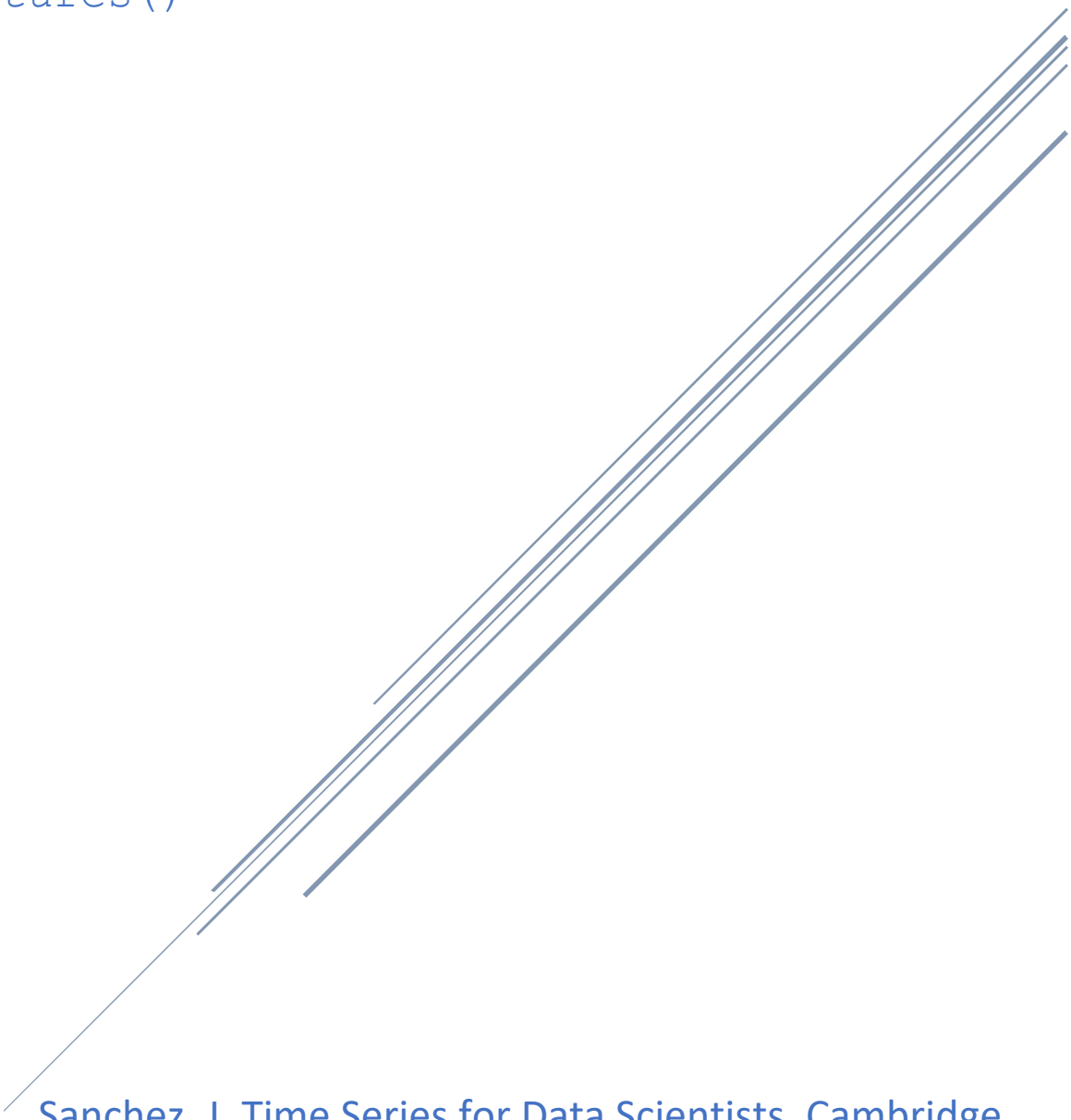
CHAPTER 10. CASE STUDY 3. BIRMINGHAM PARKING OCCUPANCY

Unsupervised Machine Learning with `summary()` and `tsfeatures()`



Sanchez, J. Time Series for Data Scientists. Cambridge University Press, 2023

In several chapters of the book, we have been calculating summary features of time series. Summary features are convenient when we have a large number of time series and we are interested in determining whether there are hidden patterns or commonalities in how the series behave. For example, if we have time series for parking spaces at several locations in a large city, we might be interested in determining whether some locations share the same temporal behavior.

In past chapters, however, we have been using very simple features familiar to anyone that has had some exposure to introductory statistics or features that were studied in the corresponding chapters where the example was presented: mean, standard deviation, number of observations at certain distance from the mean, autocorrelations up to a fixed lag k. Certainly there are many more features that we could considered as good summaries of a time series.

The *tsfeatures* package allows us to generate more features of our time series beyond just summary statistics like those seen in other chapters. We will get some practice using this package by generating some features that could be appropriate for the **Smart Cities Parking data seen in Chapter 4, Case Study of Section 4.7** (from now on "the data."). We include that data again in the space where this file is for your convenience. The R program used for doing the analysis in this case study is ***Ch10-Birmingham-auto-features.R***. Read the information about the data that we presented there. We will then perform some unsupervised machine learning by doing a clustering analysis with the features that we selected.

I. The *tsfeatures* package

In order to use the *tsfeatures* (https://cran.r-project.org/web/packages/tsfeatures/vignettes/tsfeatures.html#tsfeatures) package, one needs to install it in R by running,

```
install.packages("tsfeatures")
```

and then including it in their code along with their other packages with

```
library(tsfeatures)
```

The *tsfeatures* package has a whole number of functions to generate different types of features of the time series it is given. The input can vary depending on the function being used. For example, functions sometimes return a list of output, like `acf_features()`, or just a single value like `hurst()`. To ensure you get an output you expect, and give the correct inputs to each function, it will be useful to refer the documentation provided, as they provide a list of functions as well as examples.

With this in mind, we will now explore this data and use this package to see and include features that could be useful for this data.

I.1 Feature Selection using the *tsfeatures* package

Feature selection can be difficult to achieve as it requires a good understanding of the features themselves, knowledge of your data, and domain knowledge to have a good idea of what features might be useful in differentiating data. To get started it is always good to plot your data to get an idea of what it looks like and then one can see if any features can be used to highlight certain differences that we visually see.

Step 1. Know your data.

Within the code provided we generated Figure 1 by selecting 6 parking structures at random out of the 28 we have in our data after removing "BHMBRTARC01".

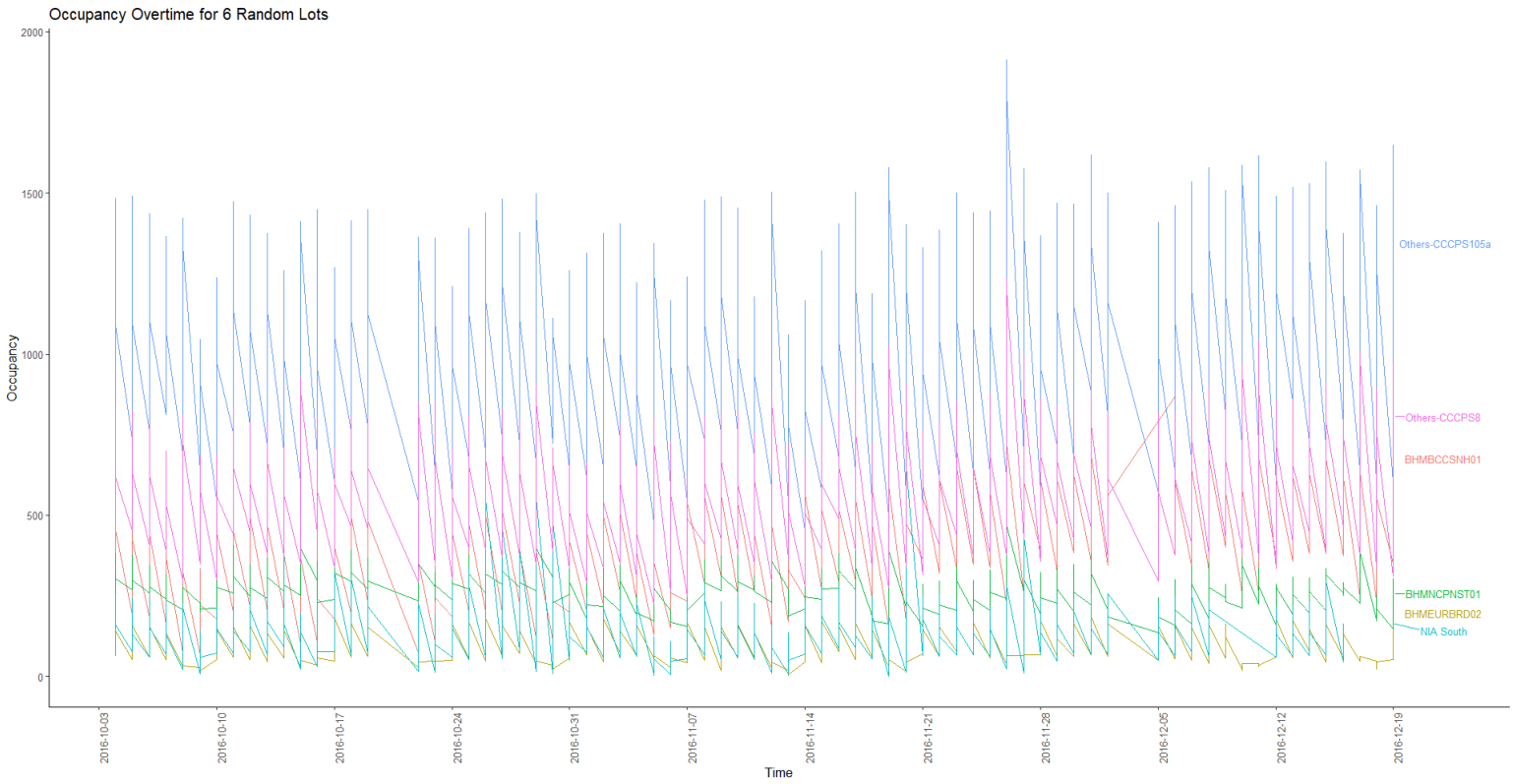


Figure 1. Multiple time plots image of occupancy to view weekly and daily seasonality.

Figure 1 helps us see the weekly variation and the daily variation of the different types of parking lots within this data; we also notice that some have a much larger occupancy than others. We can also see some strange zeros in the data when there are sudden drops in occupancy. Certain time series also have significantly lower occupancy than others, even when they are expected to be high.

The reader could modify the R code for this plot in the R file, to get the percentage

Step 2. Study the features available in the `tsfeatures` package, and decide which ones to use.

To familiarize yourself with the features that could possibly be extracted from the time series using the *`tsfeatures`* package, it would be useful to look at https://cran.r-project.org/web/packages/tsfeatures/vignettes/tsfeatures.html

In this case study, we try a few features, but the reader is encouraged to think, after reading all the possible features to include, and their description, which ones the reader would prefer. We will describe next the ones that we chose for illustration purposes.

`acf_features` and `hurst` under the function `acf1` in the R program. The former gives the first autocorrelation coefficient or the short term memory of the data, the autocorrelation coefficient of the difference and the second difference, while hurst shows the long-range dependence or self-similarity within time series, thus both can group time series with similar short term memory and self-dependency.

The data also has a lot of randomness, and features like `entropy` and `stl_features(spike)` could help capture the general random differences as entropy measures the uncertainty or randomness, while spike measures the presence and magnitude of sudden, short-duration peaks or irregular events, which both can be used to group those with similar randomness.

The variability in the capacity of the time series, those high or low points, could be captured by `crossing_points` and `flat_spots`. The former shows the number of times a time series crosses the median line, and the latter measures the maximum run length, which may help to capture time series with sequences of high or much lower occupancy.

`stl_features()` contains many other features that were included in our R program. Please, see the documentation for the package and the R program, to see what we chose. With more precise domain knowledge and understanding of one's data these features can be used to discover similarities among different time series.

The features mentioned are just an example of possible features to consider. Explore more.

II. Unsupervised Machine Learning- Clustering

Unsupervised machine learning, unlike supervised machine learning, is not given any knowledge of the actual groups/labels of the data, and instead finds the natural grouping and structure within the data. One example of unsupervised machine learning is cluster analysis, which we will use to cluster the parkings' time series based on the features generated. We will be using `kmeans()` from the base R package to perform cluster analysis.

Kmeans cluster analysis

There are two arguments given to `kmeans()`, the number of clusters, 'k', and the columns of the features data set that we woult like to use to base the clusters off of. We will only give kmeans() the columns generated from tsfeatures, but in order to determine the optimal

number of clusters we will make an *elbow plot* of how the Mean Square Error (MSE) changes for each k value, and choose the optimal number of cluster values by where the elbow is on the plot. We do this by performing kmeans a few times, with small k to a large k=10. When we plot the MSE for each k value we get Figure 2.

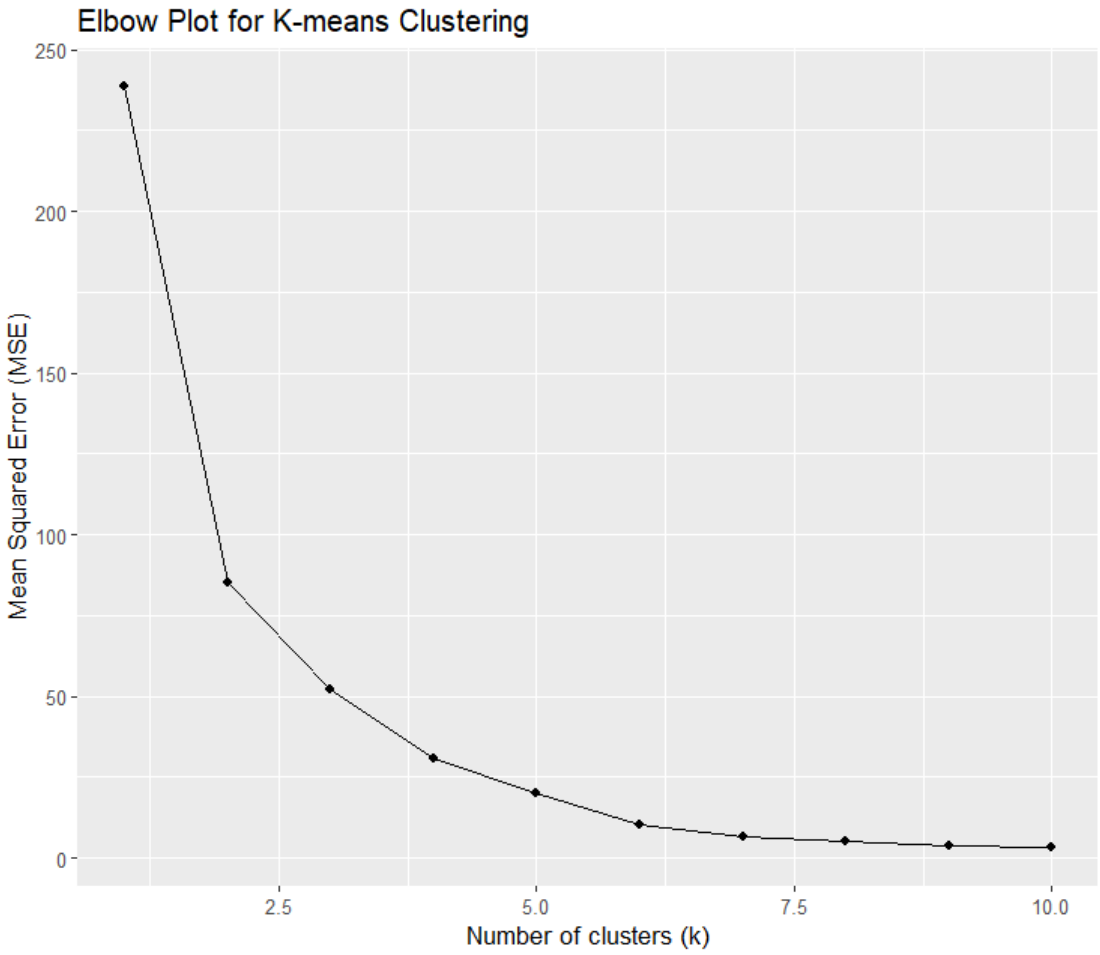


Figure 2. The elbow plot allows us to see how the MSE changes as the number of clusters considered in kmeans() changes.

From Figure 2, it seems like 3-5 clusters seem appropriate to cluster the set of time series using only the features created with the tsfeatures package. We will proceed with using a k=3.

With our chosen k, we can now apply k means again to our data using k=3, and save those clusters. We now can plot our data and color by the clusters to see if there is any natural grouping to the data. In order to plot the data in way that we can visualize the clusters, however, we need a horizontal and a vertical axis, thus we need to choose two features for the axis. We want to use the features (the dimensions) of the features data set that best help separate the clusters. That is, when choosing these features, we want those that have the most variability between the cluster groups. A pairs plot like that in Figure 3 helps us select the variables that best separate the clusters.

Figure 3. Pairs plot to visualize clusters in different dimensions. This plot lets us see the distribution of all variables per cluster (right column box plots), the proportion of parking lots in each cluster, the scatter plot of the separated clusters (colored scatter plots) and the distribution of a given variable by cluster.

Based on Figure 3, the only variable that has very clear separation between groups is the feature *crossing_points*, which we will make our x axis. For our y axis, there are not many other features with great separation between groups, thus we will choose flat spots because at least there is 2 distinct nodes. With our axis chosen, we can plot the data and color the clusters as indicated in Figure 4.

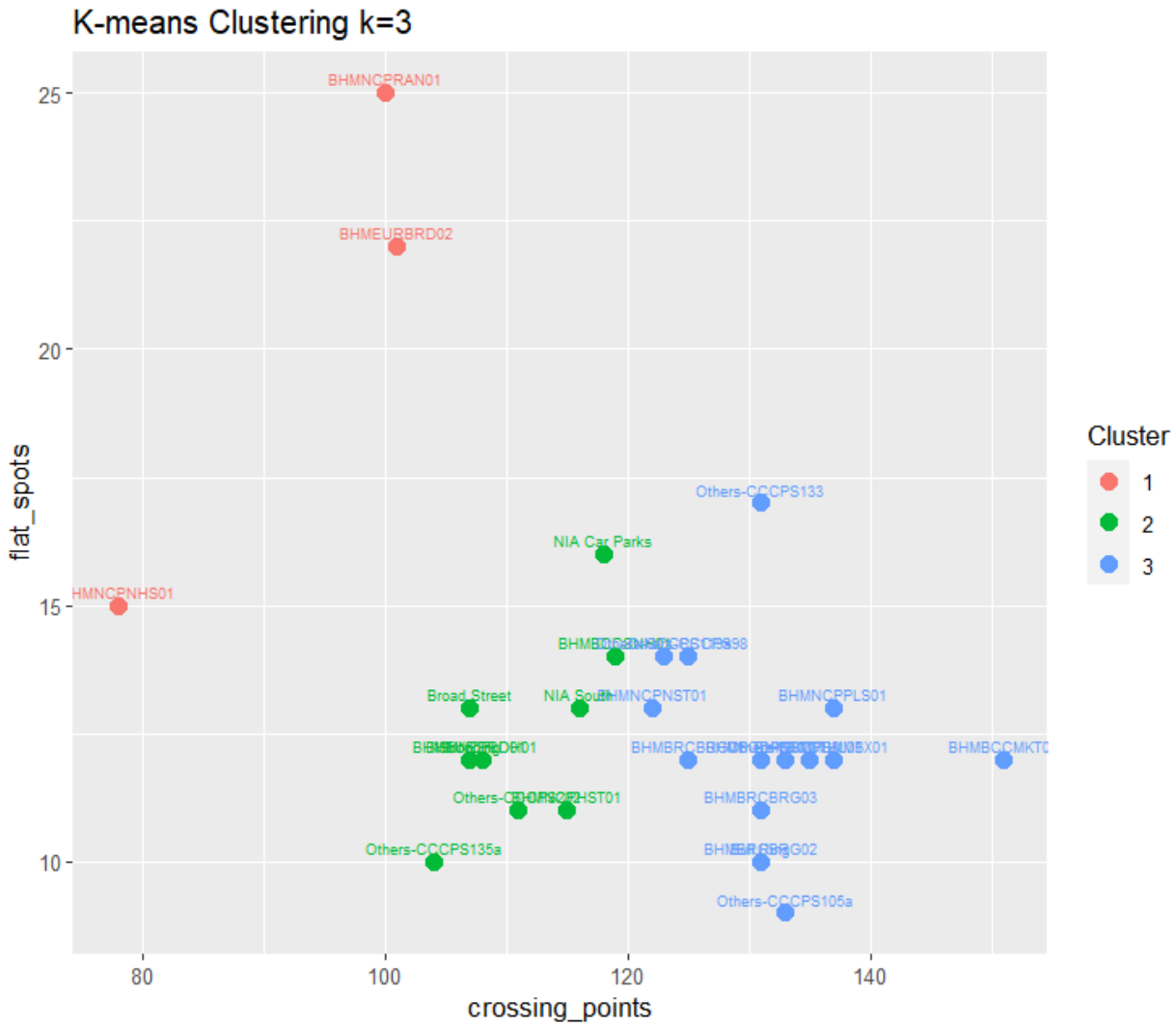


Figure 4. After selecting the two dimensions that make the time series most dissimilar, we plot the data and the cluster allocation based on those dimensions.

Figure 4 indicates that there seems to be decent clusters formed, however that is mainly due to crossing points splitting the data well. With better domain knowledge and with other features from the tsfeatures package or the summary features extracted without the package, perhaps there could be better features to include that might be able to better separate the clusters. The reader is encouraged to use the R program to try other features.