HIGH-PERFORMANCE EXOPLANET TRANSIT FITTING

SYDNEY JENKINS

Github link: https://github.com/sydneyjenkins/Julia_Project

Abstract. Discovering exoplanets with the transit technique requires continuous monitoring of O(100,000) stars. It is therefore critical to have an efficient, automated pipeline for identifying the dips in brightness characteristic of stars with transiting exoplanets. We perform a preliminary investigation of whether a Julia fitting pipeline could fulfill this role by comparing the speed of Python and Julia implementations and comparing the accuracy of two different fitting models, called the Box model and the Mandel-Agol model. These models are fit using our own Markov chain Monte Carlo sampler. We find that the Julia implementation outperforms the Python implementation, and that parallelization provides an additional increase in speed. Additionally, more complex models require a trade-off between accuracy and efficiency.

1. Introduction. Since the first discovery of the first exoplanet in 1992 [5], over 5,000 exoplanets have been discovered. Nearly 4,000 of these worlds were found using transit techniques [e.g., 1]. As a planet orbits around its host star, it blocks a portion of its light. This signal is detectable in the system's brightness as a function of time, commonly known as the light curve. The change in brightness is closely related to the relative sizes of the planet r_p and star $r_\star$, and is proportional to $(\frac{r_p}{r_\star})^2$. Figure 1 provides a demonstration of how the exoplanet's motion affects the observed light curve. This light curve encodes important information about the system, such as the planet's orbital period, radius, and density (when combined with stellar radial velocity data).

In order to find exoplanets using the transit method, many stars must be monitored. For instance, past missions such as Kepler observed O(100,000) stars, while future missions such as the Nancy Grace Roman Space Telescope plan to observe O(100,000,000) stars. Being able to efficiently fit a large number of light curves will therefore be critical for quickly identifying exoplanets hidden in upcoming telescope data.

We conduct a preliminary investigation into whether Julia would provide a suitable platform for creating a transit fitting pipeline. To do this, we create a Julia pipeline to test the accuracy and efficiency of different fitting methods. Specifically, we test two different transit models: the Box model and the Mandel-Agol model. We also run our code using Python and Julia implementations, and serial and parallel implementations. We then compare the speed and accuracy of our different implementations. To fit our models, we write our own Markov chain Monte Carlo sampler.

The paper is organized as follows. We describe our data in section 2, our fitting models in section 3, and our MCMC sampler in section 4. Our experimental set-up is outlined in section 5, results are discussed in section 6, code availability is described in section 7, and future work is proposed in section 8. We then conclude in section 9.

2. Data. We use data from Kepler, a space-based telescope that operated between 2009 and 2018. Kepler had a fixed field of view, meaning that it continuously looked at the same patch of sky. It monitored ∼150,000 stars in this region. This long-term monitoring of a large number of stars makes it the ideal mission for studying transiting exoplanets.

We place several restrictions on the light curves used in this analysis. We focus on bright targets (with magnitudes < 14) and easily-observed transit depths ($0.01 - 0.05$) that correspond to large planets. This period depth is defined as the fractional

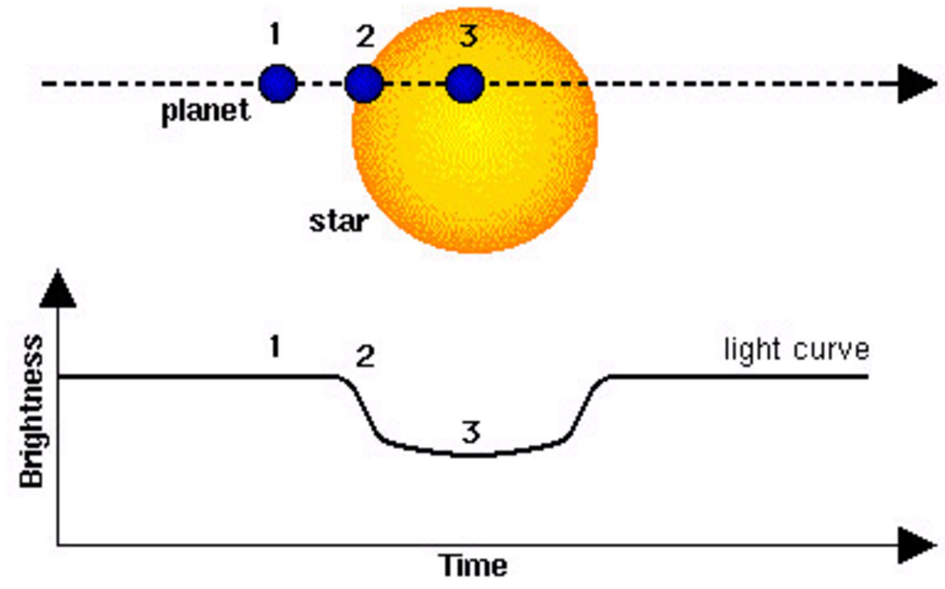

Fig. 1. *Diagram depicting an exoplanet transit. If the star, planet, and obsrvor are properly aligned, we observe a dip in the star's brightness as the planet passes in front of it. Position 1, 2, and 3 correspond to the out-of-transit brightness, ingress brightness, and in-transit brightness, respectively. Diagram from the European Space Agency (ESA).*

change in flux during the transit. We also require that the planet's orbital period is greater than 5 days, as this avoids light curves with unusual shapes. In future work, these constrains could be loosened in order to fit a wider variety of light curves. This is further discussed in section 8. 115 Kepler stars with known planets satisfy our magnitude, depth, and period constraints. Several example light curves are shown in Figure 2. These light curves all correspond to different planet sizes, periods, and orbital configerations.

Each light curve in our sample is normalized using a package called keplersplinev2 [4]. This package also smooths the light curve and removes long-term variability caused by stellar effects and instrumental artifacts.

In this analysis, we choose to divide each light curve into individual transit events, rather than fitting the entire light curve or fitting the phase-folded light curve. This was done to minimize the fitting time. While this reduction in our data size may reduce the quality of our fit, it still provides a sufficient basis for this preliminary investigation, as we can still compare the calculated running times and accuracy measurements of the different fitting methods.

3. Fitting Algorithms.

To investigate the trade-off between speed and accuracy, we compare two exoplanet transit models: the Box model and the Mandel-Agol model. This allows us to explore the effects of model complexity on both the running time and accuracy of our fitting code. These models are described in detail below.

3.1. Box Model.

The Box model is the simplest exoplanet transit fitting model. It models the transit event as a box-shaped function that takes either a high (out-of-transit) or low (in-transit) value. It requires four parameters: the period P, transit duration ΔT, transit depth d, and a reference time t_0. Because we are only fitting an individual transit, our input data does not contain information about the planet's orbital period. We therefore use the Kepler catalogue period for this initial investigation. Additionally, while the reference time normally corresponds to some mid-transit time of an earlier transit, in this case we define it as the starting time of the observed transit. Our Box function is therefore a piece-wise function of the following form:

$$F(t, t_0, \Delta t, d) = \left\{ \begin{array}{ll} 1 & \text{if } t < t_0 \text{ or } t > t_0 + \Delta t \\ 1 - d & \text{if } t > t_0 \text{ and } t < t_0 + \Delta t \end{array} \right\} \tag{3.1}$$

F corresponds to the observed stellar flux. This equation assumes that the light curve has been normalized to a value of 1.

3.2. Mandel-Agol Model. The Mandel-Agol (MA) model [3] is a more complex model for fitting exoplanet transits. Specifically, this model accounts for an optical effect called limb darkening. This occurs because, when looking at a star closer to its center, one can see further into its atmosphere. Because these deeper layers are at a higher temperature, the star appears brighter. Therefore, stars will seem brighter at their center and dimmer at their edges, or "limbs." This effect is visible in Figure 1. Because of limb darkening, when the planet passes in front of the edges of the star, it will block less flux than it does when it passes in front of the center of the star. This causes the light curve to become more rounded.

We model this effect using a quadratic limb darkening model. This impacts the stellar intensity I, which is defined in terms of the limb darkening coefficients (γ_1 and γ_2) and μ (where $\mu = \sqrt{1 - r^2}$). μ accounts for the changing angle between the planet and star. Specifically, I is defined as:

$$I(\gamma_1, \gamma_2, \mu) = 1 - \gamma_1(1 - \mu) - \gamma_2(1 - \mu)^2 \tag{3.2}$$

To establish a baseline, we also model the flux ratio for a uniform source $F^e(p, z) = 1 - \lambda^e(p, z)$. This value depends on the size ratio between the planet and star p and the normalized separation of the object centers z. Specifically, p is defined in terms of the planetary radius r_p and stellar radius $r_\star$ to be $r_p/r_\star$. z is defined in terms of the center-to-center distance between the star and planet d and the stellar radius to be $d/r_\star$. λ^e is defined below:

$$\lambda^e(p, z) = \left\{ \begin{array}{lr} 0 & \text{if } 1 + p < z \\ \frac{1}{\pi}\left[p^2\kappa_0 + \kappa_1 - \sqrt{\frac{4z^2 - (1+z^2-p^2)^2}{4}}\right] & \text{if } |1 - p| < z \leq 1 + p \\ p^2 & \text{if } z \leq 1 - p \\ 1 & \text{if } z \leq p - 1 \end{array} \right\} \tag{3.3}$$

The parameters κ_0 and κ_1 are defined as $\cos^{-1}[(p^2 + z^2 - 1)/2pz]$ and $\cos^{-1}[(1 - p^2 + z^2)/2pz]$, respectively.

We then combine these equations to define the final light curve:

$$F(r, p, z) = \left[\int_0^1 2rI(r)\,dr\right]^{-1} \int_0^1 I(r)\frac{d[F^e(p/r, z/r)r^2]}{dr}\,dr \tag{3.4}$$

F and r correspond to the observed stellar flux and the normalized radial coordinate on the disk of the star, respectively. We note that this gives the flux as a function of

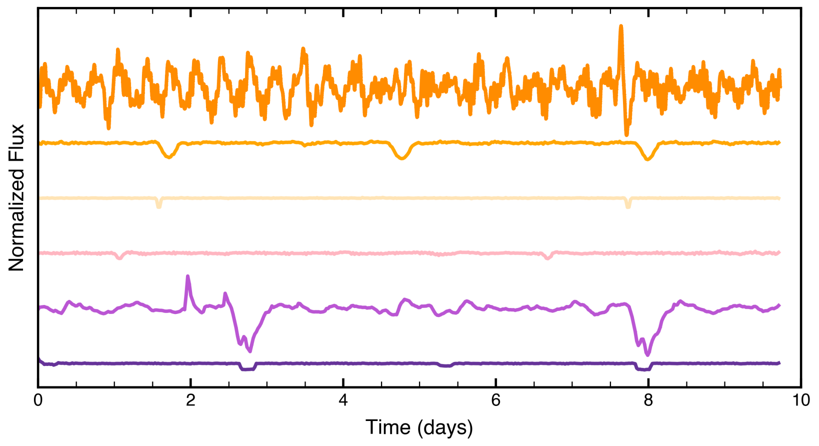

FIG. 2. *Example exoplanet transits from Kepler data, shown as the normalized stellar flux as a function of time. The passage of the planet in front of the star is detectable as a periodic dip in the host star's flux. Different planet sizes and orbital configurations create a diverse variety of light curve shapes.*

position instead of time. To convert this to a function of time, we use the following equation for z:

$$z(t) = R_a^{-1}\left[(\sin\omega t)^2 + (\cos i \, \cos\omega t)^2\right]^{1/2} \tag{3.5}$$

Here, R_a is the stellar radius over the planet's semi-major axis, i is the inclination angle, and ω is $\frac{2\pi}{P}$, where P is the planet's orbital period.

Because we are only fitting a single transit, rather than a full light curve or phase-folded transit, we expect there to be degeneracies between these many parameters. We therefore simplify the fitting process by assuming that the star, planet, and observor are perfectly aligned ($i = 0$). We also use the value of the period provided by the Kepler catalog. We therefore restrict ourselves to fitting the following parameters: the time of transit t_0, γ_1, γ_2, p, and R_a.

4. MCMC Fitting. We write our own Markov chain Monte Carlo (MCMC) sampler to find the best-fit parameters. We create multiple walkers to explore the parameter space, all starting from slightly different starting guesses. Each walker has their own corresponding chain, where they can update their parameter values.

We update each set of parameters by adding a small perturbation. We then calculate the acceptance probability of this new state using the log likelihood function and log prior function. The log likelihood function computes the logarithm of the likelihood of observing the data given the current set of model parameters. Taking the logarithm simply simplifies the calculations performed. However, maximizing the log likelihood function is equivalent to maximizing the likelihood function. We can therefore use it to determine whether one set of parameters is a better fit than another set of values.

The priors restrict the parameters to certain allowed ranges. For both the Box and MA model, we allow t_0 to range within 0.2 days of a pre-calculated value. This is meant to simplify the fitting process. For the Box model, we then allow Δt to range between 0.01 and 0.5 days, and d to range between 0.01 and 0.1. For the MA model, we allow γ_1 and γ_2 to range between -1 and 1, p to range between 0 and 0.5, and R_a to range between 0.001 and 1.

Once we have calculated the sum of the log likelihood and log prior values for a given iteration i, denoted by l_i, we can determine the acceptance probability. This is calculated by taking the difference with the previous sum (i.e. $l_i - l_{i-1}$). We then generate a random number n to determine whether the new parameter values are accepted ($l_i - l_{i-1} > \log(n)$) or rejected ($l_i - l_{i-1} < \log(n)$). In this way, the sampler will converge toward increasingly better fitting parameter values.

After a set number of iterations, some number of parameter values are discarded from each chain in a process known as burn-in. This allows us to take the mean and standard deviation of the chains after they have reached some equilibrium value, thus providing us with the best-fit parameter values.

Two example fits are shown in Figure 3. This figure showcases both the Box and MA models, both fit to the transit data of star KIC 6965293 (shown in black) using our MCMC code. The best-fit parameter values are also provided.

5. Experimental Set-Up.

5.1. Comparisons. We perform several comparisons in this investigation. These are summarized below:

- **Python vs. Julia** We implement our algorithms in both Python and Julia. This is done to compare the speed of calculations in each language. We expect the Julia implementations to be faster, given that Julia was designed for efficiency.
- **Box vs. MA Model** We fit our light curves with both the Box and MA model. The Box model requires us to fit only three parameters, while the MA model requires us to fit five parameters. The MA model is also significantly more computationally expensive, as it requires many more computations, including two integrals. We therefore expect it to take a longer time to perform each iteration. However, we also expect it to provide a better fit to the data, as it accounts for limb darkening effects.
- **Serial vs. Parallel** We fit each model using both a serial and parallel implementation, allowing us to investigate the benefits of parallelization. Because we are running this pipeline locally, we only use four threads (corresponding to the number of cores on our machine). However, we still expect to observe a speedup for the parallel implementation.

5.2. Measurements. In this experiment, we are interested in the speed and accuracy of our implementations. We estimate these parameters using the methods described below:

- **Speed** To perform a fair comparison of algorithm speeds, we time each implementation using the same number of walkers and iterations per walker. Because Python is slow, we limit ourselves to 128 walkers and 100 iterations per walker for the comparison between Python and Julia. We then time the MCMC code, using the time modules in Python and Julia. We decide not to use the Julia BenchmarkTools package in order to provide a fairer comparison with the Python time measurements. We acknowledge that, given the relatively small (12800) number of total parameter updates performed per implementation in this experiment, the estimated times include large contributions from the MCMC overhead calculations, such as determining the initial parameter

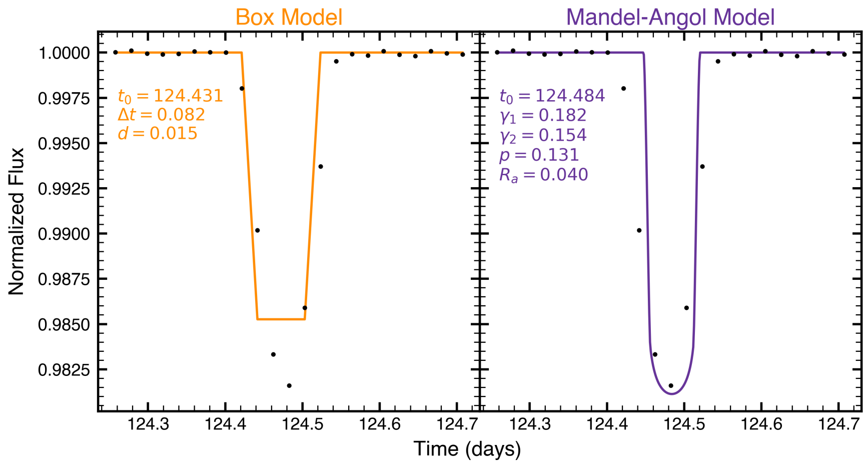

FIG. 3. *Example model fits to star KIC 6965293. Data is shown in black. The best-fit parameter values and fit are shown for the Box model (left) and for the Mandel-Agol (MA) model (right). The MA model provides a more accurate fit to the transit event, but is more computationally expensive.*

guesses and initializing the chains. To provide a better comparison of the time requirements for the Julia implementations, we also measure the time taken to compute 10,000 iterations per walker.

- **Accuracy** To estimate the accuracy of a given fit, we calculate the Root Mean Square Error (RMSE) of the model's predictions. The RMSE is calculated using the following equation:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{5.1}$$

where n is the size of the data set, y_i is the ith data point, and $\hat{y}_i$ is the ith model prediction. Because the average RMSE is expected to be the same for a given model, regardless of whether the model is fit in Julia or Python, or fit using a serial or parallel implementation, we simply calculate a mean RMSE using our parallel Julia implementation. We fit 20 unique transits and use the mean RMSE values to compare the accuracy of the Box and MA models.

6. Results and Discussion. We measure the time per iteration for each of eight implementations, and the RMSE for both the Box and Mandel-Agol models. Our results are summarized in Table 1.

We find that the Julia implementation is faster, as expected. Specifically, we find that the speed of the Julia implementations is ∼3 orders of magnitude faster than the corresponding Python versions. This indicates that Julia would be a suitable language for large-scale exoplanet transit fitting, as it would be able to efficiently fit the large number of light curves that will be generated by upcoming telescope missions.

We also find that parallelizing the Julia code provides an additional speed-up for the 10,000-iteration comparison. This is consistent with what we expected, as having multiple threads iterate over the chains reduces the overall time required to perform all the computations. We find that the serial versions take 1.4 and 2.7

times longer than the parallel versions when fitting the Box model and MA models, respectively. However, the increased speed is less than one might naively expect when using four threads. That is, even though we use four threads, our code does not run four times faster. This makes sense, as there are computational overheads to both our fitting algorithm (such as initializing the chains) and the process of multi-threading (such as creating multiple threads). This also explains why, in the 100-iteration comparison, the parallel version sometimes performs worse than the serial version—with a lower number of iterations per walker, the advantage gained by parallelizing the code becomes dominated by the overhead costs.

Finally, we find that fitting the Box model is ∼2 orders of magnitude faster than fitting the MA model. This is also consistent with our expectations, as the Box model has fewer parameters to fit and requires fewer computations. However, there is a trade-off between speed and accuracy, as the MA model has a lower RMSE (0.002±0.001) compared to the Box model (0.003±0.002). This performance reflects the MA model's inclusion of limb darkening effects. These results indicate that, when choosing which model to fit to a light curve, it is important to consider the relative importance of speed compared to accuracy.

TABLE 1

Results for each method used, including for different languages (Julia or Python), models (Box model or Mandel-Agol model), and computing methods (serial or parallel). We report the time taken to fit a single transit based on the number of iterations per chain (either 100 or 10,000), and the root-mean-square error (RMSE).

Model	Language	P/S	100-it Time (s)	10,000-it Time (s)	RMSE
Box	Julia	Serial	0.03	2.1	0.003±0.002
		Parallel	0.09	1.5	
	Python	Serial	52.1		
		Parallel	49.9		
Mandel-Agol	Julia	Serial	3.9	365.7	0.002±0.001
		Parallel	1.3	131.7	
	Python	Serial	2638.5		
		Parallel	2926.7		

7. Code Availability. The relevant code is included in the GitHub link provided at the beginning of this study. The GitHub page includes six files, including the Julia implementation, the Python implementation, and the data. The Julia implementation includes fits to both the Box and MA models, and includes both serial and parallel options. The Python implementation includes many of the same functions, but in Python. Both implementations are done in an interactive format (Pluto and Jupyter Notebook for the Julia and Python implementations, respectively).

Finally, the data files include 115 transits from the stars that satisfy all the criteria described in section 2. There are also additional files containing the t_0 values, transit depths, and orbital periods for each light curve. The transit depths and orbital periods are collected from the Kepler catalog, while the t_0 values are estimated during the separation of the 115 light curves into individual transit events. These values are used to help improve our fits, but would ideally not be necessary in the final implementation of any fitting pipeline.

8. Future Work. Because this is a preliminary investigation into the benefits of using a Julia pipeline, there are many areas for future work to expand upon these results. Specifically, we limit ourselves to a subset of the available light curves, and only investigate two transit models. Additionally, we use a relatively simple MCMC sampler, and restrict ourselves to just four threads. We describe ways to address these limitations in further detail below.

- **Increased Diversity of Light Curves** In this experiment, we select light curves that satisfy the following constraints: $P > 5$ days, magnitude < 14, and transit depth between 0.01 and 0.05. This restricted our light curves to those with relatively straight-forward, easily detectable signals. We additionally only fit individual transit events. However, to identify transits in real survey data, we would want to expand our pipeline to be able to fit full light curves with more complex shapes. Therefore, future work could evaluate the accuracy of the Box and MA models on a representative set of light curves. Additionally, rather than fitting a single transit, they could fit the entire light curve. This would allow future implementations to fit the planet's orbital period, in addition to the other parameters.
- **Additional Transit Models** We could also compare the speed and accuracy of other exoplanet transit models. Additional parameters that could be fit include the inclination (which is included in the MA model, but set to a value of zero in this study), eccentricity, and period. Additionally, there are a wide range of limb darkening model that can be adopted, including uniform, linear, logarithmic, and exponential models. Fitting more parameters would allow us to account for a larger range of behavior in the light curves, thus providing us with better fits and making our code applicable to a more diverse set of light curves. Packages that allow for these complexities already exist in Python [2] and are widely used. A similar implementation in Julia would provide similar accuracy paired with improved efficiency.
- **Optimized Sampler** In addition, we could further optimize our sampler. For instance, we are currently using a set perturbation distribution to update our parameter values. However, future work could have an update value that decreases in size as the sampler converges on the best-fit values. This would provide a more precise estimate of the best-fit parameters. Similarly, we could break the fitting into multiple stages, using the average chain values from the previous iteration as the starting parameters for the next iteration. The perturbations could then be modified for each of these iterations. This could also provide improved convergence.
- **Increased Parallelization** In this experiment, we restrict the number of threads to the number of cores on our local machine. However, future work could run our code on a computing cluster and dramatically increase the amount of parallelization. This would likely greatly increase the speed of our code. We would expect that, as we increase the number of threads, the benefit of using parallelization would increasingly dominate the associated overhead cost of setting up the parallelization. However, once the number of threads is equal to the number of walkers, there would no longer be any more advantages to increasing the number of threads, as the code is designed to designate at least one walker per thread.

9. Conclusions. We perform a preliminary comparison of the speed and efficiency of several exoplanet transit fitting methods, with a specific focus on the effects of the coding language, the transit model used, and parallelization. As expected, we find that using Julia and multithreading both lead to increases in speed. Additionally, we find that the MA model provides a more accurate fit to light curve data. However, it also takes longer to fit, indicating that its computational costs should be weighed against its improved accuracy when deciding which model to use. All relevant code is provided on GitHub.

Our findings indicate that a Julia transit fitting pipeline could provide improved accuracy relative to more traditional pipelines in Python. As future telescope missions look at an ever increasing number of stars, this efficiency will become a pivotal advantage, making Julia the ideal language for future transit fitting.

References.

[1] D. Charbonneau, T. M. Brown, D. W. Latham, and M. Mayor. Detection of Planetary Transits Across a Sun-like Star. , 529(1):L45–L48, Jan. 2000.

[2] L. Kreidberg. ttbatman/tt: BAsic transit model cAlculatioN in python. *Publications of the Astronomical Society of the Pacific*, 127(957):1161–1165, nov 2015.

[3] K. Mandel and E. Agol. Analytic Light Curves for Planetary Transit Searches. , 580(2):L171–L175, Dec. 2002.

[4] A. Vanderburg and J. A. Johnson. A Technique for Extracting Highly Precise Photometry for the Two-Wheeled Kepler Mission. , 126(944):948, Oct. 2014.

[5] A. Wolszczan and D. A. Frail. A planetary system around the millisecond pulsar PSR1257 + 12. , 355(6356):145–147, Jan. 1992.