The Astronomical Journal, 158:143 (15pp), 2019 October https://doi.org/10.3847/1538-3881/ab3984

Wōtan: Comprehensive Time-series Detrending in Python

Michael Hippke[1], Trevor J. David[2], Gijs D. Mulders[3,4], and René Heller[5]
[1] Sonneberg Observatory, Sternwartestr. 32, D-96515 Sonneberg, Germany; michael@hippke.org
[2] Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109, USA
[3] Department of the Geophysical Sciences, University of Chicago, USA
[4] Earths in Other Solar Systems Team, NASA Nexus for Exoplanet System Science, USA
[5] Max Planck Institute for Solar System Research, Justus-von-Liebig-Weg 3, D-37077 Göttingen, Germany
Received 2019 June 13; revised 2019 July 24; accepted 2019 August 6; published 2019 September 11

Abstract

The detection of transiting exoplanets in time-series photometry requires the removal or modeling of instrumental and stellar noise. While instrumental systematics can be reduced using methods such as pixel level decorrelation, removing stellar trends while preserving transit signals proves challenging. As a result of vast archives of light curves from recent transit surveys, there is a strong need for accurate automatic detrending, without human intervention. A large variety of detrending algorithms are in active use, but their comparative performance for transit discovery is unexplored. We benchmark all commonly used detrending methods against hundreds of *Kepler*, *K2*, and *TESS* planets, selected to represent the most difficult cases for systems with small planet-to-star radius ratios. The full parameter range is explored for each method to determine the best choices for planet discovery. We conclude that the ideal method is a time-windowed slider with an iterative robust location estimator based on Tukey's biweight. This method recovers 99% and 94% of the shallowest *Kepler* and *K2* planets, respectively. We include an additional analysis for young stars with extreme variability and conclude they are best treated using a spline-based method with a robust Huber estimator. All stellar detrending methods explored are available for public use in `Wōtan`, an open-source Python package on GitHub (https://github.com/hippke/wotan).

Key words: eclipses – methods: data analysis – methods: statistical – planetary systems – planets and satellites: detection

1. Introduction

Over the last decade, millions of stellar light curves have been collected with ground-based surveys, e.g., HATNet (Bakos et al. 2004), WASP (Pollacco et al. 2006), KELT (Pepper et al. 2007), and CHESPA (Zhang et al. 2019), and from space, e.g., with *CoRoT* (Auvergne et al. 2009), *Kepler* (Borucki et al. 2010), and *K2* (Howell et al. 2014), and more missions are underway, such as *TESS* (Ricker et al. 2015) and *PLATO* (Rauer et al. 2014). These data are searched for tiny dips in brightness, potentially indicative of transiting planets. Due to the large number of light curves, search algorithms must be automatic, ideally performing detrending simultaneously with the transit search to avoid detrending-induced distortions. Alternatively, the most common method to find transits in stellar light curves with variability is to "pre-whiten" the data (Aigrain & Irwin 2004), an approach that is supposed to remove all time-dependent noise (or irrelevant signals) from the data prior to a transit search. That said, no detrending algorithm is perfect and in fact they can actually induce transit-like false-positive signals in the light curves that were not genuinely present in the data (Rodenbeck et al. 2018). We are left with the question as to which detrending algorithm works best for a subsequent transit search.

Instrumental trends in these data are usually mitigated by fitting and subtracting cotrending basis vectors (Thompson et al. 2016), decorrelation techniques between photometrically extracted light curves and the telescope pointing (Vanderburg & Johnson 2014; Lund et al. 2015; Aigrain et al. 2016), or methods such as the trend-filtering algorithm (Kovács et al. 2005) using simultaneously observed stars to measure and remove systematic effects, and similar procedures (Kim et al. 2009).

However, even with post-instrumental processing, transit dips are often overshadowed by stellar trends (Hippke & Angerhausen 2015). Stellar noise is star-specific and occurs with diverse characteristics on all timescales. Stellar activity cycles, such as the 11 yr solar activity cycle (Schwabe 1844; Usoskin et al. 2009), appear to be present on different timescales in other stars as well (García et al. 2010; Montet et al. 2017). The solar rotation of ∼27 days (Bartels 1934; Beck 2000) produces noise from spots, as discovered by Galileo Galilei in 1612. Other stars exhibit brightness variations on many different timescales and with a range of amplitudes. Young stars can show bursts with amplitudes of up to 700% on timescales of 1–10 days (Cody et al. 2017; Cody & Hillenbrand 2018), and RRab Lyrae stars have amplitudes of ∼50% over 0.5 day. Photometrically quiet stars like our Sun or Kepler-197 (Rowe et al. 2014), exhibit only weak (0.1%) long-time (months) variation. Intermediate cases are common, e.g., Kepler-264 b (Hippke 2015) with trends on timescales of hours to days. Systems where stellar variability and transit signals occur on similar timescales require robust detrending methods; see Figure 1 as an example of filter-size dependency, where the transit depth is altered when using the Savitzky–Golay method.

The importance of variability on short timescales has motivated the usage of a broad range of different detrending methods. After an extensive literature research, however, it appears that there is no systematic overview of their strengths and weaknesses.[6] Common methods are sliding medians (e.g., Charbonneau et al. 2009; Tal-Or et al. 2013; Burke et al. 2014; Sinukoff et al. 2016; Dorn-Wallenstein et al. 2019), sliding means

[6] That said, we found a brief comparison of a sliding median with a sum-of-200-sines filter by Bonomo et al. (2009). And Rodenbeck et al. (2018) investigated the suitability of various detrending methods in the context of exomoon detections.

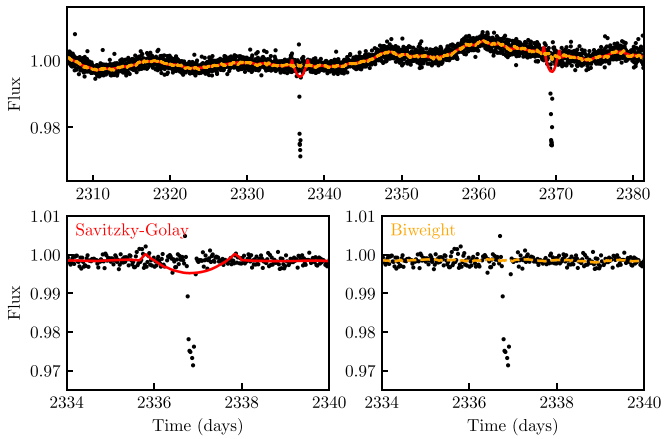

Figure 1. Light curve of EPIC211995398 (black points, Pope et al. 2016) with a Savitzky–Golay filter (red) as used in the `lightkurve` package (Barentsen et al. 2019) with a default of $w = 101$ cadences, polynomial order $p = 2$, and a time-windowed estimator (orange, Tukey's biweight, $w = 0.5$ day). The last represents a reasonable fit to the data, while the Savitzky–Golay overfits the in-transit points and reduces the transit depth by ∼15%, making detection more difficult (although in this example still trivial).

(Moldovan et al. 2010; Wheatley et al. 2010), polynomial filters (Fabrycky et al. 2012; Gautier et al. 2012; Giles et al. 2018), splines (Mann et al. 2016a; Vanderburg et al. 2016; Livingston et al. 2018; Mayo et al. 2018; Rice et al. 2019), LOESS regressions (Luger et al. 2017; Caceres et al. 2019), Cosine Filtering with Autocorrelation Minimization (CoFiAM; Kipping et al. 2013; Rodenbeck et al. 2018), Gaussian processes (GPs; Aigrain et al. 2015; Crossfield et al. 2016; Cañas et al. 2019), the Savitzky–Golay (Savitzky & Golay 1964) filter (Batalha et al. 2011; Gilliland et al. 2011; Stumpe et al. 2012; Howell et al. 2014), wavelets (Carter & Winn 2009; Huber et al. 2013; Batalha et al. 2013; Barclay et al. 2015; Seader et al. 2015; Grziwa & Pätzold 2016), frequency filtering through Fourier decomposition (Quintana et al. 2013), sometimes as a Butterworth filter (Niraula et al. 2018), and combinations such a sliding median followed by a polynomial, (Davenport 2016; Paudel et al. 2018), and sliding median filters (Heller et al. 2019). Occasionally, adaptive filtering is performed. After determining the main modulation period from stellar rotation, a specific window (or kernel) can be defined (Pope et al. 2016; Rizzuto et al. 2017).

These myriad methods open the valid question of what works best for the purpose of blind transit searches, and how big the discrepancy is in using the best, versus some other method. Here, we present a review and comparison of the relevant methods for removing stellar variability. The algorithms presented have been included in `Wōtan`, an open-source Python package.

An alternative approach to detrending with a subsequent transit search is the simultaneous modeling of trends with a transit search. This approach was applied by Foreman-Mackey et al. (2015) to one *K2* campaign, suggesting that the computational cost is acceptable. Recent speed-ups from linear algebra and gradient-based methods introduced by new tools such as Starry (Luger et al. 2019) and Celerite (Foreman-Mackey et al. 2017a) may make this approach more attractive. It is, however, still unclear whether the simultaneous method is beneficial. A recent analysis concluded that "the simple (and more traditional) method that performs correction for systematics first and signal search thereafter, produces higher signal recovery rates on the average" (Kovács et al. 2016).

The paper is structured as follows: In Section 2, we review the commonly used estimators and detrending methods. In Section 3, we present three experiments to test these methods, with the results in Section 4.

2. Algorithms

Common algorithms (Section 1) can broadly be categorized as sliding filters, splines and polynomials, and GPs. This section reviews the relevant methods.

2.1. Sliding Filters

In the statistical literature, a common filter like a sliding (rolling, walking, running) mean or median is categorized as a *scatterplot smoother* in the form

$$y = s(x) + \epsilon \tag{1}$$

where a trend (or smooth) s is estimated. After sorting the data in chronological order so that $x_1 < x_2 < ... < x_n$, a simple (cadence-based) sliding mean smoother S can be calculated for each x_i by averaging a range of y_j values corresponding to each x_i:

$$S(x_i) = \sum_{j \in W(x_i)} (y_j)/w_i \tag{2}$$

in a neighborhood with indices $W(x_i)$ that contains a total of w_i data points. Typically, the neighborhood is symmetrical with the nearest $2k + 1$ points, including k data points taken before x_i, k data points taken after x_i, and x_i itself:

$$W(x_i) = \{\max(i - k, 1)..., i - 1, i, \\ i + 1, \ ..., \min(i + k, n)\}. \tag{3}$$

This kernel smoother uses a rectangular window, or a "box" (sometimes called a tophat, or boxcar) kernel. Other commonly used kernels are Gaussian densities (computationally expensive because they never reach zero) and, e.g., the parabola of Epanechnikov (1969). To calculate $W(x_i)$ with a non-rectangular kernel, the jth point is given a weight

$$g_{ij} = \frac{c_i}{\lambda} d\left(\frac{|x_i - x_j|}{\lambda}\right), \tag{4}$$

where λ is a bandwidth-tuning constant, c_i is used to normalize the sum of the weights to unity for each x_i, and d describes the kernel. For example, the Epanechnikov kernel has $d(z) = 3/4$ $(1-z^2)$ for $z^2 < 1$, (with z as the distance from the midpoint of the kernel) and zero otherwise.

Cadence-based filters are applied to a fixed number of data points, whereas time-windowed filters use a window fixed in a second dimension. Cadence-based filters are simple and fast (Section 5.3), but require equally spaced data without gaps to function as designed. In the real world, data points are often missing due to, e.g., cosmic ray hits or momentum dumps. Their spacing in time typically originates with a constant cadence of a *local* clock. On a body moving around the Sun in an Earth-like orbit, barycentering these observations causes cadence spacings to periodically drift by 16 minutes over the course of a year. The most severe effect for cadence-based filters in practice, however, are longer gaps, e.g., a cadence-based filter

stitching together sinusoidal variation where half a rotation period is missing.

2.2. Cadence-based Sliding Filters

For cadence-based sliders, we test a sliding median (using the `scipy.medfilt` implementation, Jones et al. 2001) and the Savitzky & Golay (1964) method `scipy.savgol_filter`. It fits successive sub-sets of equally spaced points with a low-degree polynomial (typically order $p = 2$ or $p = 3$) with a least-squares calculation. The resulting analytical solution is then applied to all subsets. The Savitzky–Golay is the default detrending method in the `lightkurve` package (Barentsen et al. 2019) with a default of $w = 101$ cadences and $p = 2$. Following its original definition, it is cadence-based, and we are not aware of any other usage.

2.3. Time-windowed Sliding Filters

A time-windowed filter avoids issues faced by cadence-based ones. We can define a time-windowed filter with a window length w centered on the time $t(x_i)$ for each x_i. A rectangular (i.e., symmetric and with uniform weights) window of size w contains those data points $N(x_j)$ whose $t(x_j)$ are in $[t(x_j)-w/2...t(x_j) + w/2]$. We implement a module for such a time-based slider in Python to allow for a comparison of various estimators from the astrophysical and statistical literature.

2.3.1. Mean and Median

The most common smooth location estimators at each time $t(x_j)$ are the mean and the median. These two represent the extreme ends of estimators with respect to robustness and efficiency. For a normal distribution, the mean is the most efficient estimator. A maximally efficient estimator is defined as that which has the minimal possible variance of the location estimate (the minimum mean squared error in the unbiased case) for a given noise distribution (Kenney & Keeping 1947, Section 3.2, 4.8f). The median is robust for distributions with up to 50% outliers, but its efficiency, measured as the ratio of the variance of the mean to the variance of the median, is lower. The uncertainty on a sample median is $\sqrt{\pi/2} \approx 1.25$ times larger than the uncertainty on a sample mean (Press 1992, p. 694, Huang 1999). The mean, however, can be pushed toward $\pm\infty$ with a single outlier. This is known as the bias–variance tradeoff. The median results in maximal robustness, but minimal efficiency—this extra source of jitter can be seen by eye (Figure 2). Another issue with the median is visual: its trend is always exactly equal to one of the points inside the sliding window. Dividing the flux by the trend, and plotting these detrended data, results in a large number of points equal to unity. This cluster of points parallel to the abscissa in a diagram may appear visually unprofessional and disqualify the median for publication quality figures.

There is a rich statistical literature about robust estimators in between these two extremes, each maximizing some ratio of efficiency versus robustness. For example, the trimmed mean is a simple robust estimator of location that deletes a certain percentage of observations (e.g., 10%) from each end of the data, then computes the mean in the usual way. It often performs well compared to the median and mean, but more robust estimates are available for many use cases. The following subsections review some of these estimators.

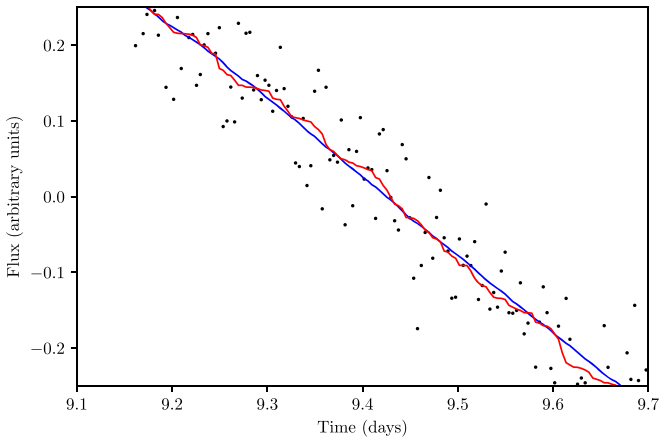

Figure 2. Synthetic white noise data (black) with a sliding mean (blue) and sliding median (red) trend, both of the same window size of 101 cadences. The sliding mean is very close to the true trend, while the median shows additional jitter, because its estimate is 25% less efficient.

2.3.2. Trimmed Mean and Median

A very simple robust location estimator is the trimmed mean which rejects a small fraction of the most extreme values in a time series of observations prior to averaging the remaining data. This brings robustness over the simple mean, while maintaining most data, allowing for a better efficiency compared to the median. When this clipping is increased, it becomes intuitively clear that the median is the result of clipping 50% of each side of the distribution. A one-sided x-sigma clipper is expected to remove

$$Y = \frac{2}{1 - \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)} \tag{5}$$

values from a Gaussian distribution, where $\mathrm{erf}(x)$ is the Gauss error function. For $x = 3\sigma$, where σ is the standard deviation of the sample, we expect to clip $\sim 1/741$ values, or ~ 5 from a typical *K2* light curve worth 3500 data points. In practice, often many more outliers are present. One issue with the truncated mean is that it produces a biased estimator if the underlying sample is not symmetric. In the presence of transits (and few flares), this is often the case.

In effect, a trimmed mean is a two-step procedure. First, outliers are detected and removed. Second, the efficient estimation method (the least-squares) is applied to the remaining data. There are two general challenges with this approach. First, the outlier detection method relies on a non-robust initial estimate. This can result in the masking effect, where a group of outliers mask each other and escape detection (Rousseeuw & Leroy 1987). Second, if this is avoided with a high breakdown initial fit (i.e., a larger proportion of the data is removed), the following estimate is inefficient (similar to the median), plus it inherits the inefficiency of the initial (non-robust) estimate (He & Portnoy 1992).

2.4. The Huber Function

While the trimmed mean performs well, compared to the median and mean, more robust estimators are often available. Peter J. Huber proposed a generalization of the maximum likelihood estimation called M-estimators (Huber 1964). Non-linear least-squares and maximum likelihood estimation are

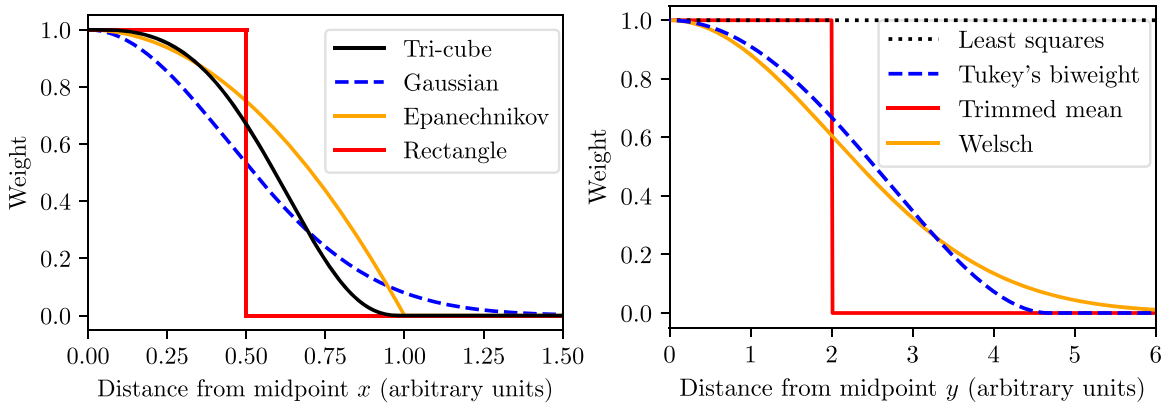

Figure 3. Left: common window functions for weights of points of the independent variable (usually x, or t in our case). Most commonly, a rectangular window (red line) assigns weights of unity to all values within the current (sliding) window, and zero otherwise. Other well-known choices are the Gaussian and the Epanechnikov. The tri-cube is somewhat steeper in slope, giving more weight to the center (but not as much as the rectangle). Right: relative weights of robust estimators as a function of distance from the mid-point y (here: flux), where the ordinary least-squares method has constant weight. The trimmed mean is a step-function shown for a cut fraction of 0.1, which corresponds to a $\sim 2\sigma$ clipping in a Gaussian distribution. The estimators by Tukey (here $c = 4.685$) and Welsch ($c = 2.11$) have smoother transitions.

special cases of M-estimators. Given an assumed distribution, one can select an estimator with the desired properties for the bias–variance tradeoff. The estimator is optimal if the data are close to the assumed distribution.

Estimators can be distinguished by their loss function. The ordinary least-squares method uses the squared loss ($L(a) = a^2$), where a are the residuals, i.e., the difference between the observed values and the midpoint. A robust estimator with a loss function that is less affected by very large residual values has been described by Huber (1981) and Huber & Ronchetti (2011, p. 172). The Huber function is quadratic for small values, and becomes linear for larger values. Its loss function can be approximated as (Charbonnier et al. 1997)

$$L(a) = c^2(\sqrt{1 + (a/c)^2} - 1), \quad (6)$$

where c is a tuning parameter to adjust the steepness of the transition between quadratic loss (mean squared error, $a^2/2$ for small values) and linear loss (absolute deviation, a for large values). For $c \to \infty$, the function becomes the usual least-squares estimator.

2.4.1. Tukey's Biweight (Bisquare)

A well-known robust location estimator is Tukey's biweight (or bisquare) (Mosteller & Tukey 1977, pp. 203–209).[7] Compared to the ordinary least-squares, where the weights are constant at unity, the biweight has weights as a distance from the central location as $L(a) = (-(a/c)^2)^2$ for $|a| < c$, and zero otherwise (Figure 3). Typical values are $c = 4.685$ for an asymptotic efficiency of 95% compared to that of least-squares, for a normal distribution. Larger values for c make the estimate less robust, but more efficient. Another commonly found value is $c = 6$ which includes data up to four standard deviations from the central location (Mosteller & Tukey 1977). Then, the efficiency is ∼98% (Kafadar 1983). The biweight with $c = 6$ has proven useful for velocity clustering of galaxies (Beers et al. 1990). Our tests with *K2* data show $c \sim 5$ to be optimal, although the variation for $c = 4$ and $c = 6$ results only in the loss of one planet (of 100) in the experiment in Section 4.1.

The biweight can be used as a one-step estimate (as implemented by `Astropy`; Price-Whelan et al. 2018) with the median as the first guess for the central location. Alternatively, it can be approximated with iterative algorithms such as Newton–Raphson, where the result of each iteration is used as the starting guess for the next. We have determined the difference between the one-step estimate and a converged solution numerically. This difference is typically small, but not negligible (in agreement with Hampel et al. 2011, p. 152): usually between a few and a few tens of ppm (10^{-6}) in *Kepler* and *K2* data. This corresponds to a few percent of the depth of an Earth–Sun transit (∼84 ppm). This additional detrending noise, caused by a sub-optimal location estimator, is avoidable with a few (three to five) Newton–Raphson iterations for convergence to the $<10^{-6}$ (<1 ppm) level. Our implementation within `Wōtan` allows us to set this threshold to an arbitrary value.

2.4.2. Other Robust Estimators

There is a variety of other robust M-estimators which share the idea that data near the center are assigned more weight. Among the most common are the Hodges–Lehmann–Sen (Hodges & Lehmann 1963; Sen 1963) with a breakdown point (fraction of outliers above which results become incorrect) of 0.29 (median: 0.5). It is most suited if the underlying distribution is a mixture of normal distributions. Other estimators include the Welsch–Leclerc with an exponential decay, $L(a) = \exp((a/c)^2)$ (Dennis & Welsch 1978; Leclerc 1989), and Andrews' sine wave (Andrews et al. 1972, pp. 3–28). A related class of estimators is rank-based (using adaptive weights), aiming to maximize both robustness and efficiency. From this class, we include the τ estimator in the experiment (Yohai & Zamar 1988).

2.5. Local Regressions: LOWESS

LOWESS/LOESS (locally weighted/estimated scatterplot smoothing) is a non-parametric regression technique developed by Cleveland (1979, 1981). In essence, the method is a generalization of a moving window, where each window is used to fit a locally weighted polynomial regression (see

[7] This method is named after the mathematician John Tukey (1915–2000) who also coined the terms "bit" (Shannon 1948) and "software" (Buchholz & Shapiro 2000), invented the fast Fourier transform (Cooley & Tukey 1965), and the boxplot (Tukey 1977; McGill et al. 1978), among many other things.

Chapter 8 of Chambers & Hastie 1992; Wilcox 2017). The window can be a boxcar in the x-axis, or any other window function (e.g., a Gaussian). The regression on y is typically performed with least-squares or a robust estimator such as the biweight. For our tests, we use the implementation from the Python package `statsmodels` (Seabold & Perktold 2010) based on Hastie et al. (2009) with a bisquare hardcoded to $c = 6$ on the y values, selectable iterative depth, and a tricube window in x with weights as $(1-d^3)^3$ with distance d from x. Similar parameters have been used to determine galaxy properties (Cappellari et al. 2013). Giles et al. (2018) re-invented a filter similar to the LOWESS with outlier-clipping for robustness, instead of a smooth biweight. LOWESS appears to be rarely used, with ony two occurrences in a literature review (Luger et al. 2017; Caceres et al. 2019).

2.6. Friedman's "Super-Smoother"

Friedman (1984) proposed a regression smoother based on local linear regression with adaptive bandwidths. In multiple passes, it selects a "best" bandwidth from initial estimates at each data point over the range of the predictor variable. For our tests, we use the implementation by Vanderplas (2015) and Vanderplas & Willmer (2015). Finding appropriate duration ranges for the three rounds of iterations, and the "bass enhancement" α which sets the "smoothness" proves computationally expensive. A major problem for the "Super-Smoother" in light-curve data is the fact that it assumes Gaussian errors, which is problematic due to the presence of outliers (e.g., transits). The method has been used with success, however, for finding periods in eclipsing binaries (Becker et al. 2011).

2.7. Gaussian Processes

GPs are popular in many sub-fields within astrophysics, such as spectroscopy (e.g., Gibson et al. 2012), pulsar timing (van Haasteren & Vallisneri 2014), asteroseismology (Brewer & Stello 2009), redshift prediction (Way et al. 2009), and supernova luminosity (Kim et al. 2013).

GPs have also found an application in the simultaneous modeling of noise and transits (e.g., Grunblatt et al. 2017). Here, we only benchmark GPs against other available methods for our scope of removing stellar trends while preserving transits. The reason we do not perform simultaneous transit searches and GP fits (or any detrending, for that matter) is simple: it is prohibitively computationally expensive.

Usually, GPs are used assuming Gaussian noise. GPs can be used as priors with other (more robust) likelihood functions, such as the Student's t-distributed noise model (Neal 1997).

Angus et al. (2018) provides an example of using GPs to model and stellar rotation. First, a Lomb–Scargle periodogram is calculated to determine the rotation. A periodic GP is then informed about the strongest periodic signal. This method was partially successful ($\gtrsim$80%) in fitting the test planets.

We test GPs using george (Ambikasaran et al. 2015; Foreman-Mackey et al. 2017a) and Celerite (Foreman-Mackey et al. 2017b; Foreman-Mackey 2018) with a squared-exponential kernel (Danielski et al. 2013) as used by Barclay et al. (2015), Crossfield et al. (2016), and Thompson et al. (2018), the Matern 3/2 (Wang et al. 2019), and the quasi-periodic (Brahm et al. 2019) for our test sample of young stars.

We test our GPs in the one-pass mode. In addition, we test an iterative 2σ outlier clipping from the fitted trend in each iteration until convergence. The GP hyper-parameters of each kernel are determined by iterating over a grid of plausible values, choosing those that maximize the planet detection yield.

2.8. Splines

Schoenberg (1946) introduced "splines" as a smooth, piecewise polynomial approximation. Today, the most commonly used types are univariate (cardinal) basis (B-) splines (de Boor 1980) where all knots lie in a single dimension and are equally spaced (Ferguson 1964; Ahlberg et al. 1967). Usually, splines are fit by minimizing the sum of the squared residuals. As with a sliding mean, the estimate is affected by outliers. A common method to increase the robustness is an iteratively sigma-clipping re-weighting procedure, also with least-squares calculations. In each iteration, outliers beyond some threshold are clipped, and the procedure is repeated until convergence. We implement this method based on the `SciPy` package in order to offer segmentations of the light curve which has similar methods to the package `untrendy`.[8]

An alternative approach is to use a linear loss (Lawson 1961; Rice & Usow 1968), or a robust M-estimator such as the Huber or the biweight. In our test we include a spline estimated with a Huber-regressor, provided by the `scikit-learn` package.

With these methods, the knot distance must be chosen manually. As with window sizes, there is a trade-off between under- and overfitting. There exists a range of heuristic methods to determine a sensible knot spacing (Friedman & Silverman 1989; Kooperberg & Stone 1991). The most established method, however, is penalized (P-) splines (Eilers & Marx 1996). The extra degrees of freedom from additional knots is judged against the smaller residuals from an improved fit. The optimal weight of the penalty can be determined using cross-validation and Bayesian arguments. We implement this automatic spline-fitter using `pyGAM` (Serven & Brummitt 2018), which conveniently offers the creation of generalized additive models and their selection. We add iterative σ-clipping for 2σ outliers from the fitted trend at each iteration until convergence.

2.9. Cosine Filtering with Autocorrelation Minimization

The CoFiAM algorithm (Kipping et al. 2013) was developed for the application of exomoon-hunting, where it is crucial to protect a time window longer than the transit duration from method-induced trends. It was specifically optimized to reconstruct the morphology of a previously identified transit, and the surrounding region, as faithfully as possible. To do this, it is required to know the location and duration of the transits, and mask them, to assure that the trend is unaffected by the lower in-transit points. Thus, CoFiAM is not per se suited for our blind search experiment.

While the code was described in-depth in Kipping et al. (2013), it was not released as open source. However, it was re-written for validation purposes by Rodenbeck et al. (2018) following the description in the original paper. This implementation is also available through our `Wōtan` package.

CoFiAM uses a sum-of-(co)sines approach, where the optimal detrending is determined iteratively. In each iteration, one harmonic order of a cosine is added and evaluated in the least-squares sense, so that the autocorrelation of the residuals

[8] https://github.com/dfm/untrendy

is minimal. Autocorrelation is the correlation of the signal with a delayed copy of itself as a function of delay. CoFiAM employs the Durbin–Watson statistic to quantify the level of an autocorrelation with a lag of 30 minutes in the default setting. Many models with sums of cosines are tested, of increasing orders until these are too large so that the chosen window size would be compromised.[9] Afterwards, the model with the lowest autocorrelation is selected.

The method can be made outlier-resistant by applying a sliding filter before the actual process. In the original implementation, a 20-cadence window is used with the median as a center estimate, clipping all points more than three standard deviations away. In our `Wōtan` package, a feature-rich outlier-clipping method is available, offering a time-windowed slider with the mean or median as the center estimate, and the standard deviation or the median absolute deviation as the σ clipper.

2.10. Iterative Sum of Sines and Cosines

Similar to CoFiAM described in the previous section, a sum of sines and cosines can be fit to the data (Mazeh & Faigler 2010). Again, the highest degree can be chosen so that a defined window in time is protected. Now, however, no decision is made based on autocorrelation measurements. Instead, the data are divided by the trend, and outliers (e.g., more than two standard deviations from the local mean) are temporarily removed). The procedure is repeated iteratively until convergence, i.e., until no further clipping occurs. Afterwards, this trend is applied to the complete data set.[10]

2.11. Edge Treatment

Any detrending method is challenged near the edges of a time series, where data are missing on one side. There are various methods to handle such situations. On the one hand, the data can be padded to enlarge a full window w (or kernel). Padding can be done by adding a constant value, by mirroring the values nearest to the edge, or by wrapping the beginning and the end of the time series. On the other hand, data near the edge can be discarded after detrending. Some algorithms use yet other methods, such as the Savitzky–Golay filter in `scipy.savgol_filter`, which fits a polynomial to the end segments.

Dropping data near the edges removes at least $2 \times 0.5w \sim 1$ day out of 80 days of *K2* data ($\sim$1%), and more if additional gaps during the time series occur.

3. Test Samples

To cover a maximally diverse sample of transits, we decided to split the experiment into three sample groups. In each, we search for known transits (real or injected) using the transit least squares (`TLS`) transit detection algorithm (Hippke & Heller 2019) with stellar parameters on temperature, surface gravity, radius, and from the EPIC (Brown et al. 2011), KIC (Huber et al. 2014, 2016; Mathur et al. 2017), and TIC (Stassun et al. 2018) catalogs, cross-matched with theoretical limb-darkening values from Claret (2018a, 2018b). This procedure is similar to that applied by the transit least-squares survey (Heller et al. 2019). We set up `TLS` with a fine search grid of the parameter space using an oversampling factor of five for the transit period and a transit duration grid spacing of 5%, with all else as the default in `TLS` version 1.0.23. A detection was accepted if the published period matched the detected period as detected by `TLS` within 1%, and if it was the strongest period detected. Usually, a fixed-value threshold is applied below which detections are not accepted, in order to limit the number of false positives due to noise. For `TLS` and white noise, an SDE (signal detection efficiency) of 9 yields a false alarm probability of 0.01%. An SDE value of x for any given period means that the statistical significance of this period is $x\sigma$ compared to the mean significance of all other periods. This assumes Gaussian (white) noise.

A similar measure of significance is the signal-to-noise ratio (S/N) which compares the depth of the transit model with the out-of-transit noise:

$$\mathrm{S/N} = \sqrt{N_T}\frac{T_{\mathrm{dep}}}{\sigma_{\mathrm{OT}}} \tag{7}$$

with N_T as the number of transit observations, T_{dep} the transit depth, and σ_{OT} the standard deviation of out-of-transit observations. For Gaussian noise, SDE and S/N are qualitatively identical.

In reality, noise is often time-correlated (red), so that spurious signals mimic real transits. Then, the false alarm probability is underestimated. A metric which takes correlated noise into account is the signal-to-pink-noise ratio derived by Pont et al. (2006). It weights the S/N with the amount of red noise in the vicinity of the transit.

For our experiments presented here, we employ the most commonly used metric SDE. We have not applied a threshold, as we are interested not in absolute but relative performances. The lower 10 percentile of SDE values gives a useful insight into potentially missed planets due to a user-chosen cut-off.

Most previous transit searches used the BLS (box least squares) algorithm (Kovács et al. 2002), where the search filter is not a transit-shaped curve, but a box. This reduces the SDE (and thus the recovery rate for small planets) by $\sim$5%–10%. Other than that, we expect no difference in the relative performance of the detrending algorithms. For a comparative analysis of the recovery rates for small planets, see Section 3.1 in Hippke & Heller (2019).

3.1. Robustness Sample of Real Planets in K2

In the first set, we use 100 light curves from the *Kepler K2* mission taken in 13 of the 19 *K2* campaigns in different target fields along the ecliptic between 2014 and 2018. Each light curve covers a time span of about 75 days. We use data from the `EVEREST` reduction (Luger et al. 2016, 2018) which removes most of the instrumental noise. These data are relatively recent and are still being searched for additional planets (Heller et al. 2019). While most systematics are removed by `EVEREST`, some light curves are still contaminated with unflagged outliers. These (low-flux) outliers, together with red-noise systematics, make detrending difficult.

To test a wide variety of parameters, we select candidates from the NASA Exoplanet Archive[11] (Akeson et al. 2013). We include planets among the largest and smallest transit depths, and those with the longest and shortest transit durations (but

[9] A similar method, based on polynomials, was used by Cañas et al. (2019).
[10] We thank Aviv Ofir for suggesting this method.
[11] https://exoplanetarchive.ipac.caltech.edu/

requiring $n \geqslant 3$ transits). The remainders are selected randomly for a total of 100 light curves of planet candidates.

The orbital periods (P) in this set range between 0.6 and 35.4 days, with a mean of 10.4 days and a median of 7.4 days. The transit depths are between 0.009% (96 ppm) and 5% with an mean of 0.5% and a median of 0.1%. The transit durations (T_{14}) from first to fourth contact (Seager & Mallén-Ornelas 2003) cover the range 0.004–0.14 T_{14}/P (mean 0.02, median 0.01).

3.2. Performance Sample of Shallowest Planets in Kepler

The original *Kepler* mission provided more consistent data quality (compared to *K2*) with overall lower noise and, more importantly, few low outliers or other artefacts. It is an effective test bed for a search of the shallowest transiting planets.

Among the planets most difficult to find are those with the shallowest transits. These are also of high interest, as they contain the smallest (Earth-sized), rocky planets. To test the optimal detrending for the shallowest planets, we select the 100 validated planets with the lowest S/N from *Kepler* as listed by the NASA Exoplanet Archive. We use the Pre-search Data Conditioning light curves (Smith et al. 2012; Stumpe et al. 2012).

The period ranges in this set are 0.7–385 days with a mean (median) of 9.6 days (29.4 days). The transit durations cover the range 0.034–0.67 day (mean 0.18 day, median 0.15 day). S/Ns range from 6 to 15 (mean 12.6, median 13).

3.3. Injected Sample of Real Planets in Young Stars from TESS

Young star systems ($\lesssim$Gyr) are important probes to study the primary processes of planetary evolution, in particular if they are host to transiting planets. They serve as benchmark systems to get insights into planet formation, e.g., the migration of hot Jupiters (Heller 2019), the evolution of planetary atmospheres (Madhusudhan et al. 2016), and other physical properties such as mass–radius relationships.

The *K2* and the *TESS* missions monitored several young open clusters such as the Pleiades (*K2* C4, age 110 Myr; van Leeuwen et al. 2017), the Hyades (*K2* C4 and C13, 800 Myr; Brandt & Huang 2015a), and Praesepe (*K2* C5 and C16, 800 Myr; Brandt & Huang 2015b). In addition, *K2* observed the Taurus-Auriga star-forming region (*K2* C4 and C13,1–5 Myr; Kraus et al. 2017) and the Upper Scorpius subgroup of Sco-Cen (*K2* C2 and C15,10 Myr; Mamajek et al. 2012).

Fewer than 15 transiting planets have been discovered in young clusters and associations, although thousands of light curves have been searched (e.g., David et al. 2016, 2019; Libralato et al. 2016; Mann et al. 2016a, 2016b, 2017; Obermeier et al. 2016; Pepper et al. 2017).

Only a few planets are known to be transiting young active stars, not enough for our experiment. This low number may be in part due to the fact that light curves of young stars feature periodic variability of typically 1%–5% in amplitude, usually phased at the stellar rotation period. Often, significant aperiodic variability is present in addition. The (mostly rotational) variability is often temporally complex, with evolution over the course of a ~70 day *K2* observing campaign (Rizzuto et al. 2017).

We extracted light curves for 316 high-probability members of known young moving groups and clusters (Gagné et al. 2018) from the *TESS* full-frame images using the `eleanor` pipeline (Feinstein et al. 2019). Using different methods, we performed a transit search followed by by-eye vetting for each target. No new planet candidates were detected. Therefore, we treat these light curves as free of significant transits and use them for an injection and retrieval experiment.

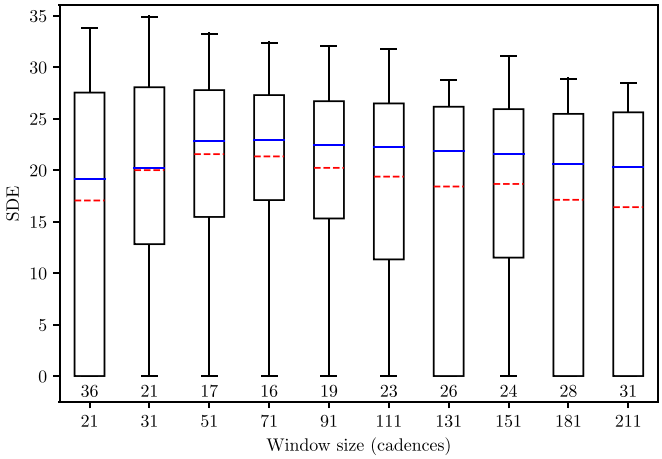

Figure 4. Boxplot for different window sizes of the Savitzky–Golay filter for a `TLS`-search in the *Kepler K2* sample. With polyorder $p = 2$, a window size of $w = 71$ cadences (horizontal axis) yields the lowest number of non-detections (16, bottom row), the highest median SDE (dashed red line) and the second-highest mean SDE (blue line). Higher polyorders yield strictly worse results (not shown). Thus, $p = 2$, $w = 71$ is the optimal parameterization of the Savitzky–Golay filter for a `TLS`-based transit search in *K2* data. Boxes cover the lower to upper quartiles; whiskers show the 10 and 90 percentiles.

We injected planetary transits based on a Mandel & Agol (2002) model using the `batman` package (Kreidberg 2015), with P drawn randomly from the range [1, 15] days, a transit impact parameter of zero, an orbital eccentricity of zero, and R_P/R_* set so that in each case the planetary radius corresponds to a 0.5 $R_{\rm Jup}$ planet. Stellar parameters were pulled from the *TESS* input catalog (Stassun et al. 2018).

To limit the computational effort, only a subset of the most promising detrending methods was tested. We chose to compare the performance of a sliding median and biweight, the LOWESS, and a robust (Huber) spline. In addition, we added a new method owing to the strong periodic variability seen in many young stars. First, we ran a Lomb–Scargle periodogram. Then, with the most significant period, we informed a GP with a quasi-periodic kernel. A Matern kernel was added to capture the remaining non-periodic variation.

3.4. Optimal Parameter Determination for Each Algorithm

To allow for a fair comparison of the individual algorithm, the parameters of each method were optimized individually for the sample at hand. The optimization goal was to maximize the number of detections in the sample (as defined in Section 4). For algorithms with multiple degrees of freedom, all combinations were explored extensively. As an example, we show the result for the Savitzky–Golay method in Figure 4. This method has two parameters: the width of the sliding window w (in cadences), and the degree of the polynomial (p). For $p = 2$, a global maximum is found near $w = 71$ (as the number of missed detections, 16, is minimal), and a subsequent finer grid around this value (not shown) confirms this location. For $p = 3$, 4, 5, ...10 (and many trial windows), the number of misses increases and thus $p = 2$, $w = 71$ is the best setting for a Savitzky–Golay filter for this particular sample.

The window size is a trade-off between local removals of trends, versus the protection of transit-like signals. With the transit duration being mainly a function of planetary period, the window size must be increased when searching for longer transits. Thus, the optimal window size is determined by the period distributions in our experiments (see also Section 3).

Our visualizations make use of the boxplot, where the box extends from the lower to upper quartile value. The median and mean values are indicated with a blue line and a red dashed line. The whiskers extend to the [10, 90] percentiles. Between the 10 percentile and the zero SDE value, we print the number of missed detections (if any).

4. Results

We now compare the results of each algorithm using their optimal settings per data set, unless noted otherwise.

4.1. Robustness Sample of Real Planets in K2

The most severe data quality issue in our *K2* experiment was low outliers caused by the rolling of the spacecraft due to the loss of a reaction wheel. The resulting instrumental flux drops are only partially corrected in the `EVEREST` pipeline. While high outliers can be removed at the small cost of skewing the distribution of data points around the normalized (or local) flux, low points cannot simply be removed—they are often the signal we seek. If these low outliers are kept as is, ∼20% of our simulated transiting planet sample is undetectable. We have determined an optimal threshold to clip low outliers. By testing each algorithm (using individual optimal parameters) against different values of σ with an outlier clipping below various multiples of the standard deviation, a clipping of outliers that were 10σ below the mean after detrending turned out to be optimal. In this case, only $\geqslant$6 planets were missed in our injection-retrieval experiment, depending on the detrending. All planets can be recovered using the "biweight" detrending in two runs, one without clipping, and one with 10σ clipping. This is not true for all other detrending methods, which still miss some planets in this two-pass scenario. Most importantly, the ranking of methods does not change for different choices of the clipping threshold in multiples of σ.

In the reference one-pass scenario, where the 10σ clipping has shown to be optimal (Figure 5), the "biweight" and "Welsch" location estimators are the winners with six misses (out of 100). The time-windowed median slider is only marginally worse (seven misses). A direct comparison of the time-windowed median to a cadence-based median of the same (average) length shows that the cadence-based version misses two planets more (nine instead of seven), and its mean and median SDE values are slightly worse (ΔSDE $\sim$ 0.1). The time-based slider is superior to the (typically used) cadence-based version. Other time-based sliders are worse in performance, namely the trimmed mean, the Hodges, the Huber, and the winsorized mean. The most suboptimal choice in this group is the sliding mean, which is strongly affected by outliers (e.g., in-transit points). Yet, while missing 12 planets, a sliding mean is still superior to the Savitzky–Golay, which misses 16.

The comparative performance of GPs is worse than all robust methods. Missing 10–11 planets (depending on the kernel used) yields a mediocre performance. Interestingly, the result does not improve for the robust (iterative) GPs. It did, however, change slightly the specific planets to be found, by changing the effective low-point outlier clipping.

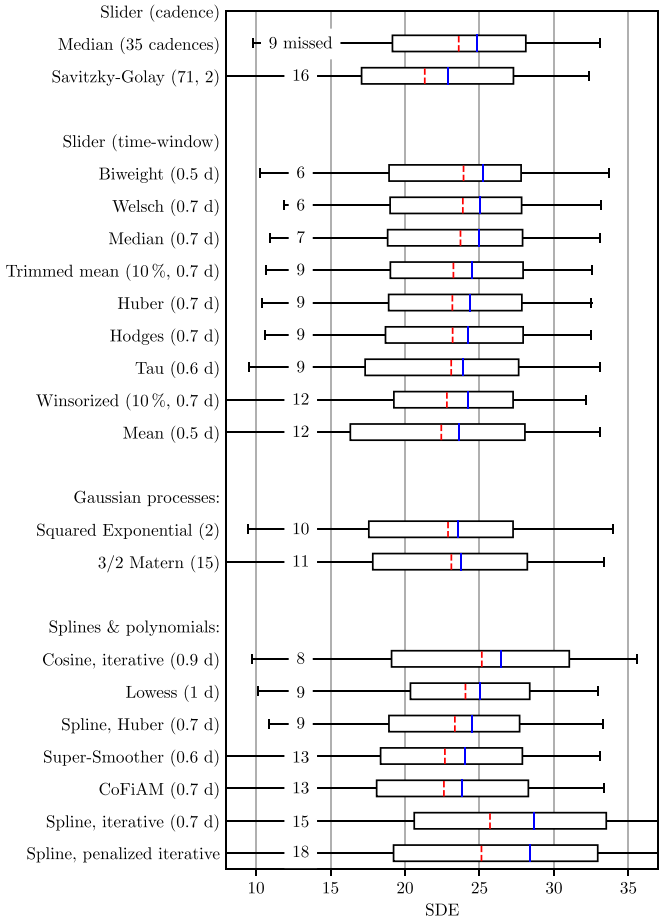

Figure 5. *K2* robustness experiment. Boxplot of all methods, using optimal parameters for each. Boxes cover the lower to upper quartiles; whiskers show the 10 and 90 percentiles. Dashed red lines indicate the median SDE, blue lines the mean SDE. The number left of the box is the number of missed detections (of 100). In the first group of cadence-based sliding windows, the median `scipy.medfilt` with a width of 35 cadences recovers 91 of 100 planets, but misses nine (text on the upper left). The winner of this experiment is a time-windowed slider with a robust estimator of type Tukey's biweight (or Welsch), missing only six planets.

Finally, splines and polynomials perform inferior to time-windowed methods. Interestingly, some methods achieve a higher mean SDE compared to the "biweight." These methods perform well for highly significant transits and allow for a detection with even higher confidence. However, they miss more of the difficult planets, something we consider the more important metric.

We have repeated the experiment using the `K2SFF` reduction from Vanderburg & Johnson (2014), which is noisier. Out of the 100 light curves with injected transits, the highest number of recoveries is 78, again achieved by the "biweight" method. The other methods perform only marginally worse, e.g., "LOWESS" recovers 76. This indicates that the leading-order difficulty in transit recovery from `K2SFF` data is noise, and not the detrending method—at least for our sample. Using `EVEREST`, however, the detrending method is very important.

4.2. Performance Sample of Shallowest Planets in Kepler

Test results from this subset are very similar to the *K2* sample. In the group of sliders, we find that the biweight is

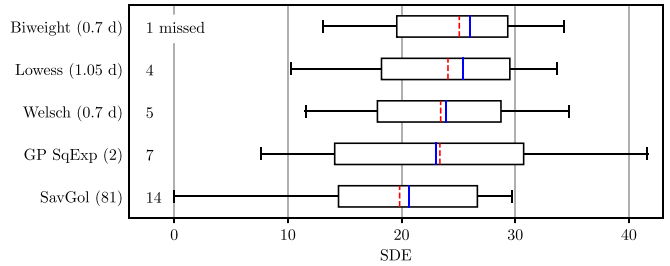

Figure 6. *Kepler* experiment showing boxplots and missed recoveries (e.g., 1 for the biweight) for a subset of all methods, due to computational expense. Again, the biweight slider is the most suitable method, while the performance of of GPs and the Savitzky–Golay filter are substantially worse. Boxes cover the lower to upper quartiles; whiskers show the 10 and 90 percentiles. Dashed red lines indicate the median SDE, blue lines the mean SDE.

slightly better than the Welsch and the median, and better by a large margin than the other estimators. The Savitzky–Golay shows a very weak performance, from which we conclude that this issue is not caused by *K2* systematics. Instead, this method should be avoided for any light-curve preparation before a transit search, as it removes a relevant fraction of transits (10%–20%). In the group of GPs, we tested only a squared exponential kernel, again with a mediocre performance. In the subgroup of splines and polynomials, we find again LOWESS with an acceptable performance, although still inferior to the best robust slider. Figure 6 shows a performance summary of the relevant methods.

Overall, it appears that the relative sensitivity (i.e., recovery rate) and robustness (i.e., SDE) of the detrending methods do not depend significantly on the data quality. Initially, we had expected that some methods might exhibit better performance than others in the presence or absence of strong outliers. This is not the case in our experiment. We were not able to extract a change of preference of the detrending methods depending on whether we explored *K2* data with strong systematics or *Kepler* data with the most shallow transits.

4.3. Injected Sample of Real Planets in Young Stars from TESS

The most surprising result from our experiment with injected transits into *TESS* light curves of young stars is the low recovery rate of all methods. It ranges from 33.6% for the sliding median to 37.2% for the robust spline (Figure 7). In less noisy data, the recovery rate of such planets with 0.5 $R_{\rm Jup}$ is close to 100%.

The second surprise is that all methods perform very similarly and that their parameter settings have little influence on the recovery rates. At first glance, this suggests that the detrending method is less important, and other factors dominate. At second glance, it is interesting that the overlap of actual planets recovered with each method is large, but not complete. Combining all five methods pushes the recovery rate to 43.8%, at the cost of five times as many vetting sheets to be examined by humans (or a suitable robo-vetter). A typical failure mode is shown in Figure 8 and discussed below (Section 4.6).

Finally, it can be noted that the sliding biweight is not the answer to all detrending questions. It performed best in the *Kepler* and *K2* experiment, but was beaten by the robust spline in young *TESS* stars. Although the difference was marginal, it reiterates the point that the best method for a given task can (and often should) be determined quantitatively.

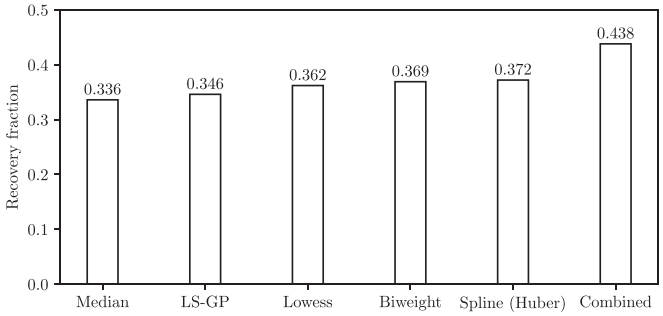

Figure 7. Injection-retrieval experiment using 316 young stars from *TESS*. In each light curve, a 0.5 $R_{\rm Jup}$ central-transit planet was injected with periods in the range 1–15 days. As stellar variability was strong and often periodic, we added a new method: a periodic Gaussian process informed by a period pre-search (based on a Lomb–Scargle periodogram), plus a Matern kernel to capture the remaining non-periodic variation. This method was optimized by varying the kernel size. As a global maximum, it recovered 34.6% of the transits. Due to computational expense, only a few methods were re-tested here. Interestingly, the robust Huber spline (with $w = 0.3$ day) slightly outperformed the sliding biweight ($w = 0.25$ day). Combining all methods increases the total yield to 43.8%.

4.4. Best Edge Treatment

We tested the influence of the possible edge treatments (see Section 2.11) using a sliding median as implemented in `scipy.medfilt`, and a time-windowed biweight. The worst results (as measured in the fraction of planets recovered) were found when wrapping the data from the beginning to the end of the time series. This should work well if the data are periodic and the period is equal to the length of the time series, which is usually not the case. Padding with constant values, and mirroring, produce very similar, usually acceptable, results. Slightly better results are achieved when the sliding window shrinks near edges. The best results, albeit only by a slight margin of ΔSDE ~ 0.1, were achieved when *discarding* data (half a window size) near edges.

Using more data improves the SDE if these data are generally acceptable. With difficult data, it is preferable to remove them instead. Discarding data near edges can be avoided by verifying by eye that the trend does diverge near the borders. Usually, this is the preferred method when presenting results in publications, and figures should include all usable data. For automatic search programs, where this decision must be made automatically, it is (slightly) preferable to discard these few points (usually of order one percent of the data).

4.5. Best Window Shape for Sliders

Throughout the paper, we have used a boxcar (rectangular) window for the time-window based methods, except for the LOWESS, which uses a tri-cube. This requires a justification. Other window options, such as a Gaussian or an exponentially weighted moving estimator, concentrate more weight toward the central moment. With correlated noise, this should improve the efficiency of the location estimate, as correlated noise decays over some finite time. In practice, however, too much of a concentration toward the center is futile. During long-duration transits, the central moment falls into the transit dip, so that the trend fits out the actual transit. This can be compensated for by making the window longer, and/or stronger emphasis on the robustness. Longer windows, however, leave substantial residue after detrending when the stellar noise periodicity is short. We have tested a subset of the robust sliders (the biweight, the mean, and the median) with Gaussian and Epanechnikov windows. In all

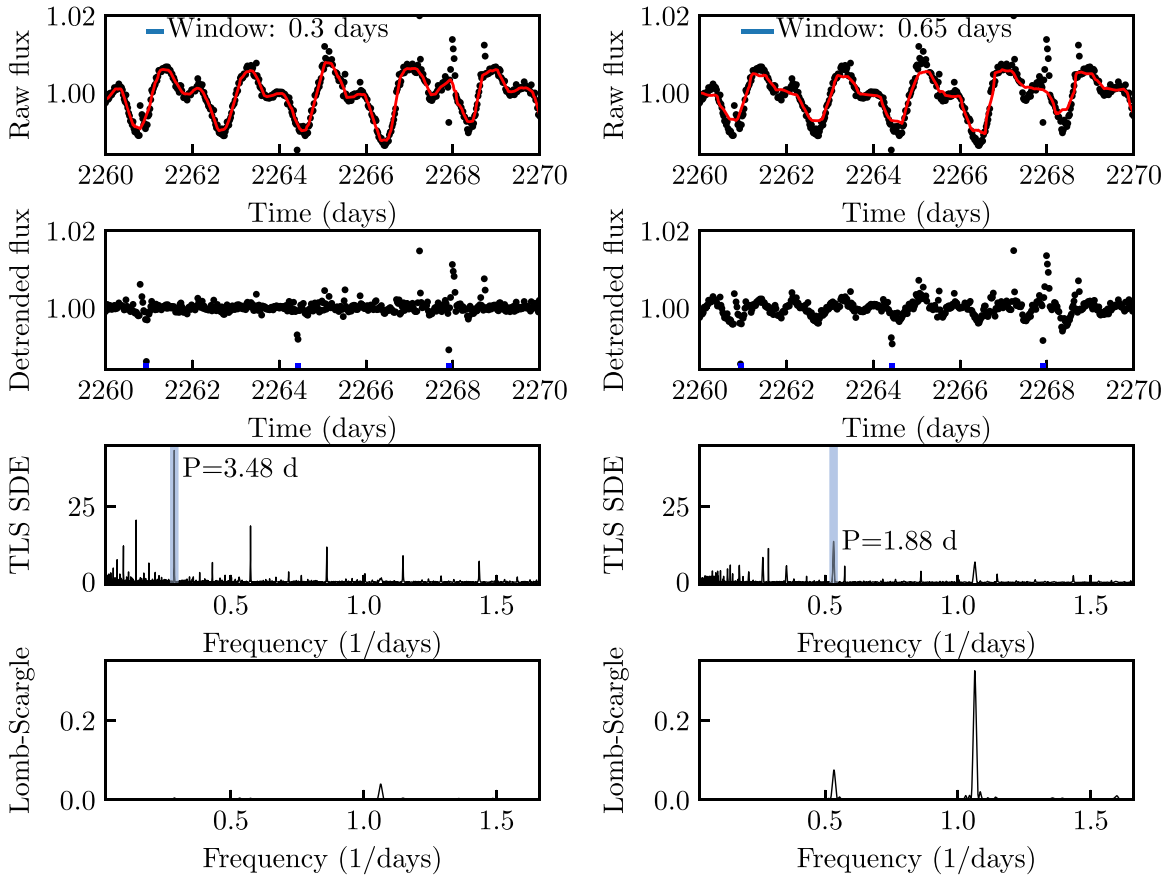

Figure 8. Detrending of K2-25 b (Mann et al. 2016b) with the biweight detrending filter and two different window sizes: short (0.3 day, left panels), which allows for a detection of the planet ($P = 3.4843$ days), long (0.65 day, bottom panels). From top to bottom: raw flux, detrended flux (both only shown for a segment of the total time series), TLS signal detection efficiency, Lomb–Scargle periodogram. The longer window leaves stellar noise at a level so that the automatic transit search detects the stellar variation instead of the transit as the most significant signal. The shorter window, however, would remove planets where $T_{14} \gtrsim 2w$, which is $P \gtrsim 12$ days for this $M = 0.3\,M_{\odot}$, $R = 0.3\,R_{\odot}$ star and a $R_{\rm Jup}$ planet.

cases, the recovery results were worse. Thus, we recommend to use the rectangular window or, as the case may be, the tri-cube weighting as in the LOWESS method.

4.6. Optimal Window Length

The window length (kernel size, knot distance...) should be as short as possible to maximize the removal of stellar variability (Figure 8), but as long as necessary to preserve the transits. With another experiment, we now determine the optimal window size for the time-windowed biweight slider.

In this separate experiment, we re-use the *Kepler* sample of the 100 lowest-S/N planets. Now the window size was set for each planet individually as a multiple of its published transit duration T_{14}, as queried from the NASA Exoplanet Archive. As shown in Figure 9, the highest minimum (and mean) SDE is achieved for a window of size $3T_{14}$. This choice recovers all planets. We have also checked the limiting case of white noise using synthetic injections and retrievals into Gaussian noise. Then, the window can be slightly shorter ($w/T_{14} \gtrsim 2.2$) to preserve almost all ($\gtrsim$98%) of the flux integral (Figure 10).

If we instead use a fixed window for all stars, we may reasonably decide to choose it so that the planet with the longest transit duration is still preserved, which is $T_{14} \sim 0.67$ day for Kepler-1638b with $P = 259$ days. Selecting $w = 2.5T_{14} \sim 1.65$ days, we run another search with this window

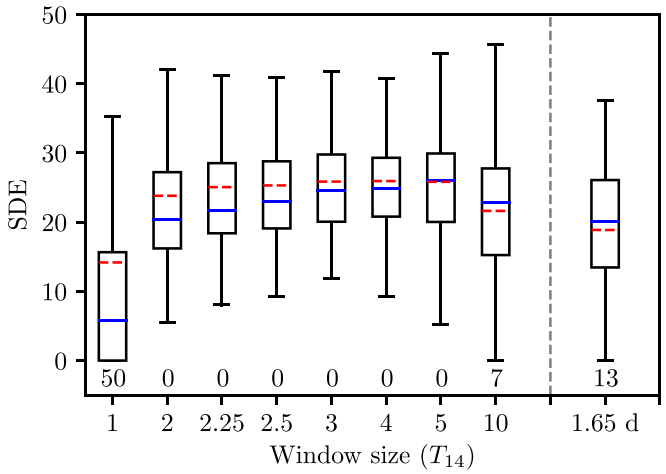

Figure 9. *Kepler* sample of the 100 lowest-S/N planets, searched with TLS using the sliding biweight. In this separate experiment, the window size was adjusted for each planet individually as a multiple of its known transit duration T_{14}. The highest minimum (and mean) SDE is achieved for a window of size $3T_{14}$. The last boxplot shows an alternative experiment, where the window is fixed to 1.65 days for all planets, because 1.65 days is 2.5× the transit duration of the planet with the longest transit duration in the sample (0.67 day, Kepler-1638b, $P = 259$ days). Clearly, an adaptive window (per star) is superior to a fixed size. Boxes cover the lower to upper quartiles; whiskers show the 10 and 90 percentiles. Dashed red lines indicate the median SDE, blue lines the mean SDE. Numbers at the bottom show the percentage of missed planet detections.

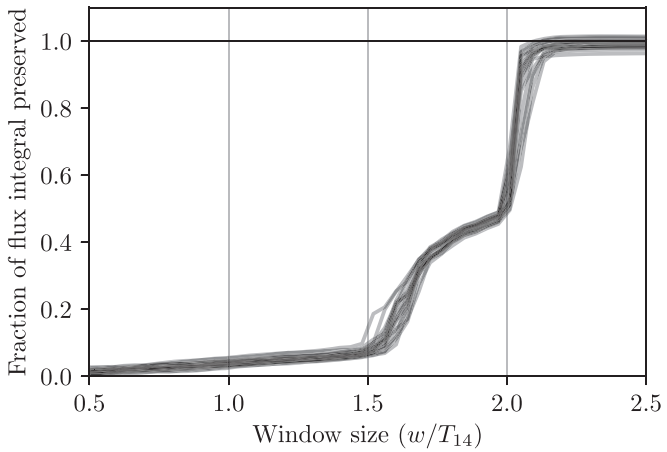

Figure 10. Fraction of the flux integral preserved as a function of the window size (in units of T_{14}) for a sliding biweight. Data from an injection-retrieval experiment of Mandel–Agol transit model into Gaussian noise where the transit depth is 10× the standard deviation per data point. For $w/T_{14} \gtrsim 2.2$, most ($\gtrsim$98%) of the flux integral is preserved.

for all stars in the sample. The recovery fraction is now only 87%, losing 13 planets. Most of these have short periods (with corresponding short transit durations) and strong stellar variation, which is not sufficiently removed with a 1.65 days window. This shows that an adaptive window (per star, per period, per transit duration) is superior to a single fixed size. But there is a problem: for a blind search, the period is unknown, and thus T_{14} is unknown. However, we can estimate the transit duration it as a function of period given stellar mass and radius. As a sensible limit, we can use a large planet ($2R_{\rm Jup}$) and a circular orbit, and find (see Equation (10) in Hippke & Heller 2019)

$$T_{14,\max} = (R_s + R_p)\left(\frac{4P}{\pi G M_s}\right)^{1/3}. \quad (8)$$

With $w/T_{14} = 2.2$ we have $w = 1.54$ days for the longest periods, but $w = 0.2$ days for a short period ($P = 1$ day) (the longest transit durations from the *Kepler* mission are actually $T_{14} \sim 1$ day (e.g., Kepler-849; Morton et al. 2016)). Simply choosing a unique long window length for all searches would result in sub-optimal detrending, leading to reduced sensitivity: for the shorter-period planets, the sliding window can be 10× shorter. In the future, improved transit search algorithms should include a custom detrending for each period. The resulting gain in the computational expense could be mitigated by defining clusters of ranges of periods.

5. Discussion

5.1. Which Method to Choose?

The subject of this paper is the removal of stellar trends with simultaneous preservation of transit signals that will be detectable with `TLS`. Our results may apply to other problems as well, where the signal occupies small segments of a time series of measurements. Other problems such as asteroseismology, however, are so different in nature or phenomenology from the transit problem considered in this work that the methods explained here may be inapplicable. For example, when filtering light curves with the aim of finding phase curve variations, the window size must be much longer than for transit searches (but short enough to remove stellar rotation). Again, the right detrender can be chosen through a comparison of several methods based on simulated or partly simulated data as done in this paper.

5.2. Issues with GPs

GPs have performed better than a sliding mean, but worse than robust estimators including a simple sliding median in this experiment. Why is that the case?

The fact that the performance of GP as a detrender is similar to that of the sliding mean gives a first clue to the problem. GPs are built, as their name suggests, on the assumption of data which have a Gaussian distribution. This is obviously not the case in the presence of substantial short-term astrophysical activity such as transits. The GP has no way of "knowing" that the in-transit dip is not part of the trend in the light curve that it is supposed to fit. Based on least-squares calculations, the problem becomes more severe for stronger signals (see Figure 11 for a visualization). There are two choices to tackle this issue.

The first is the classical clip-and-repeat method, similar to a multi-pass trimmed mean, where outliers are clipped and the fit is repeated until convergence. Of course, this clipping is applied only temporarily to estimate the trend: when dividing the raw data by the trend, the in-transit points are still present. Compared to other robust (single-pass) methods, the issue is that not only are the in-transit points clipped (the desired behavior), but also some of the points before ingress and after egress (middle panel in Figure 11). This reduction of (otherwise good and useful) out-of-transit data points decreases the fit quality. It can be visually seen in this example, but is usually invisible in practice, where individual transits are drowned by noise (but still present). This is why iterative GPs (and splines) are inferior to robust methods for the purpose of transit searches.

Other methods to make GPs more robust are currently being studied. Our literature review revealed a range of methods such as the mixture of two Gaussians (Tresp 2001; Rasmussen & Ghahramani 2002), kernel ridge regressions, and outlier identification (Kuss & Rasmussen 2005). Since the essential structure of the GP remains unchanged, in all these cases the noise is assumed (near) normal, and none may be appropriate when errors appear with their own structure. We are not aware of any work that explicitly targets the problem of clustered outliers.

5.3. Computational Expense

The computational expense required for our search of hundreds of transits repeatedly throughout the full parameter space of various detrending methods was significant. While most of the computing power was devoted to the actual transit search, it is crucial to ensure that the detrending itself is as fast as possible, because it is performed for millions of light curves in current and future surveys.

We have measured the computational speed of the implementations used in this work (Figure 12). The time-windowed sliding mean is the fastest, despite its naive implementation where it is re-calculated at each position (similarly to the median, trimmed mean, and winsorized mean). The biweight (and similarly Andrew's sine wave and the Welsch estimator) are slower. A robust spline has higher initiation cost, but a better performance toward large data. The

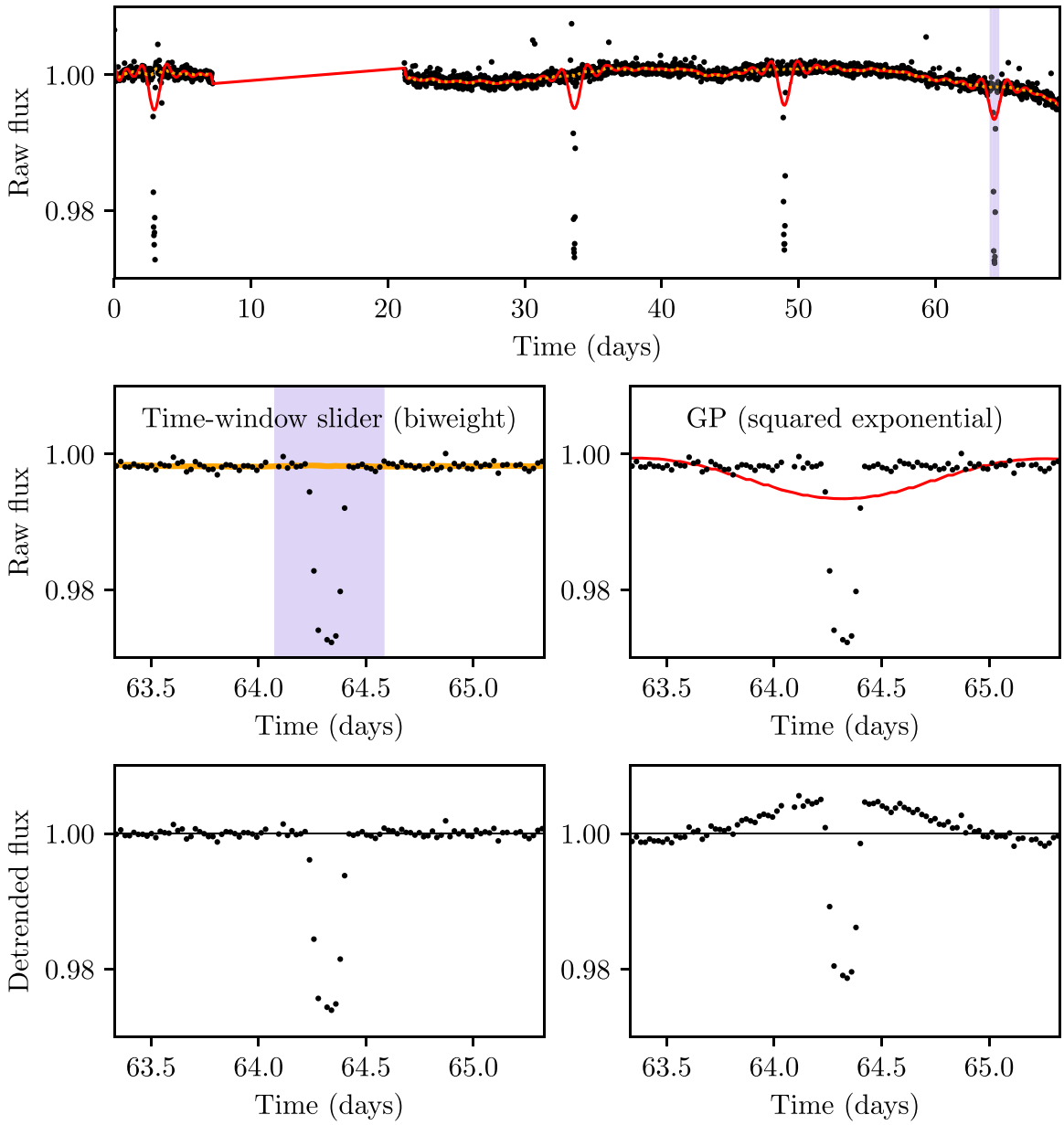

Figure 11. Detrending for EPIC 228707509 (Livingston et al. 2018) using the time-windowed biweight (top and left, orange lines) with its optimal window size of 0.5 day (blue shade), and a Gaussian process with a squared exponential kernel of optimal size (top and right, red lines). *K2* data are shown with black points; the detrending functions are illustrated as red lines. The detrending function determined with GP exhibits a substantial long-trend dip caused by the deep transit, which creates extra noise in the detrended light curve and reduces sensitivity in a transit search. An iterative approach reduces the problem in this specific example, but does not eliminate it. Overfitting can be avoided, in cases where the transit ephemeris is known, by masking the in-transit points.

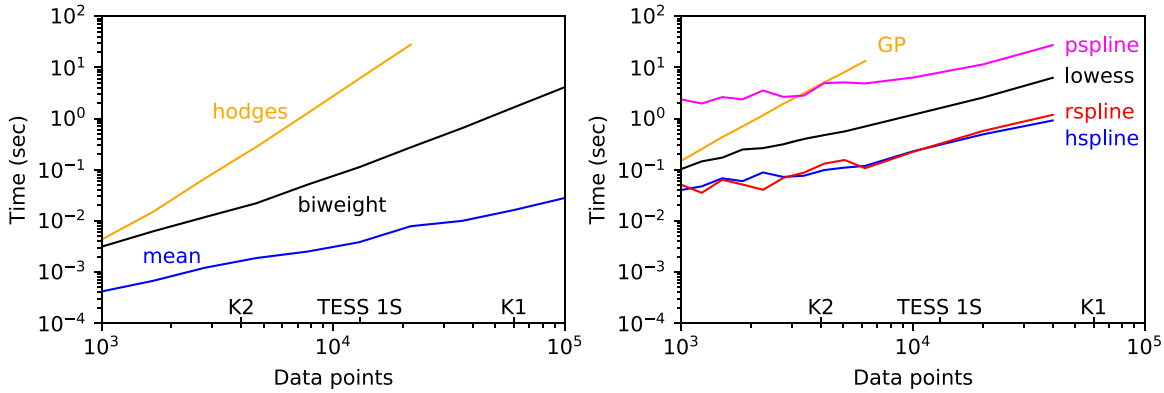

Figure 12. Run-time comparison of various detrending algorithms. Left: sliders; right: LOWESS, GP, and splines (pspline: penalized; rspline: robust iterative; hspline: Huber estimator).

slider with the Hodges estimator and the GPs are of acceptable speed ($\lesssim$10 s) for data with $\leqslant 10^4$ points, i.e., *TESS* 1-sector SC and *K2* LC. The other methods here are fine for data with $\leqslant 10^5$ points, which includes the complete *Kepler* LC set.

Some methods like the iterative biweight benefit from a run-time compilation which transforms the pure Python code into a lower-level machine language before execution. We chose to implement many of the time-critical parts of `Wōtan` with the specialized `numba` package (Lam et al. 2015). This procedure is called "just-in-time" compilation and brings a 30-fold gain in performance.

Generally, many methods can be implemented in a faster way if required. It is straightforward to see that a sliding mean filter does not require a re-calculation of the points inside the window at each move. With one value entering the window, and one value being dropped, it is trivial to re-calculate the mean efficiently. For the median, however, naive (but common) implementations do recalculate the in-window points. It was shown by Hardle & Steiger (1995) that this is not necessary. Instead, the values in the window need to be sorted only once. Afterwards, each new (dropped) value is inserted (removed) at the known spot. This operation can be done in $\mathcal{O}(\log_2 n)$ using the bisection algorithm (Knuth 1998, 6.2.1). Open-source implementations are available[12] (Suomela 2014). This method is commonly known as the "Turlach," after the implementation by Berwin A. Turlach[13] (1995, unpublished). Interestingly, this method is re-invented regularly (e.g., Handberg & Lund 2014). It has also been shown that a robust (truncated) sliding mean (robust only in x) is possible (Jonsson & Felsberg 2016), but it is unclear if it is possible to extend it for y and/or other robust estimators.

In summary, most algorithms implemented in `Wōtan` scale as $\mathcal{O}(n)$ for small window sizes, and $\mathcal{O}(n^2)$ if the window is a significant fraction of the data. The only exceptions are the Hodges slider (which is in $\mathcal{O}(n^3)$) and GPs ($\mathcal{O}(n^3)$ or $\mathcal{O}(n \log^2 n)$; see Ambikasaran et al. 2015, depending on the implementation).

5.4. Masking Known Transits

Our work is focused on exoplanet discovery, and during such a search it is unknown which data points are in-transit (if any). Thus, the trend removal filter should cause minimal distortions to the transit shape, while removing as much stellar and instrumental variation as possible. All filters tested here achieve this task only imperfectly. Some filters, such as the biweight, are more resistant to fitting out transits than others (such as the Savitzky–Golay). Still, any filter will, on average, cause some (usually unwanted) distortion on the transit shape.

Such distortions can be avoided (or at least maximally minimized) by masking in-transit points when known. In a post-detection analysis of the planet parameters, it is best practice to mask the in-transit points (see, e.g., Figure 4 in Luger et al. 2018). `Wōtan` offers a `transit_mask` feature for that use case.

5.5. The Way Into the Future

While we have given a comprehensive overview of the most widely used detrending methods for stellar light curves, there are certainly further, less well known filters that could be tested. For example, Rizzuto et al. (2017) fit a polynomial plus a periodic notch-filter, which is a box-shaped region of data points that are masked from the trend. By iterating over the notch width, period, and phase, planetary in-transit points can be blanked so that they do not affect the fit. A Bayesian model selection is used to determine the filter. This is half-way toward a simultaneous search and detrending, with the search being a box-shaped filter (BLS; Kovács et al. 2002). Such an approach has been tested by Foreman-Mackey et al. (2015). A disadvantage of this method is that the computational expense is very high, but with Moore's law at work (Moore 1965), it may be a beneficial approach for young, very active stars.

To further improve splines, current research is making progress toward L1 smoothers which are tailored for skewed distributions (Rytgaard 2016).

By having optimized planet-finding using known planets, we may have biased the optimization toward their parameter space. For example, we have estimated transit duration assuming circular orbits, whereas eccentric planets can have longer (or shorter) transit durations. Therefore, it can be justified to choose parameters outside of the "optimal" ranges listed here, for experimental and explorative purposes (Loeb 2010). More philosophically: adjusting a boat to perform well in the oceans we know does not necessarily make it fit for all unknown waters, whose nature we do not know. As of now, however, it is the best we can do. Finding more planets in the known parameter space is a good start.

In the future, it may be possible to adjust the method and/or their parameters on a per-light-curve basis using machine learning (Hinners et al. 2018; Pearson et al. 2018). With such an approach, over- and underfitting could be reduced, increasing the planet yield. Alternatively, one may consider to circumvent detrending entirely and use machine learning to search for transits.

6. Conclusion

We have investigated, for the first time, the relative performance of various detrending methods of stellar light curves with stellar activity and outliers on the detectability of exoplanetary transits. We devised and executed an injection–retrieval experiment of simulated transits into *Kepler*, *K2*, and *TESS* light curves with a simulated search using the `TLS` algorithm. As a consequence, our results can be used to efficiently search for transits in stellar light curves from these surveys. Determining empirically which tool works best for a given job should be the norm, and our framework can serve as an example. Our toolkit for time-series detrending, dubbed `Wōtan`, is open source under the MIT license and is available at https://github.com/hippke/wotan with documentation and tutorials. It can serve as a one-stop solution for various smoothing tasks, and its open-source nature offers the possibility to the community to contribute improvements.

Based on our experiments for transit detection, we find that a time-windowed slider with an iterative robust location estimator based on Tukey's biweight is an optimal choice in many cases. This is usually superior to the median or trimmed methods. In the presence of a lot of stellar variation, spline-based methods work comparably well. We also determined that the optimal width of the sliding window is about three times the transit duration that is searched for.

[12] For example https://github.com/suomela/median-filter.
[13] https://svn.r-project.org/R/trunk/src/library/stats/src/Trunmed.c

We thank Adina D. Feinstein for assistance with *TESS* light curves, Kai Rodenbeck and Alex Teachey for providing code to implement the CoFiAM algorithm, David Kipping for corrections about the details of the CoFiAM method, Andrew Vanderburg for help to with K2SFF files, and Aviv Ofir for ideas on a sum-of-sines method with iterative sigma clipping. This project was developed in part at the *Building Early Science with TESS* meeting, which took place in 2019 March at the University of Chicago. R.H. is supported by the German space agency (Deutsches Zentrum für Luft- und Raumfahrt) under PLATO Data Center grant 50OO1501. T.J.D. gratefully acknowledges support from the Jet Propulsion Laboratory Exoplanetary Science Initiative. Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA.

Software: eleanor (Feinstein et al. 2019), numpy (Van Der Walt et al. 2011), numba (Lam et al. 2015), scipy (Jones et al. 2001), scikit-learn (Pedregosa et al. 2011), astropy (Astropy Collaboration et al. 2013; Price-Whelan et al. 2018), statsmodel (Seabold & Perktold 2010), george (Ambikasaran et al. 2015; Foreman-Mackey et al. 2017a), celerite (Foreman-Mackey et al. 2017b; Foreman-Mackey 2018), pyGAM (Serven & Brummitt 2018), matplotlib (Hunter 2007), iPython (Pérez & Granger 2007), Jupyter Notebooks (Kluyver et al. 2016), TLS (Hippke & Heller 2019).

ORCID iDs

Michael Hippke https://orcid.org/0000-0002-0794-6339
Trevor J. David https://orcid.org/0000-0001-6534-6246
René Heller https://orcid.org/0000-0002-9831-0984

References

Ahlberg, J. H., Nilson, E. N., & Walsh, J. L. 1967, Mathematics in Science and Engineering (New York: Academic)
Aigrain, S., Hodgkin, S. T., Irwin, M. J., et al. 2015, MNRAS, 447, 2880
Aigrain, S., & Irwin, M. 2004, MNRAS, 350, 331
Aigrain, S., Parviainen, H., & Pope, B. J. S. 2016, MNRAS, 459, 2408
Akeson, R. L., Chen, X., Ciardi, D., et al. 2013, PASP, 125, 989
Ambikasaran, S., Foreman-Mackey, D., Greengard, L., et al. 2015, ITPAM, 38, 252
Andrews, D. F., Bickel, P. J., Hampel, F. R., et al. 1972, Robust Estimates of Location: Survey and Advances (Princeton, NJ: Princeton Univ. Press), 3
Angus, R., Morton, T., Aigrain, S., et al. 2018, MNRAS, 474, 2094
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Auvergne, M., Bodin, P., Boisnard, L., et al. 2009, A&A, 506, 411
Bakos, G., Noyes, R. W., Kovács, G., et al. 2004, PASP, 116, 266
Barclay, T., Endl, M., Huber, D., et al. 2015, ApJ, 800, 46
Barentsen, G., Hedges, C., Vincius, Z., et al. 2019, KeplerGO/lightkurve: Lightkurve v1.0b29, Zenodo, doi:10.5281/zenodo.2565212
Bartels, J. 1934, TeMAE, 39, 201
Batalha, N. M., Borucki, W. J., Bryson, S. T., et al. 2011, ApJ, 729, 27
Batalha, N. M., Rowe, J. F., Bryson, S. T., et al. 2013, ApJS, 204, 24
Beck, J. G. 2000, SoPh, 191, 47
Becker, A. C., Bochanski, J. J., Hawley, S. L., et al. 2011, ApJ, 731, 17
Beers, T. C., Flynn, K., & Gebhardt, K. 1990, AJ, 100, 32
Bonomo, A. S., Aigrain, S., Bordé, P., & Lanza, A. F. 2009, A&A, 495, 647
Borucki, W. J., Koch, D., Basri, G., et al. 2010, Sci, 327, 977
Brahm, R., Espinoza, N., Jordán, A., et al. 2019, AJ, 158, 45
Brandt, T. D., & Huang, C. X. 2015a, ApJ, 807, 58
Brandt, T. D., & Huang, C. X. 2015b, ApJ, 807, 24
Brewer, B. J., & Stello, D. 2009, MNRAS, 395, 2226
Brown, T. M., Latham, D. W., Everett, M. E., & Esquerdo, G. A. 2011, AJ, 142, 112
Buchholz, W., & Shapiro, F. 2000, IEEE Annals of the History of Computing, 22, 70
Burke, C. J., Bryson, S. T., Mullally, F., et al. 2014, ApJS, 210, 19
Caceres, G. A., Feigelson, E. D., Jogesh Babu, G., et al. 2019, AJ, 158, 57
Cañas, C. I., Wang, S., Mahadevan, S., et al. 2019, ApJL, 870, L17
Cappellari, M., McDermid, R. M., Alatalo, K., et al. 2013, MNRAS, 432, 1862
Carter, J. A., & Winn, J. N. 2009, ApJ, 704, 51
Chambers, J. M., & Hastie, T. J. 1992, Statistical models in S, Vol. 251 (Pacific Grove, CA: Wadsworth)
Charbonneau, D., Berta, Z. K., Irwin, J., et al. 2009, Natur, 462, 891
Charbonnier, P., Blanc-Feraud, L., Aubert, G., & Barlaud, M. 1997, ITIP, 6, 298
Claret, A. 2018a, A&A, 618, A20
Claret, A. 2018b, yCat, J/A+A/618/A20, 0
Cleveland, W. S. 1979, J. Am. Stat. Assoc., 74, 829
Cleveland, W. S. 1981, The American Statistician, 35, 54
Cody, A. M., & Hillenbrand, L. A. 2018, AJ, 156, 71
Cody, A. M., Hillenbrand, L. A., David, T. J., et al. 2017, ApJ, 836, 41
Cooley, J. W., & Tukey, J. W. 1965, MaCom, 19, 297
Crossfield, I. J. M., Ciardi, D. R., Petigura, E. A., et al. 2016, ApJS, 226, 7
Danielski, C., Kacprzak, T., & Tinetti, G. 2013, EPSC, 8, EPSC2013-599
Davenport, J. R. A. 2016, ApJ, 829, 23
David, T. J., Cody, A. M., Hedges, C. L., et al. 2019, AJ, 158, 79
David, T. J., Hillenbrand, L. A., Petigura, E. A., et al. 2016, Natur, 534, 658
de Boor, C. 1980, MaCom, 34, 325
Dennis, J. E., & Welsch, R. E. 1978, Communications in Statistics—Simulation and Computation, 7, 345
Dorn-Wallenstein, T. Z., Levesque, E. M., & Davenport, J. R. A. 2019, ApJ, 878, 155
Eilers, P. H. C., & Marx, B. D. 1996, StaSc, 11, 89
Epanechnikov, V. A. 1969, Theory of Probability & Its Applications, 14, 153
Fabrycky, D. C., Ford, E. B., Steffen, J. H., et al. 2012, ApJ, 750, 114
Feinstein, A. D., Montet, B. T., Foreman-Mackey, D., et al. 2019, PASP, 131, 094502
Ferguson, J. 1964, JACM, 11, 221
Foreman-Mackey, D. 2018, RNAAS, 2, 31
Foreman-Mackey, D., Agol, E., Ambikasaran, S., & Angus, R. 2017a, AJ, 154, 220
Foreman-Mackey, D., Agol, E., Ambikasaran, S., & Angus, R. 2017b, Celerite: Scalable 1D Gaussian Processes in C++, Python, and Julia, v0.3.1, Astrophysics Source Code Library, ascl:1709.008
Foreman-Mackey, D., Montet, B. T., Hogg, D. W., et al. 2015, ApJ, 806, 215
Friedman, J. H. 1984, A Variable Span Smoother, Tech. Rep., 5, Stanford Univ. CA Lab for Computational Statistics
Friedman, J. H., & Silverman, B. W. 1989, Technometrics, 31, 3
Gagné, J., Mamajek, E. E., Malo, L., et al. 2018, ApJ, 856, 23
García, R. A., Mathur, S., Salabert, D., et al. 2010, Sci, 329, 1032
Gautier, T. N. I., Charbonneau, D., Rowe, J. F., et al. 2012, ApJ, 749, 15
Gibson, N. P., Aigrain, S., Roberts, S., et al. 2012, MNRAS, 419, 2683
Giles, H. A. C., Bayliss, D., Espinoza, N., et al. 2018, MNRAS, 475, 1809
Gilliland, R. L., Chaplin, W. J., Dunham, E. W., et al. 2011, ApJS, 197, 6
Grunblatt, S. K., Huber, D., Gaidos, E., et al. 2017, AJ, 154, 254
Grziwa, S., & Pätzold, M. 2016, arXiv:1607.08417
Hampel, F., Ronchetti, E., Rousseeuw, P., & Stahel, W. 2011, Robust Statistics: The Approach Based on Influence Functions (New York: Wiley)
Handberg, R., & Lund, M. N. 2014, MNRAS, 445, 2698
Hardle, W., & Steiger, W. 1995, Applied Statistics, 44, 258
Hastie, T., Tibshirani, R., & Friedman, J. 2009, The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.; New York: Springer)
He, X., & Portnoy, S. 1992, AnSta, 20, 2161
Heller, R. 2019, A&A, 628, A42
Heller, R., Rodenbeck, K., & Hippke, M. 2019, A&A, 625, A31
Hinners, T. A., Tat, K., & Thorp, R. 2018, AJ, 156, 7
Hippke, M. 2015, ApJ, 806, 51
Hippke, M., & Angerhausen, D. 2015, ApJ, 810, 29
Hippke, M., & Heller, R. 2019, A&A, 623, A39
Hodges, J. L., & Lehmann, E. L. 1963, Ann. Math. Statist., 34, 598
Howell, S. B., Sobeck, C., Haas, M., et al. 2014, PASP, 126, 398
Huang, J. 1999, Statistics & Probability Letters, 42, 185
Huber, D., Bryson, S. T., Haas, M. R., et al. 2016, ApJS, 224, 2
Huber, D., Carter, J. A., Barbieri, M., et al. 2013, Sci, 342, 331
Huber, D., Silva Aguirre, V., Matthews, J. M., et al. 2014, ApJS, 211, 2
Huber, P. 1981, Robust Statistics (New York: Wiley)
Huber, P., & Ronchetti, E. 2011, Robust Statistics (New York: Wiley)
Huber, P. J. 1964, Ann. Math. Statist., 35, 73
Hunter, J. D. 2007, CSE, 9, 90
Jones, E., Oliphant, T., & Peterson, P. 2001, SciPy: Open Source Scientific Tools for Python, http://www.scipy.org

Jonsson, E., & Felsberg, M. 2016, arXiv:1601.08003
Kafadar, K. 1983, JRNBS, 88, 105
Kenney, J., & Keeping, E. 1947, Mathematics of statistics, Mathematics of Statistics No. Teil 1 (Princeton, NJ: Van Nostrand-Reinhold)
Kim, A. G., Thomas, R. C., Aldering, G., et al. 2013, ApJ, 766, 84
Kim, D.-W., Protopapas, P., Alcock, C., et al. 2009, MNRAS, 397, 558
Kipping, D. M., Hartman, J., Buchhave, L. A., et al. 2013, ApJ, 770, 101
Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, Jupyter Notebooks (The Netherlands: IOS Press)
Knuth, D. 1998, The Art of Computer Programming: Sorting and Searching, The Art of Computer Programming (Reading, MA: Addison-Wesley)
Kooperberg, C., & Stone, C. J. 1991, Computational Statistics & Data Analysis, 12, 327
Kovács, G., Bakos, G., & Noyes, R. W. 2005, MNRAS, 356, 557
Kovács, G., Hartman, J. D., & Bakos, G. Á. 2016, A&A, 585, A57
Kovács, G., Zucker, S., & Mazeh, T. 2002, A&A, 391, 369
Kraus, A. L., Herczeg, G. J., Rizzuto, A. C., et al. 2017, ApJ, 838, 150
Kreidberg, L. 2015, PASP, 127, 1161
Kuss, M., & Rasmussen, C. E. 2005, JMLR, 6, 1679
Lam, S. K., Pitrou, A., & Seibert, S. 2015, in LLVM '15 Proc. Second Workshop on the LLVM Compiler Infrastructure in HPC (New York: ACM), 7
Lawson, C. L. 1961, PhD Dissertation, Univ. Calif.
Leclerc, Y. G. 1989, Int. J. Comput. Vis., 3, 73
Libralato, M., Nardiello, D., Bedin, L. R., et al. 2016, MNRAS, 463, 1780
Livingston, J. H., Endl, M., Dai, F., et al. 2018, AJ, 156, 78
Loeb, A. 2010, Natur, 467, 358
Luger, R., Agol, E., Foreman-Mackey, D., et al. 2019, AJ, 157, 64
Luger, R., Agol, E., Kruse, E., et al. 2016, AJ, 152, 100
Luger, R., Kruse, E., Foreman-Mackey, D., et al. 2018, AJ, 156, 99
Luger, R., Sestovic, M., Kruse, E., et al. 2017, NatAs, 1, 0129
Lund, M. N., Handberg, R., Davies, G. R., et al. 2015, ApJ, 806, 30
Madhusudhan, N., Agúndez, M., Moses, J. I., & Hu, Y. 2016, SSRv, 205, 285
Mamajek, E. E., Quillen, A. C., Pecaut, M. J., et al. 2012, AJ, 143, 72
Mandel, K., & Agol, E. 2002, ApJL, 580, L171
Mann, A. W., Gaidos, E., Mace, G. N., et al. 2016a, ApJ, 818, 46
Mann, A. W., Gaidos, E., Vanderburg, A., et al. 2017, AJ, 153, 64
Mann, A. W., Newton, E. R., Rizzuto, A. C., et al. 2016b, AJ, 152, 61
Mathur, S., Huber, D., Batalha, N. M., et al. 2017, ApJS, 229, 30
Mayo, A. W., Vanderburg, A., Latham, D. W., et al. 2018, AJ, 155, 136
Mazeh, T., & Faigler, S. 2010, A&A, 521, L59
McGill, R., Tukey, J. W., & Larsen, W. A. 1978, The American Statistician, 32, 12
Moldovan, R., Matthews, J. M., Gladman, B., et al. 2010, ApJ, 716, 315
Montet, B. T., Tovar, G., & Foreman-Mackey, D. 2017, ApJ, 851, 116
Moore, G. E. 1965, Electronics, 38, 114
Morton, T. D., Bryson, S. T., Coughlin, J. L., et al. 2016, ApJ, 822, 86
Mosteller, F., & Tukey, J. 1977, Data Analysis and Regression: A Second Course in Statistics (Reading, MA: Addison-Wesley)
Neal, R. M. 1997, arXiv:physics/9701026
Niraula, P., Redfield, S., de Wit, J., et al. 2018, arXiv:1812.09227
Obermeier, C., Henning, T., Schlieder, J. E., et al. 2016, AJ, 152, 223
Paudel, R. R., Gizis, J. E., Mullan, D. J., et al. 2018, ApJ, 858, 55
Pearson, K. A., Palafox, L., & Griffith, C. A. 2018, MNRAS, 474, 478
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, JMLR, 12, 2825
Pepper, J., Gillen, E., Parviainen, H., et al. 2017, AJ, 153, 177
Pepper, J., Pogge, R. W., DePoy, D. L., et al. 2007, PASP, 119, 923
Pérez, F., & Granger, B. E. 2007, CSE, 9, 21
Pollacco, D. L., Skillen, I., Collier Cameron, A., et al. 2006, PASP, 118, 1407
Pont, F., Zucker, S., & Queloz, D. 2006, MNRAS, 373, 231
Pope, B. J. S., Parviainen, H., & Aigrain, S. 2016, MNRAS, 461, 3399
Press, W. 1992, Numerical Recipes in C: The Art of Scientific Computing (Cambridge: Cambridge Univ. Press)
Price-Whelan, A. M., Sipócz, B. M., Günther, H. M., et al. 2018, AJ, 156, 123
Quintana, E. V., Rowe, J. F., Barclay, T., et al. 2013, ApJ, 767, 137
Rasmussen, C. E., & Ghahramani, Z. 2002, in Advances in Neural Information Processing Systems 14, ed. T. G. Dietterich, S. Becker, & Z. Ghahramani (Cambridge, MA: MIT Press), 881
Rauer, H., Catala, C., Aerts, C., et al. 2014, ExA, 38, 249
Rice, J. R., & Usow, K. H. 1968, MaCom, 22, 118
Rice, K., Malavolta, L., Mayo, A., et al. 2019, MNRAS, 484, 3731
Ricker, G. R., Winn, J. N., Vanderspek, R., et al. 2015, JATIS, 1, 014003
Rizzuto, A. C., Mann, A. W., Vanderburg, A., et al. 2017, AJ, 154, 224
Rodenbeck, K., Heller, R., Hippke, M., & Gizon, L. 2018, A&A, 617, A49
Rousseeuw, P. J., & Leroy, A. M. 1987, Robust Regression and Outlier Detection (New York: Wiley)
Rowe, J. F., Bryson, S. T., Marcy, G. W., et al. 2014, ApJ, 784, 45
Rytgaard, H. C. 2016, Master's thesis, Institut for Matematiske, Univ. Copenhagen
Savitzky, A., & Golay, M. J. E. 1964, AnaCh, 36, 1627
Schoenberg, I. J. 1946, QApMa, 4, 112
Schwabe, H. 1844, AN, 21, 233
Seabold, S., & Perktold, J. 2010, in Proc. 9th Python in Science Conf., ed. S. van der Walt & J. Millman (Austin, TX: SciPy), 57
Seader, S., Jenkins, J. M., Tenenbaum, P., et al. 2015, ApJS, 217, 18
Seager, S., & Mallén-Ornelas, G. 2003, ApJ, 585, 1038
Sen, P. K. 1963, Biometrics, 19, 532
Serven, D., & Brummitt, C. 2018, pyGAM: Generalized Additive Models in Python, v0.8, https://github.com/dswah/pyGAM
Shannon, C. E. 1948, Bell Syst. Tech. J., 27, 623
Sinukoff, E., Howard, A. W., Petigura, E. A., et al. 2016, ApJ, 827, 78
Smith, J. C., Stumpe, M. C., Van Cleve, J. E., et al. 2012, PASP, 124, 1000
Stassun, K. G., Oelkers, R. J., Pepper, J., et al. 2018, AJ, 156, 102
Stumpe, M. C., Smith, J. C., Van Cleve, J. E., et al. 2012, PASP, 124, 985
Suomela, J. 2014, arXiv:1406.1717
Tal-Or, L., Mazeh, T., Alonso, R., et al. 2013, A&A, 553, A30
Thompson, S. E., Coughlin, J. L., Hoffman, K., et al. 2018, ApJS, 235, 38
Thompson, S. E., Fraquelli, D., Van Cleve, J. E., & Caldwell, D. A. 2016, Kepler Archive Manual, Tech. Rep. KDMC-10008-006, Mikulski Archive for Space Telescopes—STScI
Tresp, V. 2001, in Advances in Neural Information Processing Systems 13, ed. T. K. Leen, T. G. Dietterich, & V. Tresp (Cambridge, MA: MIT Press), 654
Tukey, J. 1977, Exploratory Data Analysis, Addison-Wesley Series in Behavioral Science (Reading, MA: Addison-Wesley)
Usoskin, I. G., Mursula, K., Arlt, R., & Kovaltsov, G. A. 2009, ApJL, 700, L154
Vanderburg, A., & Johnson, J. A. 2014, PASP, 126, 948
Vanderburg, A., Latham, D. W., Buchhave, L. A., et al. 2016, ApJS, 222, 14
Vanderplas, J. 2015, Supersmoother: Efficient Python Implementation Of Friedman's Supersmoother, v0.3, Zenodo, doi:10.5281/zenodo.14475
Vanderplas, J., & Willmer, A. 2015, Supersmoother: Minor Bug Fix Release, v0.3.2, Zenodo, doi:10.5281/zenodo.28518
Van Der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, CSE, 13, 22
van Haasteren, R., & Vallisneri, M. 2014, PhRvD, 90, 104012
van Leeuwen, F., Evans, D. W., De Angeli, F., et al. 2017, A&A, 599, A32
Wang, S., Jones, M., Shporer, A., et al. 2019, AJ, 157, 51
Way, M. J., Foster, L. V., Gazis, P. R., & Srivastava, A. N. 2009, ApJ, 706, 623
Wheatley, P. J., Collier Cameron, A., Harrington, J., et al. 2010, arXiv:1004.0836
Wilcox, R. 2017, JMASM, 16, 29
Yohai, V. J., & Zamar, R. H. 1988, J. Am. Stat. Assoc., 83, 406
Zhang, H., Yu, Z., Liang, E., et al. 2019, ApJS, 240, 17