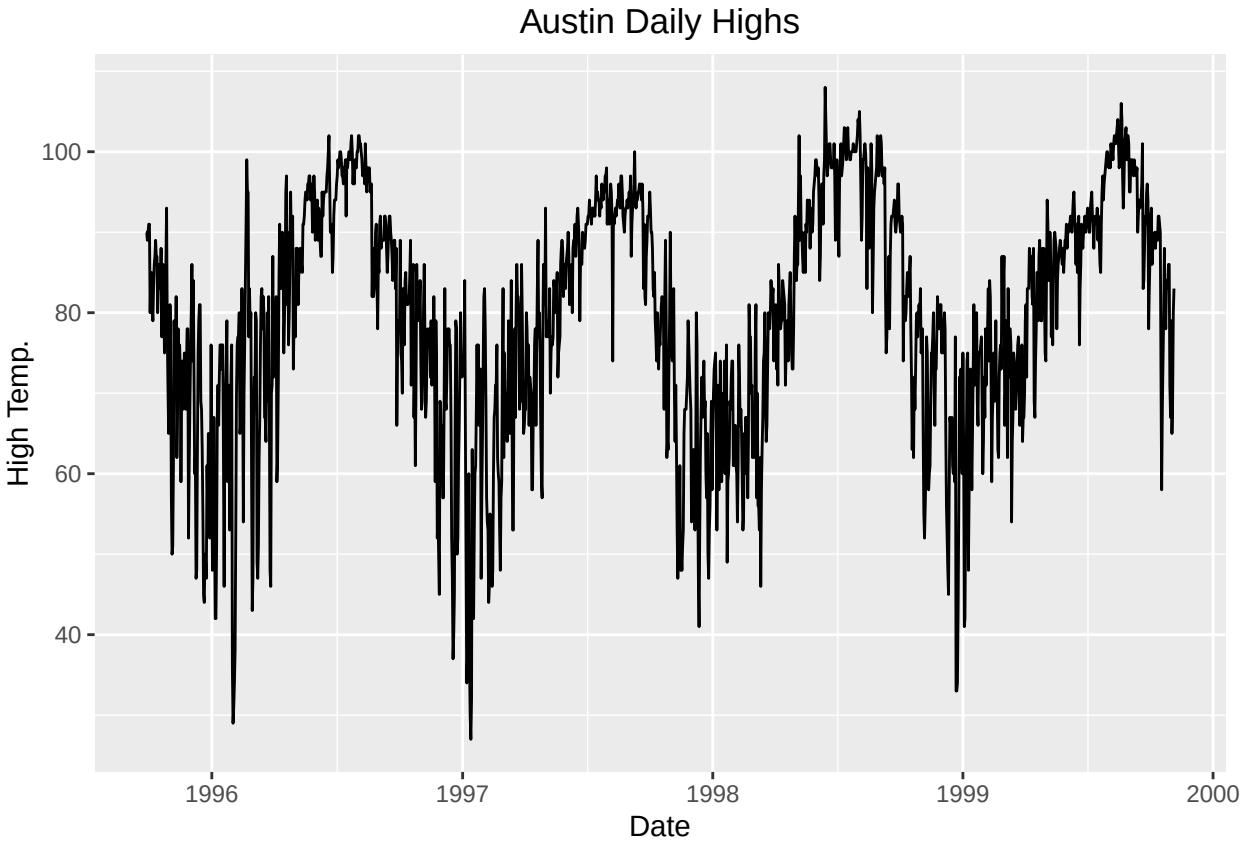
Smoothing Temperature Data

Steve Bischoff

2023-12-19

1. Introduction

Smoothing methods are a popular way to identify the trend(s) in noisy data without having to construct a model of the data-generating process. Temperature data are a particularly appealing application of these methods. Consider this graph of daily high temperatures in Austin, Texas in the late 90s: [1]



The graph is visually unappealing due to daily temperature fluctuations. This noise also makes it difficult to generalize and identify trends. As just one example: the winter of 96-97 clearly has the lowest lows of these years, but was that cold spell an aberration or did it truly have the coldest winter?

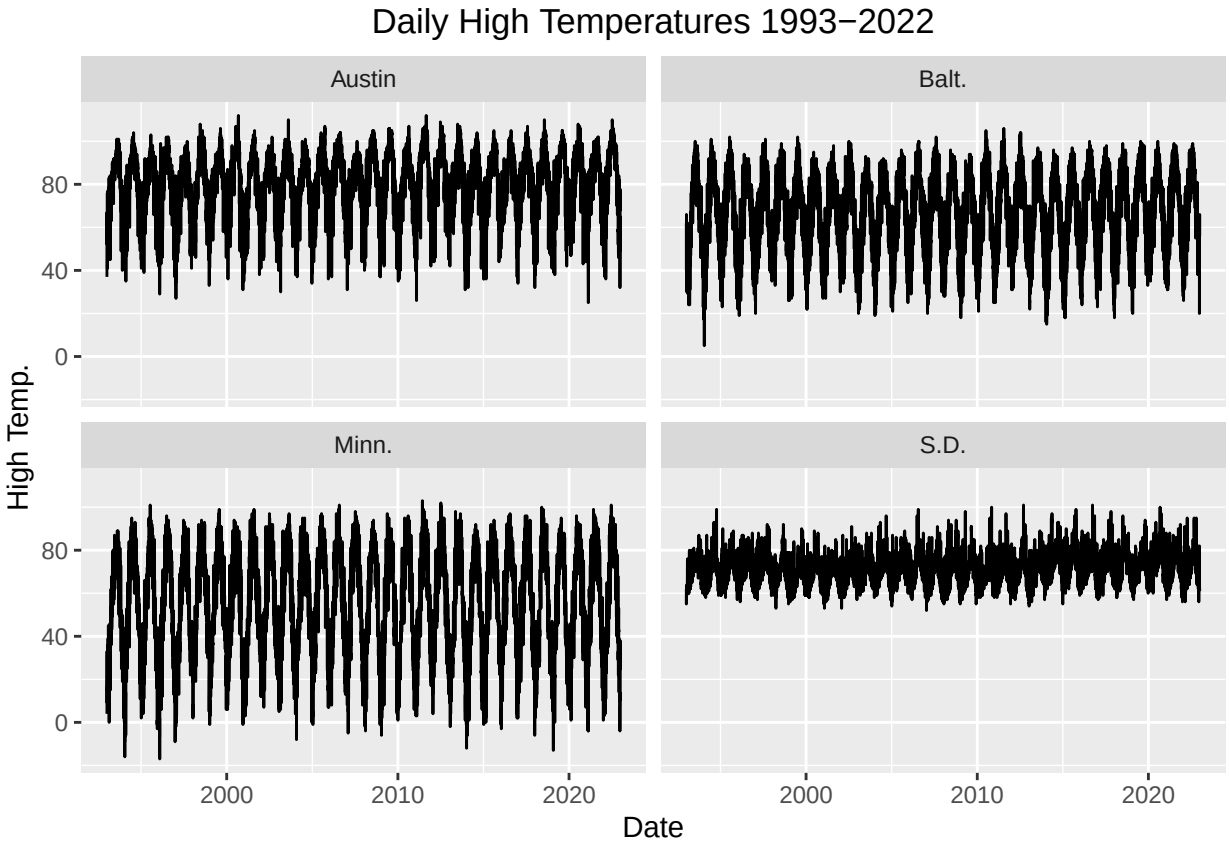
This project investigates the application of automated smoothing methods to temperature data. Temperature data has several features that complicate the task. These features are detailed in Section {X}, and potential solutions are discussed in subsequent sections. I find that *Super-Smoothing with Leave-k-Out Cross-Validation* does a pretty good job of smoothing the data where other methods fail.

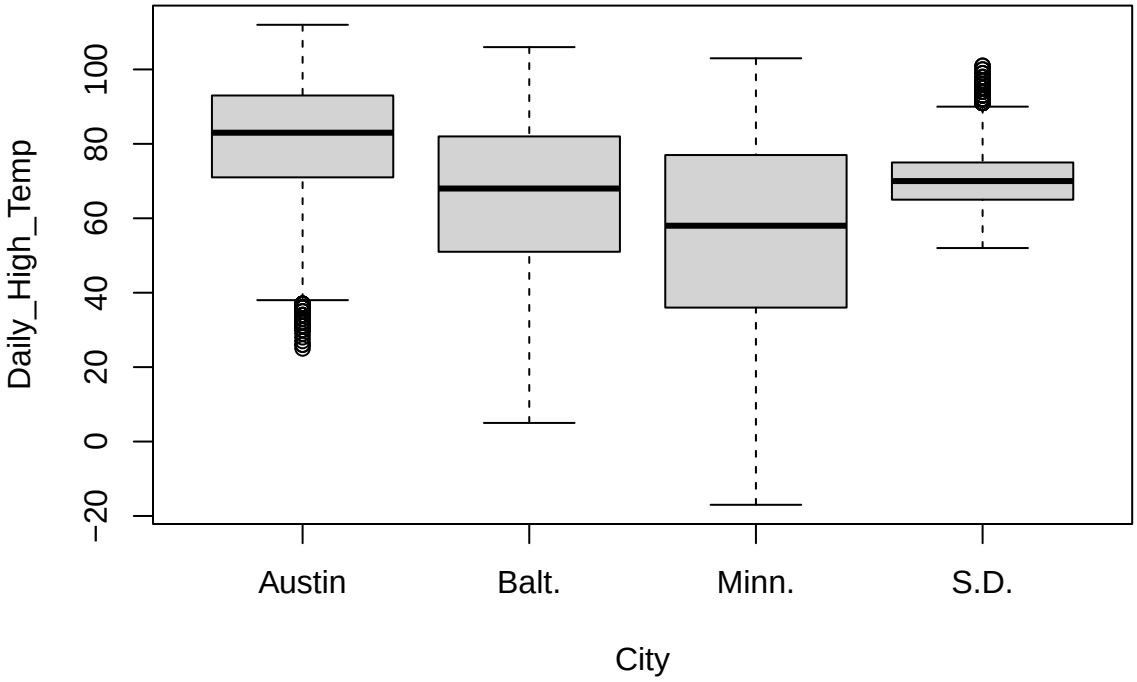
1.1 Data

NOAA data request

Austin, San Diego, Minneapolis, Baltimore

Available back into at least the 40s. Truncated to 1993 - 2022 to make computation more manageable.





2. Kernel Smoothing

Overview: Kernel smoothers apply a weighted average to the data. They take a *bandwidth* parameter that controls how smooth the output curve is.

Smoothing methods try to find a good *smoothing function* that cuts through the noise to give the underlying trend. The value of a smoothing function at a point is generally calculated as some type of weighted average of the values at surrounding points. For instance, the temperature given by the smoothing function on Oct. 1, 1996 is a weighted average of the actual temperatures on and around that date.

Perhaps the simplest smoother is the *constant-span running mean* [2 Givens and Hoeting p. 366]. This smoother is given a parameter k that defines a window size for taking the mean. Suppose we apply the constant-span running mean with $k = 7$ to the temperature data. To calculate the smoothing function on Oct. 1, 1996, we take the mean temperature of the $\frac{k-1}{2} = 3$ days before, the day itself, and the $\frac{k-1}{2} = 3$ days after: in other words, we calculate the mean high temperature from Sep. 28 to Oct. 4.

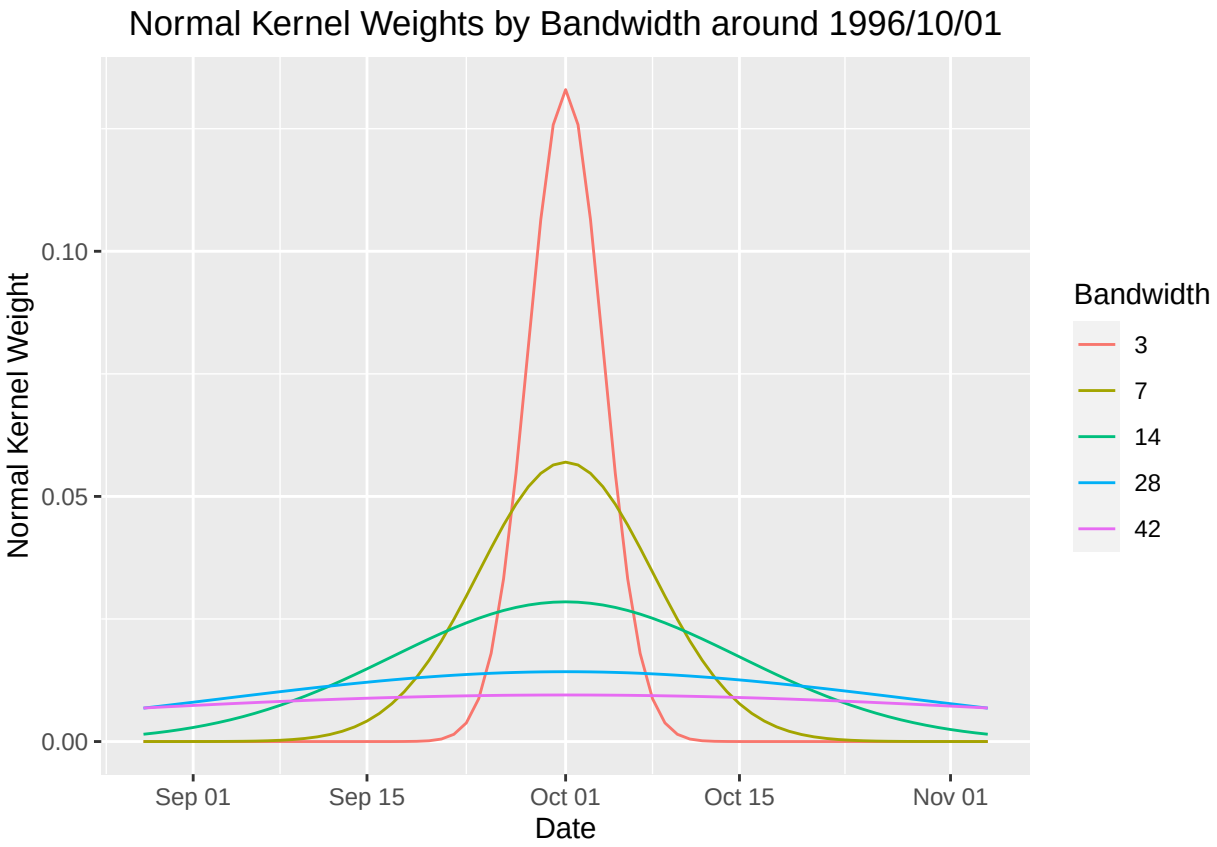
While its simplicity is attractive, the constant-span running mean has some problems: the window parameter defines a sharp cutoff for the points it considers, and points at the edge of the window are given as much weight as points in the middle.

The smoothing method used in this project is known as *kernel smoothing* and tries to fix these problems. A *kernel* is a function that allows kernel smoothers to take truly *weighted* averages. When calculating the

smooth function at a given point, the kernel weights observations based on how close they are to that point: the closer the observation, the higher the weight. This project uses a *normal kernel* that assigns weights around a given point in the bell shape of a normal probability distribution.

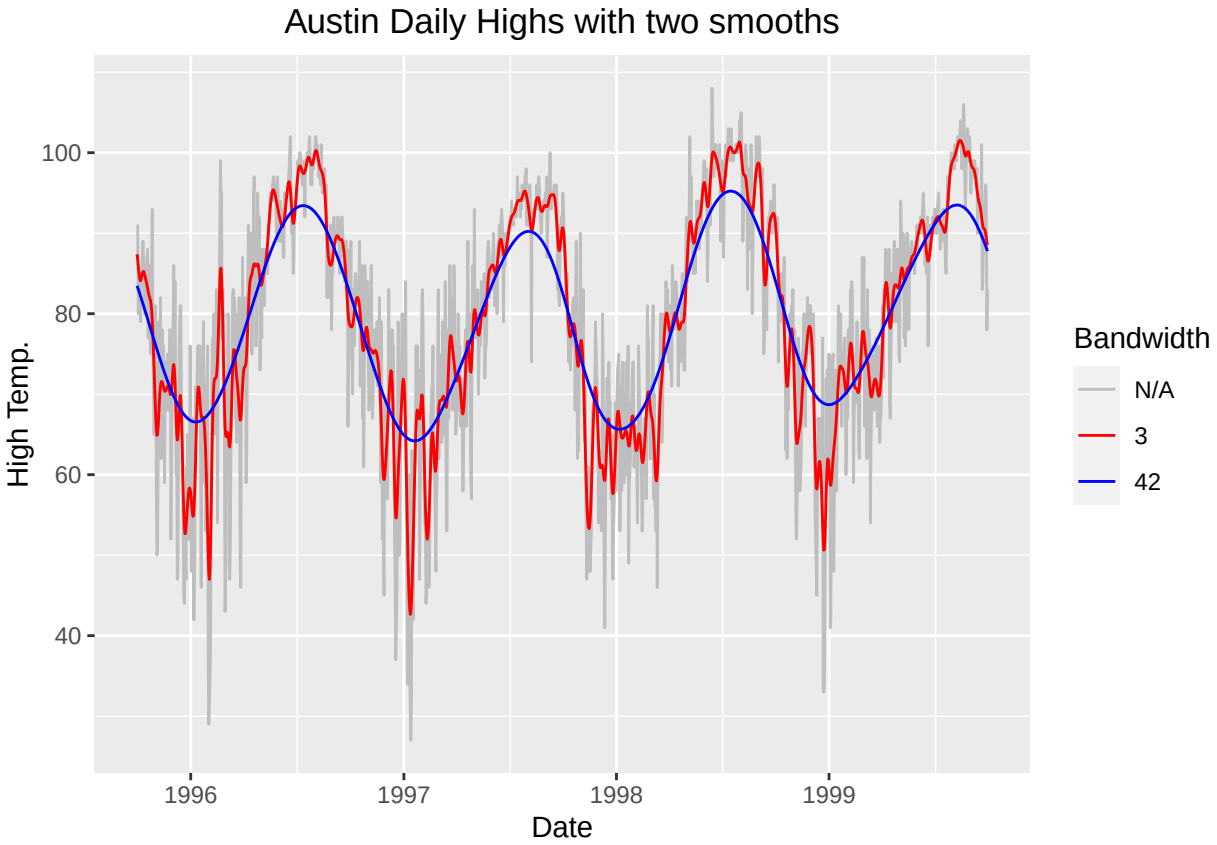
The effect of distance on weighting in controlled by the all-important *bandwidth* parameter h. If h is set to a low value, observations around the given point are weighted very highly, and weights steeply decrease as distance from the point increases. Conversely, if h is set to a high value, close points are given relatively less weight, and weights decrease less steeply with distance.

Suppose again that we're calculating the smoothed temperature on Oct. 1, 1996. The below plot shows how the temperatures on days around this date are weighted given different bandwidth parameters:



Given a bandwidth $h = 3$, the smoothed temperature on Oct. 1 mostly depends on the temperatures in a small window around that date. Given a bandwidth $h = 28$ or $h = 42$, the smoothed temperature on Oct. 1 is giving non-negligible weight to temperatures in August and November.

As seen below, the bandwidth has a significant impact on the smooth:



The kernel smoother with $h = 3$ overfits the data, following its fluctuations too closely and failing to provide much of a smooth at all. The kernel smoother with $h = 42$ underfits the data, failing to fully capture the seasonal extremes in summer and winter.

3. Bandwidth Selection

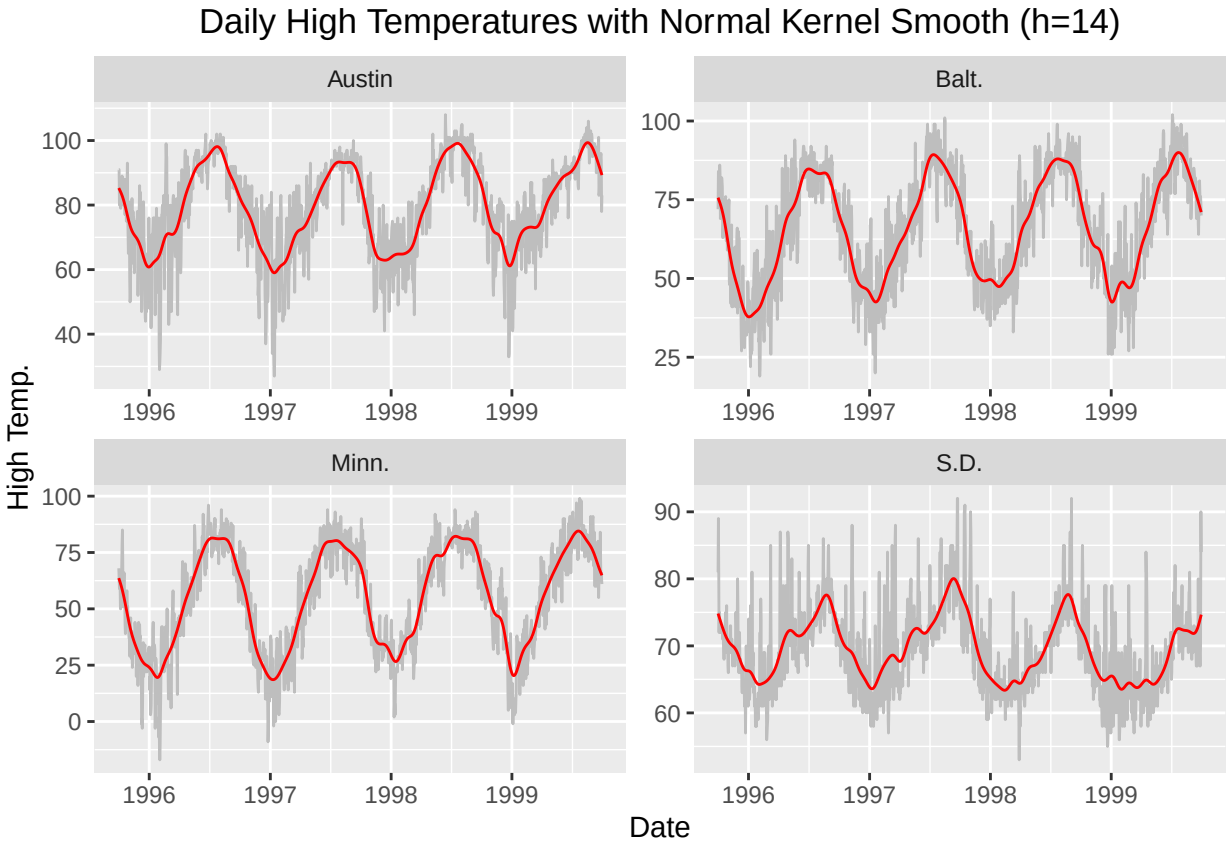
Overview: Automated bandwidth selection methods scale better, so they're preferred to manual selection when we need to smooth many data sets and / or large data sets.

Selecting an appropriate bandwidth is crucially important for getting a good smooth of the data (it's generally viewed as much more important than the choice of kernel function). We can divide methods of bandwidth selection into two broad categories, manual and automated.

3.1 Manual Bandwidth Selection

A common way to manually select a bandwidth is by trial and error with visual inspection of the smooths given by different bandwidths. We plot various smooths over the data and see how well they capture the overall trend(s).

Here are plots of a normal kernel smooth with $h = 14$ for all 4 cities:



By visual inspection, a bandwidth $h = 14$ seems to do a pretty good job of smoothing the data, especially for Baltimore and Minnesota.

3.2 Automated Bandwidth Selection

A main problem with manual bandwidth selection is that it doesn't scale well in situations with many data sets and / or large data sets. There are reasons for this beyond the obvious time-intensiveness:

- Many data sets: Even if they're all about the same type of phenomenon, different data sets can have different properties that make different bandwidths more appropriate. We shouldn't assume that a bandwidth that works well on some of the data sets will work well on all of them. Notice above how the smooth with $h = 14$ is noticeably worse for San Diego, containing many small wiggles due to short-term temperature fluctuations. A higher bandwidth could help remove those wiggles and provide a better smooth.
- Large data sets: It's difficult to visually inspect the smooth over data sets that cover a wide "x-value" range. The above graphs only show 4 years out of 30, so we don't know how well the smooth does in other years.

Methods for automated bandwidth selection are explored in the rest of this project.

4. Leave-One-Out Cross-Validation (LOOCV)

Overview: Leave-one-out cross-validation (LOOCV) chooses a bandwidth by simulating how well different bandwidths extend to new data. It falls under the training set / testing set paradigm, with each test set having size 1. LOOCV fails to select a good bandwidth for temperature data: given a set of candidate bandwidths, it selects the smallest candidate for each data set.

Given a data set of size n, leave-one-out cross-validation performs n training set / testing set splits. As indicated by the name, each LOOCV testing set contains only one data point that has been left out of the corresponding training set.

LOOCV chooses among a set of candidate bandwidths by measuring how well smoothers with those bandwidths generalize from the training sets to the test sets. Generalization ability is typically measured using some error function such as the mean-squared error (MSE) used in this project. For a given candidate bandwidth, the LOOCV algorithm iterates through the train / test splits. For a given split, it "trains" a smoother with that bandwidth on the training set and calculates the squared error when applied to the (singleton) test set.[fn.] The MSE across all splits is returned as the performance measure for that bandwidth. The candidate bandwidth with the smallest MSE is returned as the "optimal" choice.

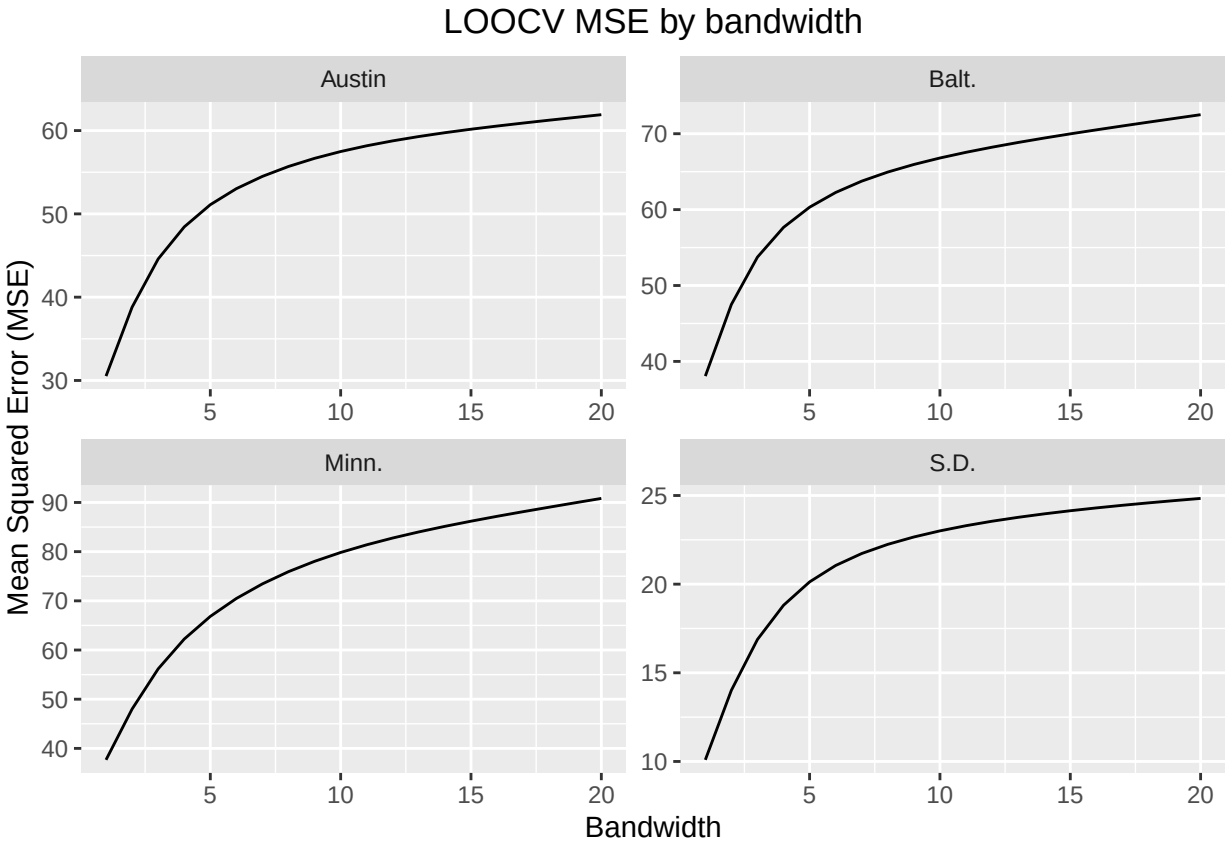
[fn.] Fortunately, a mathematical shortcut can be applied so that each smoother doesn't actually have to be trained n times. See [X] p. XXX.

I applied LOOCV to each temperature data set, testing candidate bandwidths ranging from $h = 1$ to $h = 20$ in the integers (as we'll see, testing larger bandwidths is unnecessary).

4.1. Results

The plot below shows the test-set MSE at each candidate bandwidth for each data set. The test-set MSE monotonically increases with bandwidth size, so the LOOCV algorithm selects the smallest candidate ($h = 1$) for each city.

In other words, LOOCV fails to select good bandwidths for smoothing these data sets. As seen in Section 2, very small bandwidths barely smooth the data. By selecting the smallest candidate, LOOCV is simply trying to follow the data as closely as possible instead of smoothing it.



5. Difficulties Posed By Temperature Data

Overview: Temperature data has at least 3 properties that pose difficulties for kernel smoothing and bandwidth selection:

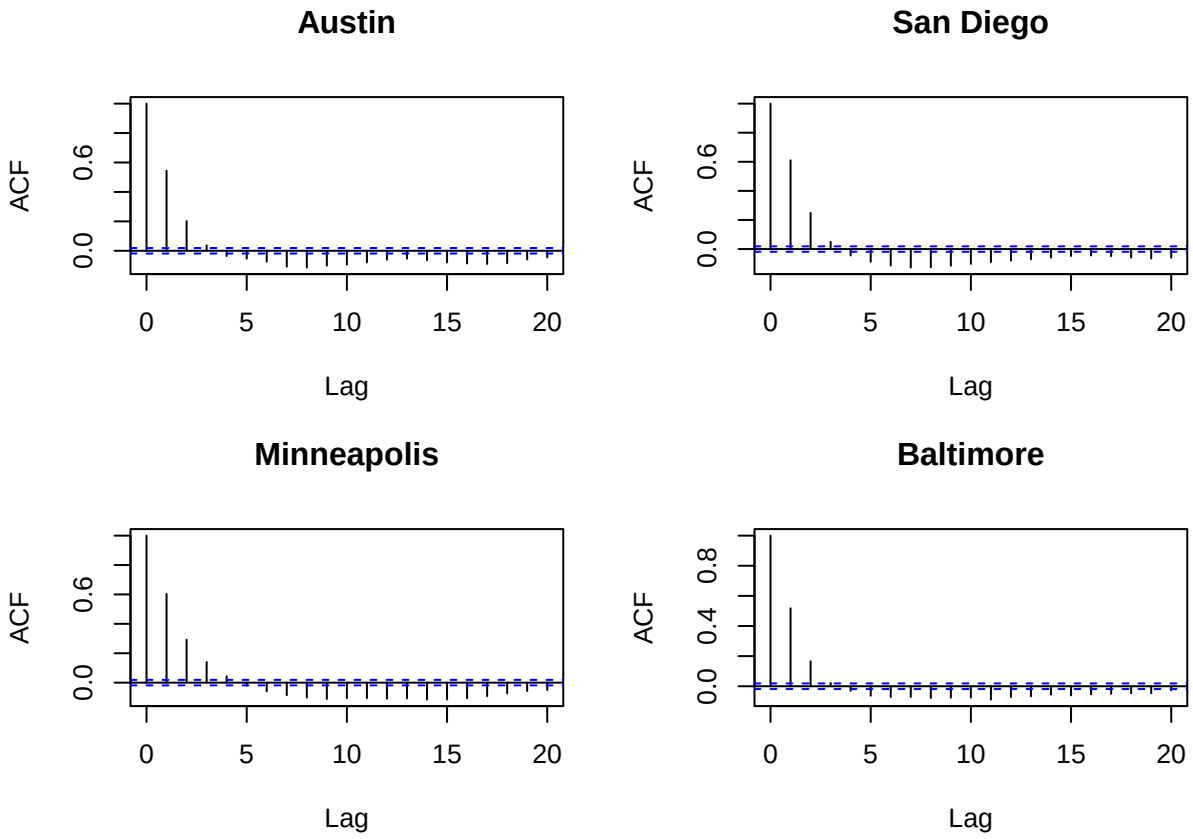
- Non-Constant Variance
- Dependent Data
- Outliers

Understanding why LOOCV fails to smooth temperature data can help us pick better methods for automated bandwidth selection.

5.1. Non-Constant Variance

5.2. Dependent Data

Plot ACF



5.3. Outliers

6. Super-Smoother

7. Leave-K-Out Cross-Validation

8. References