear) relationship between NDVI and NDMI. Indeed, none of these indices are really independent; this implies that they are likely to be more informative taken together than when looking at them one at a time (Zheng *et al.* 2021). Indeed, urban area tend to have high values of the NDWI, which makes the specific task of looking for swimming pools (for mosquito control) more challenging than it sounds (McFeeters 2013).

By picking these four transformed values, instead of simply looking at the clustering of all the bands in the raw data, we are starting to refine what the algorithm sees, through the lens of what we know is important about the system. With these data in hands, we can start building a classification algorithm.

2.3 The theory behind *k*-means clustering

In order to understand the theory underlying *k*-means, we will work backwards from its output. As a method for clustering, *k*-means will return a vector of *class memberships*, which is to say, a list that maps each observation (pixel, in our case) to a class (tentatively, a cohesive landscape unit). What this means is that *k*-means is a transformation, taking as its input a vector with three dimensions (NDVI, NDMI, NDWI), and returning a scalar (an integer, even!), giving the class to which this pixel belongs. Pixels only belongs to one class. These are the input and output of our blackbox, and now we can start figuring out its internals.

2.3.1 Inputs and parameters

In *k*-means, a set of observations $\mathbf{x}_i$ are assigned to a set of classes $\mathbf{C}$, also called the clusters. All $\mathbf{x}_i$ are vectors with the same dimension (we will call it f, for *features*), and we can think of our observations as a matrix of features $\mathbf{X}$ of size (f, n), with f features and n observations (the columns of this matrix).

Throughout this book, we will use $\mathbf{X}$ to note the matrix of features, and $\mathbf{y}$ to note the vector of labels. Instances are columns of the features matrix, noted $\mathbf{x}_i$.

The number of classes of $\mathbf{C}$ is $|\mathbf{C}| = k$, and k is an hyper-parameter of the model, as it needs to fixed before we start running the algorithm. Each class is defined by its centroid, a vector $\mathbf{c}$ with f dimensions (*i.e.* the centroid corresponds to a potential "idealized" observation of this class in the space of the features), which *k*-means progressively refines.

2.3.2 Assigning instances to classes

Instances are assigned to the class for which the distance between themselves and the centroid of this class is lower than the distance between themselves and the centroid of any other class. To phrase it differently, the class membership of an instance $\mathbf{x}_i$ is given by

Of course, the correct distance measure to use depends on what is appropriate for the data!

$$\text{argmin}_j \left\| \mathbf{x}_i - \mathbf{c}_j \right\|_2 , \tag{2.1}$$

which is the value of j that minimizes the L^2 norm ($\| \cdot \|_2$, the Euclidean distance) between the instance and the centroid; argmin_j is

the function returning the value of j that minimizes its argument. For example, $\text{argmin}(0.2, 0.8, 0.0)$ is 3, as the third argument is the smallest. There exists an argmax function, which works in the same way.

2.3.3 Optimizing the centroids

Of course, what we really care about is the assignment of *all* instances to the classes. For this reason, the configuration (the disposition of the centroids) that solves our specific problem is the one that leads to the lowest possible variance within the clusters. As it turns out, it is not that difficult to go from Equation **??** to a solution for the entire problem: we simply have to sum over all points!

This leads to a measure of the variance, which we want to minimize, expressed as

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathbf{C}_i} \|\mathbf{x} - \mathbf{c}_i\|_2 \,. \tag{2.2}$$

The part that is non-trivial is now to decide on the value of $\mathbf{c}$ for each class. This is the heart of the k-means algorithm. From Equation **??**, we have a criteria to decide to which class each instance belongs. Of course, there is nothing that prevents us from using this in the opposite direction, to define the instance by the points that form it! In this approach, the membership of class $\mathbf{C}_j$ is the list of points that satisfy the condition in Equation **??**. But there is no guarantee that the *current* position of $\mathbf{c}_j$ in the middle of all of these points is optimal, *i.e.* that it minimizes the within-class variance.

This is easily achieved, however. To ensure that this is the case, we can re-define the value of $\mathbf{c}_j$ as

$$\mathbf{c}_j = \frac{1}{|\mathbf{C}_j|} \sum \mathbf{C}_j \,, \tag{2.3}$$

where $|\cdot|$ is the cardinality of (number of istances in) $\mathbf{C}_j$, and $\sum \mathbf{C}_j$ is the sum of each feature in $\mathbf{C}_j$. To put it plainly: we update the centroid of $\mathbf{C}_j$ so that it takes, for each feature, the average value of all the instances that form $\mathbf{C}_j$.

2.3.4 Updating the classes

ting a step multiple times in a row
d an iterative process, and we will
see a *lot* of them.

Once we have applied Equation **??** to all classes, there is a good chance that we have moved the centroids in a way that moved them away from some of the points, and closer to others: the membership of the instances has likely changed. Therefore, we need to re-start the process again, in an iterative way.

But until when?

Finding the optimal solution for a set of points is an NP-hard problem (Aloise *et al.* 2009), which means that we will need to rely on a little bit of luck, or a whole lot of time. The simplest way to deal with iterative processes is to let them run for a long time, as after a little while they should converge onto an optimum (here, a set of centroids for which the variance is as good as it gets), and hope that this optimum is *global* and not *local*.

A global optimum is easy to define: it is the state of the solution that gives the best possible result. For this specific problem, a global optimum means that there are no other combinations of centroids that give a lower variance. A local optimum is a little bit more subtle: it means that we have found a combination of centroids that we cannot improve without first making the variance worse. Because the algorithm as we have introduced it in the previous sections is *greedy*, in that it makes the moves that give the best short-term improvement, it will not provide a solution that temporarily makes the variance higher, and therefore is susceptible to being trapped in a local optimum.

In order to get the best possible solution, it is therefore common to run k-means multiple times for a given k, and to pick the positions of the centroids that give the best overall fit.

2.3.5 Identification of the optimal number of clusters

One question that is left un-answered is the value of k. How do we decide on the number of clusters?

There are two solutions here. One is to have an *a priori* knowledge of the number of classes. For example, if the purpose of clustering is to create groups for some specific task, there might be an upper/lower bound to the number of tasks you are willing to consider. The other

solution is to run the algorithm in a way that optimizes the number of clusters for us.

This second solution turns out to be rather simple with k-means. We need to change the value of k, run it on the same dataset several times, and then pick the solution that was *optimal*. But this is not trivial. Simply using Equation **??** would lead to always preferring many clusters. After all, each point in its own cluster would get a pretty low variance!

For this reason, we use measures of optimality that are a little more refined. One of them is the Davies & Bouldin (1979) method, which is built around a simple idea: an assignment of instances to clusters is good if the instances within a cluster are not too far away from the centroids, and the centroids are as far away from one another as possible.

The Davies-Bouldin measure is striking in its simplicity. From a series of points and their assigned clusters, we only need to compute two things. The first is a vector $\mathbf{s}$, which holds the average distance between the points and their centroids (this is the $\left\|\mathbf{x}_i - \mathbf{c}_j\right\|_2$ term in Equation **??**, so this measure still relates directly to the variance); the second is a matrix $\mathbf{M}$, which measures the distances *between* the centroids.

These two information are combined in a matrix $\mathbf{R}$, wherein $\mathbf{R}_{ij} = (s_i + s_j)/\mathbf{M}_{ij}$. The interpretation of this term is quite simply: is the average distance *within* clusters i and j much larger compared to the distance *between* these clusters. This is, in a sense, a measure of the stress that these two clusters impose on the entire system. In order to turn this matrix into a single value, we calculate the maximum value (ignoring the diagonal!) for each row: this is a measure of the *maximal* amount of stress in which a cluster is involved. By averaging these values across all clusters, we have a measure of the quality of the assignment, that can be compared for multiple values of k.

Note that this approach protects us against the each-point-in-its-cluster situation: in this scenario, the distance between clusters would decrease really rapidly, meaning that the values in $\mathbf{R}$ would *increase*; the Davies-Bouldin measure indicates a better clustering when the values are *lower*.

There are alternatives to this method, including silhouettes (Rousseeuw 1987) and the technique of Dunn (1974). The question

In fact, there is very little enumeration of techniques in this book. The important point is to understand how all of the pieces fit together, not to make a census of all possible pieces.

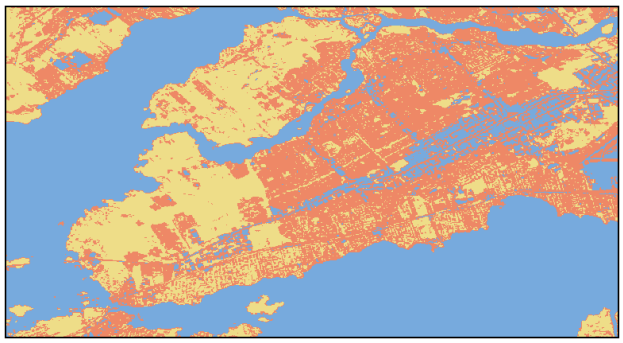


Figure 2.3: caption

of optimizing the number of clusters goes back several decades (Thorndike 1953), and it still actively studied. What matter is less to give a comprehensive overview of all the measures: the message here is to pick one that works (and can be justified) for your specific problem!

2.4 Application: optimal clustering of the satellite image data

2.4.1 Initial run

Before we do anything else, we need to run our algorithm with a random pick of hyper-parameters, in order to get a sense of how hard the task ahead is. In this case, using $k = 3$, we get the results presented in Figure **??**.

> After iterating the *k*-means algorithm, we obtain a classification for every pixel in the landscape. This classification is based on the values of NDVI, NDMI, and NDWI indices, and therefore groups pixels based on specific assumptions about vegetation and stress. This clustering was produced using $k = 3$, *i.e.* we want to see what the landscape would look like when divided into three categories.

It is always a good idea to look at the first results and state the obvious. Here, for example, we can say that water is easy to identify. In

In fact, take some time to think about how you would use k-means to come up with a way to remove pixels with only water from this image!

fact, removing open water pixels from images is an interesting image analysis challenge (Mondejar & Tongco 2019), and because we used an index that specifically identifies water bodies (NDWI), it is not surprising that there is an entire cluster that seems to be associated with water. But if we take a better look, it appears that there groups of pixels that represent dense urban areas that are classified with the water pixels. When looking at the landscape in a space with three dimensions, it looks like separating densely built-up environment and water is difficult.

This might seem like an idle observation, but this is not the case! It means that when working on vegetation-related questions, we will likely need at least one cluster for water, and one cluster for built-up areas. This is helpful information, because we can already think about how many classes of vegetation we are willing to accept, and add (at least) two clusters to capture other types of cover.

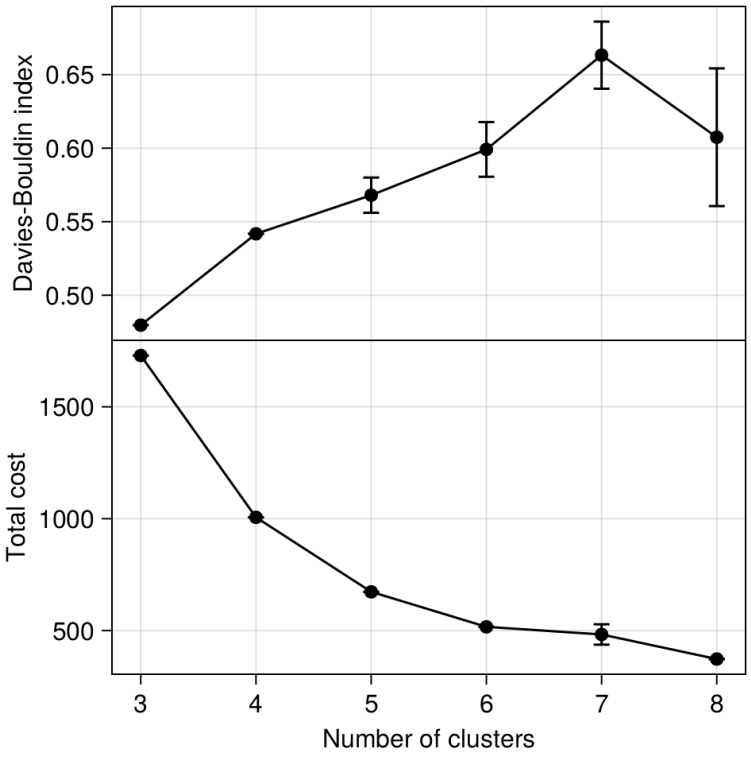
2.4.2 Optimal number of pixels

We will revisit the issue of tuning the hyper-parameters in more depth in Chapter **??**.

In order to produce Figure **??**, we had to guess at a number of classes we wanted to split the landscape into. This introduces two important steps in coming up with a model: starting with initial parameters in order to iterate rapidly, and then refining these parameters to deliver a model that is fit for purpose. Our discussion in Section **??**, where we concluded that we needed to keep (maybe) two classes for water and built-up is not really satisfying, as we do not yet have a benchmark to evaluate the correct value of k; we know that it is more than 3, but how much more?

We will now change the values of k and use the Davies & Bouldin (1979) measure introduced in Section **??** to identify the optimal value of k. The results are presented in Figure **??**. Note that we only explore $k \in [3, 10]$. More than 8 categories is probably not very actionable, and therefore we can make the decision to only look at this range of parameters. Sometimes (always!) the best solution is the one that gets your job done.

There are two interesting things in Figure **??**. First, note that for $k = \{3, 4\}$, there is almost no dispersal: all of the assignments have the exact same score, which is unlikely to happen except if the assignments are the same every time! This is a good sign, and, anecdotally,



2.4: Results of running the k-algorithm ten times for each number clusters between 3 and 8. The average vies-Bouldin and cost are depicted, as the standard deviation. As ex- the total cost decreases with more , but this is not necessarily the sign ter clustering.

something that might suggest a really information separation of the points. Second, $k = 3$ has by far the lowest Davies-Bouldin index of all values we tried, and is therefore strongly suggestive of an optimal hyper-parameter. But in Figure **??**, we already established that one of these clusters was capturing *both* water and built-up environments, so although it may look better from a quantitative point of view, it is not an ideal solution *for the specific problem we have*.

In this specific case, it makers very little sense *not* to use $k = 4$ or $k = 5$. They have about the same performance, but this gives us potentially more classes that are neither water nor built-up. This image is one of many cases where it is acceptable to sacrifice a little bit of optimality in order to present more actionable information. Based on the results in this section, we will pick the largest possible k that does not lead to a drop in performance, which in our case is $k = 5$.

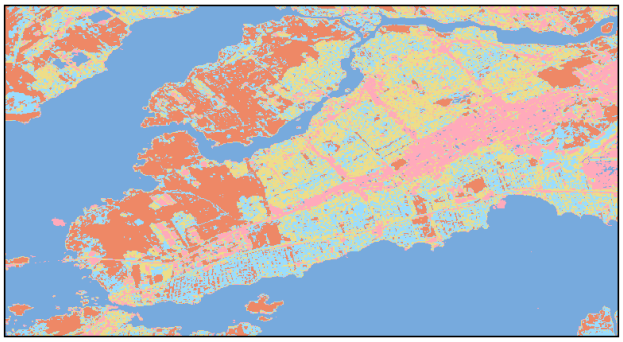


Figure 2.5: Results of the landscape clustering with k=5 clusters. This number of clusters gives us a good separation between different groups of pixels, and seems to capture features of the landscape as revealed with the false-color composites.

Table 2.2: Summary of the values for the centers of the optimal clusters found in this image. The cover column gives the percentage of all pixels associated to this class. The clusters are sorted by the NDVI of their centroid.

Cluster	Cover	NDVI	NDWI	NDMI
1	38	-0.018	0.012	0.006
4	10	0.096	-0.152	0.005
3	19	0.224	-0.262	0.08
5	17	0.32	-0.343	0.139
2	16	0.439	-0.443	0.223

2.4.3 Clustering with optimal number of classes

The clustering of pixels using $k = 5$ is presented in Figure **??**. Unsurprisingly, *k*-means separated the open water pixels, the dense urban areas, as well as the more forested/green areas. Now is a good idea to start thinking about what is representative of these clusters: one is associated with very high NDWI value (these are the water pixels), and two classes have both high NDVI and high NDMI (suggesting different categories of vegetation).

```
Warning: The clustering cost increased at iteration #3
@ Clustering ~/.julia/packages/Clustering/wNDPu/src/
```

We will revisit the issue of understanding how a model makes a prediction in Chapter **??**.

The relative size of the clusters (as well as the position of their centroids) is presented in Table **??**. There is a good difference in the size of the clusters, which is an important thing to note. Indeed, a common myth about *k*-means is that it gives clusters of the same size. This "size" does not refer to the cardinality of the clusters, but to the volume that they cover in the space of the parameters. If an area of the space of parameters is more densely packed with instances, the cluster covering the area will have more points!

In fact, this behavior makes *k*-means excellent at creating color palettes from images! Cases in point, Karthik Ram's Wes Anderson palettes, and David

The area of the space of parameters covered by each cluster in represented in Figure **??**, and this result is actually not surprising, if we spend some time thinking about how *k*-means work. Because our criteria to assign a point to a cluster is based on the being closest

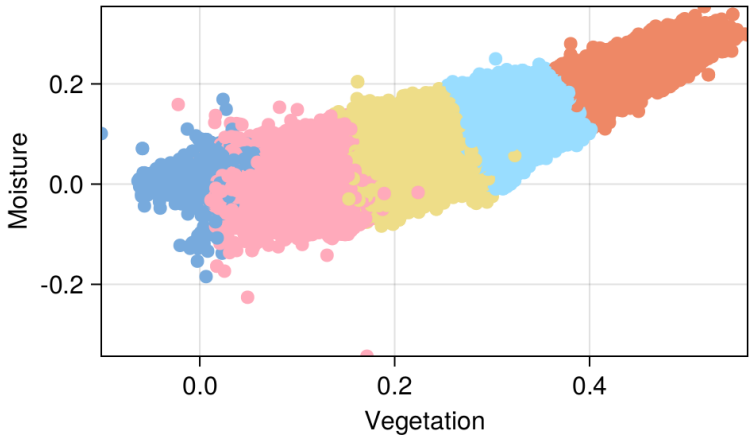


Figure 2.6: Visualisation of the clustering output as a function of the NDVI and NDMI values. Note that the limits between the clusters are lines (planes), and that each cluster covers about the same volume in the space of parameters.

to its centroid than to any other centroid, we are essentially creating Voronoi cells, with linear boundaries between them.

By opposition to a model based on, for example, mixtures of Gaussians, the assignment of a point to a cluster in k-means is independent of the current composition of the cluster (modulo the fact that the current composition of the cluster is used to update the centroids). In fact, this makes k-means closer to (or at least most efficient as) a method for quantization (Gray 1984).

2.5 Conclusion

In this chapter, we have used the k-means algorithm to create groups in a large dataset that had no labels, *i.e.* the points were not assigned to a class. By picking the features we wanted to cluster the point, we were able to highlight specific aspects of the landscape. In Chapter **??**, we will start adding labels to our data, and shift our attention from classification to regression problems.

3 Gradient descent

As we progress into this book, the process of delivering a trained model is going to become more and more complex. In Chapter **??**, we worked with a model that did not really require training (but did require to pick the best hyper-parameter). In this chapter, we will only increase complexity very slightly, by considering how we can train a model when we have a reference dataset to compare to.

Doing do will require to introduce several new concepts, and so the "correct" way to read this chapter is to focus on the high-level process. The problem we will try to solve (which is introduced in Section **??**) is very simple; in fact, the empirical data looks more fake than many simulated datasets!

3.1 A digression: what is a trained model?

Models are data. When a model is trained, it represents a series of measurements (its parameters), taken on a representation of the natural world (the training data), through a specific instrument (the model itself, see *e.g.* Morrison & Morgan 1999). A trained model is, therefore, capturing our understanding of a specific situation we encountered. We need to be very precise when defining what, exactly, a model describes. In fact, we need to take a step back and try to figure out where the model stops.

As we will see in this chapter, then in Chapter **??**, and finally in Chapter **??**, the fact of training a model means that there is a back and forth between the algorithm we train, the data we use for training, and the criteria we set to define the performance of the trained model. The algorithm bound to its dataset is the *machine* we train in machine learning.

Therefore, a trained model is never independent from its training data: they describe the scope of the problem we want to address with this

model. In Chapter **??**, we ended up with a machine (the trained k-means algorithm) whose parameters (the centroids of the classes) made sense in the specific context of the training data we used; applied to a different dataset, there are no guarantees that our model would deliver useful information.

For the purpose of this book, we will consider that a model is trained when we have defined the algorithm, the data, the measure through which we will evaluate the model performance, and then measured the performance on a dataset built specifically for this task. All of these elements are important, as they give us the possibility to *explain* how we came up with the model, and therefore, how we made the predictions. This is different from reasoning about why the model is making a specific prediction (we will discuss this in Chapter **??**), and is more related to explaining the process, the "outer core" of the model. As you read this chapter, pay attention to these elements: what algorithm are we using, on what data, how do we measure its performance, and how well does it perform?

3.2 The problem: how many interactions in a food web?

One of the earliest observation that ecologists made about food webs is that when there are more species, there are more interactions. A remarkably insightful crowd, food web ecologists. Nevertheless, it turns out that this apparently simple question had received a few different answers over the years.

The initial model was proposed by Cohen & Briand (1984): the number of interactions L scales linearly with the number of species S. After all, we can assume that when averaging over many consumers, there will be an average diversity of resources they consume, and so the number of interactions could be expressed as $L \approx b \times S$.

Not so fast, said Martinez (1992). When we start looking a food webs with more species, the increase of L with regards to S is superlinear. Thinking in ecological terms, maybe we can argue that consumers are flexible, and that instead of sampling a set number of resources, they will sample a set proportion of the number of consumer-resource