Publications of the Astronomical Society of the Pacific, 130:044504 (13pp), 2018 April https://doi.org/10.1088/1538-3873/aaaaa8


# RadVel: The Radial Velocity Modeling Toolkit

Benjamin J. Fulton[1,2,3], Erik A. Petigura[1,4], Sarah Blunt[1], and Evan Sinukoff[1,2,5]
[1] California Institute of Technology, Pasadena, California, USA
[2] Institute for Astronomy, University of Hawai'i at Maānoa, Honolulu, HI 96822, USA
*Received 2017 December 22; accepted 2018 January 16; published 2018 March 12*

## Abstract

`RadVel` is an open-source Python package for modeling Keplerian orbits in radial velocity (RV) timeseries. `RadVel` provides a convenient framework to fit RVs using maximum a posteriori optimization and to compute robust confidence intervals by sampling the posterior probability density via Markov Chain Monte Carlo (MCMC). `RadVel` allows users to float or fix parameters, impose priors, and perform Bayesian model comparison. We have implemented real-time MCMC convergence tests to ensure adequate sampling of the posterior. `RadVel` can output a number of publication-quality plots and tables. Users may interface with `RadVel` through a convenient command-line interface or directly from Python. The code is object-oriented and thus naturally extensible. We encourage contributions from the community. Documentation is available at http://radvel.readthedocs.io.

*Key words:* (stars:) planetary systems – planets and satellites: fundamental parameters – planets and satellites: general

*Online material:* color figures

## 1. Introduction

The radial velocity (RV) technique was among the first techniques to permit the discovery and characterization of extrasolar planets (e.g., Campbell et al. 1988; Latham et al. 1989; Mayor & Queloz 1995; Marcy & Butler 1996). Prior to NASA's *Kepler* Space Telescope (Borucki et al. 2010), the RV technique accounted for the vast majority of exoplanet detections (Akeson et al. 2013).

In the post-*Kepler* era, the RV field has shifted somewhat from discovery to followup and characterization of planets discovered by transit surveys. In the case of planets discovered via either technique, the need to model RV orbits to extract planet masses (or minimum masses, $M \sin i$) is a critical tool necessary to understand the compositions and typical masses of exoplanets.

Several software packages have been written to address the need of the exoplanet community to fit RV timeseries.[6] In designing `RadVel`, we emphasized ease of use, flexibility, and extensibility. `RadVel` can be installed and a simple planetary system can modeled from the command-line in seconds. `RadVel` also provides an extensive and well-documented application programming interface (API) to perform complex fitting tasks. We employ modern Markov Chain Monte Carlo (MCMC) sampling techniques and robust convergence criteria to ensure accurately estimated orbital parameters and their associated uncertainties.

The goal of this paper is to document the core features of `RadVel` version 1.0 (Fulton & Petigura 2017). Due to the evolving nature of `RadVel`, the most up-to-date documentation can be found at http://radvel.readthedocs.io (RTD page hereafter). This paper complements that documentation and is structured as follows. We describe the parameters involved to describe an RV orbit in Section 2, in Section 3 we discuss Bayesian inference as implemented in `RadVel`. The design of the code is described in Section 4 and the model fitting procedure is described in Section 5. We explain how to install `RadVel` and walk through two example fits to demonstrate how the code is run in Section 6. We describe the mechanism for support and contributions in Section 7 and close with some concluding remarks in Section 8.

## 2. The Radial Velocity Orbit

RV orbits are fundamentally described with five orbital elements: orbital period ($P$), a parameter that describes the orbital phase at a given time (we use the time of inferior conjunction, $T_c$); orbital eccentricity ($e$); the argument of periastron of the star's orbit ($\omega$); and the velocity semi-amplitude ($K$). We also include terms for the mean center-of-mass velocity ($\gamma$), plus linear ($\dot{\gamma}$), and quadratic ($\ddot{\gamma}$) acceleration terms in the RV model. Since RV measurement uncertainties generally do not take into account contributions from

[3] Texaco Fellow.
[4] Hubble Fellow.
[5] Natural Sciences and Engineering Research Council of Canada Graduate Student Fellow.
[6] A non-exhaustive list includes `RVLIN` (Wright & Howard 2009), `Systemic` (Meschiari et al. 2009; Meschiari & Laughlin 2010), `EXOFAST` (Eastman et al. 2013), `rvfit` (Iglesias-Marzoa et al. 2015), and `ExoSOFT` (Mede & Brandt 2017).

astrophysical and instrumental sources of noise we also fit for a "jitter" term ($\sigma$), which is added in quadrature with the measurement uncertainties. These parameters are listed and described in Table 1 for reference. Figure 1 depicts an example eccentric Keplerian orbit with several of these parameters annotated. We define a cartesian coordinate system described by the $\hat{x}$, $\hat{y}$, and $\hat{z}$ unit vectors such that $\hat{z}$ points away from the observer and $\hat{y}$ is normal to the plane of the planet's orbit. In this coordinate system, the x-y plane defines the sky plane.

### 2.1. Keplerian Solver

Synthesizing radial velocities involves solving the following system of equations:

$$M = E - e \sin E, \tag{1}$$

$$\nu = 2 \tan^{-1}\left(\sqrt{\frac{1+e}{1-e}} \tan \frac{E}{2}\right), \tag{2}$$

$$\dot{z} = v_r = K[\cos(\nu + \omega) + e \cos(\omega)], \tag{3}$$

where $M$ is the mean anomaly, $E$ is the eccentric anomaly, and $e$ is the orbital eccentricity, $\nu$ is commonly referred to as the true anomaly, $K$ is the velocity semi-amplitude, and $v_r$ is the star's reflex RV caused by a single orbiting planet. Equation (1) is known as *Kepler*'s equation, and we solve it using the iterative method proposed by Danby (1988) and reproduced in Murray & Dermott (1999).

Note that we equate $\dot{z}$ to $v_r$ in our description of the RV orbit. In our coordinate system, the $\hat{z}$ unit vector points away from the observer. Historically, this has been the observational convention so that positive RV can be interpreted as a redshift of the star. However, some derivations of this equation in the literature (e.g., Murray & Correia (2010) and Deck et al. (2014) define their coordinate system such that $\hat{z}$ points toward the observer and derive an equation for RV that is very similar to our Equation (3)). The key difference when defining a coordinate system with $\hat{z}$ pointing toward the observer is that the sign in Equation (3) must be flipped ($v_r = -\dot{z}$) or, equivalently, $\omega$ must refer to the argument of periapsis of the planet's orbit, which is shifted by $\pi$ relative to the argument of periapsis of the star's orbit.

In the case of multiple planets, the Keplerian orbits are summed together. We also add acceleration terms to account for additional perturbers in the system with orbital periods much longer than the observational baseline. The total RV motion of the star due to all companions ($\mathcal{V}_r$) is

$$\mathcal{V}_r = \sum_{k}^{N_{\rm pl}} v_{r,k} + \gamma + \dot{\gamma}(t - t_0) + \ddot{\gamma}(t - t_0)^2, \tag{4}$$

where $N_{\rm pl}$ is the total number of planets in the system and $t_0$ is an arbitrary abscissa epoch defined by the user.

**Table 1**
Keplerian Orbital Elements

| Parameter Description | Symbol |
|---|---|
| **Keplerian Orbital Parameters** | |
| Orbital period | $P$ |
| Time of inferior conjunction (or transit)[a] | $T_c$ |
| Time of periastron[a,b] | $T_p$ |
| Eccentricity | $e$ |
| Argument of periapsis of the star's orbit[b] | $\omega$ |
| Velocity semi-amplitude | K |
| **Mean and Acceleration Terms** | |
| Mean center-of-mass velocity[c] | $\gamma_i$ |
| Linear acceleration term | $\dot{\gamma}$ |
| Second-order acceleration term | $\ddot{\gamma}$ |
| **Noise Parameters** | |
| Radial velocity "jitter" (white noise)[c] | $\sigma_i$ |

**Notes.**
[a] Either $T_c$ or $T_p$ can be used to describe the phase of the orbit. Both are not needed simultaneously.
[b] Undefined for circular orbits.
[c] Usually specific to each instrument ($i$).

### 2.2. Parameterization

To speed fitting convergence and avoid biasing parameters that must physically be finite and positive (e.g., Lucy & Sweeney 1971), analytical transformations of the orbital elements are often used to describe the orbit. In `RadVel`, we have implemented several of these transformations into six different "basis" sets. One of the highlight features of `RadVel` is its ability to easily switch between these different bases. This allows the user to explore any biases that might arise based on the choice of parameterization, and/or impose priors on parameters or combinations of parameters that are not typically used to describe an RV orbit (e.g., a prior on $e \cos \omega$ from a secondary eclipse detection).

We have implemented the basis sets listed in Table 2. Users can easily add additional basis sets by modifying the `radvel.basis.Basis` object. A string representation of the new basis should be added to the `radvel.basis.Basis.BASIS_NAMES` attribute. The `radvel.basis.Basis.to_synth` and `radvel.basis.Basis.from_synth` methods should also be updated to properly transform the new basis to and from the existing "synth" basis.

Because priors are assumed to be uniform in the fitting basis, this imposes implicit priors on the Keplerian orbital elements. For example, choosing the "$\ln P$, $T_c$, $e$, $\omega$, $\ln K$" basis would impose a prior that favors small $P$ and $K$ values as there is much more phase space for the MCMC chains to explore near $P = K = 0$. See Eastman et al. (2013) for a detailed description of the implicit priors imposed on $e$ and $\omega$ based

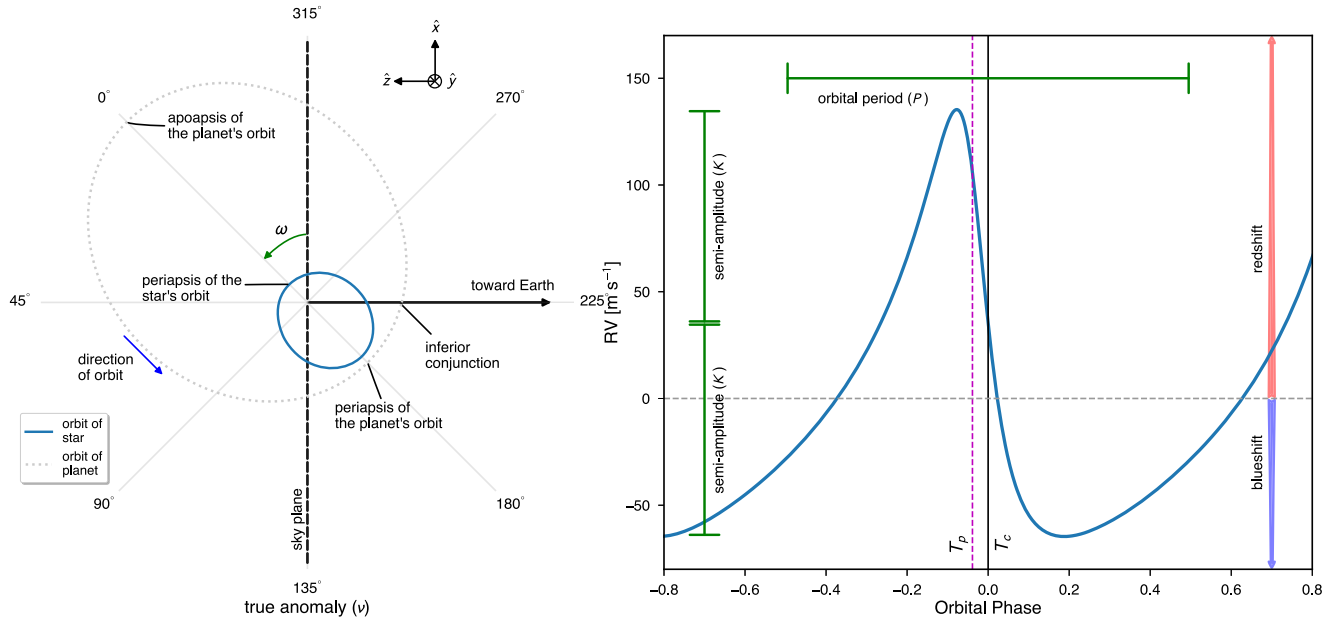


**Figure 1.** Diagram of a Keplerian orbit. Left: top-down view of an eccentric Keplerian orbit with relevant parameters labeled. The planet's orbit is plotted as a light gray dotted line and the star's orbit is plotted in blue (scale exaggerated for clarity). Right: model radial velocity curve for the same orbit with the relevant parameters labeled. We define a cartesian coordinate system described by the $\hat{x}$, $\hat{y}$, and $\hat{z}$ unit vectors such that $\hat{z}$ points away from the observer and $\hat{y}$ is normal to the plane of the planet's orbit. In this coordinate system, the x-y plane defines the sky plane. The plane of the orbit lies in the plane of the page.
(A color version of this figure is available in the online journal.)

**Table 2**
Parameterizations of the Orbit

| Parameters Describing Orbit | Notes |
|---|---|
| $P$, $T_p$, $e$, $\omega$, $K$ | "Synth" basis used to synthesize RVs |
| $P$, $T_c$, $\sqrt{e}\cos\omega$, $\sqrt{e}\sin\omega$, $K$ | Standard basis for fitting and posterior sampling |
| $P$, $T_c$, $\sqrt{e}\cos\omega$, $\sqrt{e}\sin\omega$, $\ln K$ | Forces $K > 0$ |
| $P$, $T_c$, $e\cos\omega$, $e\sin\omega$, $K$ | Imposes a linear prior on $e$ |
| $P$, $T_c$, $e$, $\omega$, $K$ | Slower MCMC convergence |
| $\ln P$, $T_c$, $e$, $\omega$, $\ln K$ | Useful when $P$ is long compared with the observational baseline |

on the choice of fitting in $e\cos\omega$ and $e\sin\omega$, $\sqrt{e}\cos\omega$ and $\sqrt{e}\sin\omega$, or $e$ and $\omega$ directly. In the case that the data does not constrain, or weakly constrains some or all orbital parameters, the choice of the fitting basis becomes important due to these implicit priors. We usually prefer to perform fitting and posterior sampling using the $P$, $T_c$, $\sqrt{e}\cos\omega$, $\sqrt{e}\sin\omega$, $K$ basis because this imposes flat priors on all of the orbital elements, avoids biasing $K > 0$, and helps to speed MCMC convergence. However, the $P$, $T_c$, $\sqrt{e}\cos\omega$, $\sqrt{e}\sin\omega$, $\ln K$, and $\ln P$, $T_c$, $e$, $\omega$, $\ln K$ bases can be useful in some scenarios, especially when $K$ is large and $P$ is long compared with the observational baseline. There is no default basis. The choice of fitting basis must be made explicitly by the user. It is generally good practice to perform the fit in several different basis sets to check for consistency.

Inspection of Equation (3) shows that $v_r$ for $K = +q$ and $\omega = \lambda$ is identical to $v_r$ when $K = -q$ and $\omega = \lambda + \pi$. For low signal-to-noise detections where the MCMC walkers (see Section 5.2) can jump between the $K = +q$ and $K = -q$ solutions, this degeneracy can lead to bimodal posterior distributions for $K$ reflected about $K = 0$. The posterior distributions for $T_c$ and/or $\omega$ will also be bimodal. In these cases, we advise the user to proceed with caution when interpreting the posterior distributions and to explore a variety of basis sets and priors (see Section 4.5) to determine their impact on the resulting posteriors.

## 3. Bayesian Inference

We model RV data following the standard practices of Bayesian inference. The goal is to infer the posterior probability density $p$ given a data set ($\mathcal{D}$) and priors as in Bayes' Theorem:

$$p(\theta|\mathcal{D}) \propto \mathcal{L}(\theta|\mathcal{D})p(\theta). \quad (5)$$

The Keplerian orbital parameters are contained in the $\theta$ vector. $\mathcal{L}(\theta|\mathcal{D})$ is the likelihood that the data is drawn from the model described by the parameter set $\theta$. Assuming Gaussian distributed noise, the likelihood is

$$\ln \mathcal{L}_i(\theta|\mathcal{D}_i) = -\frac{1}{2}\sum_j \frac{(\mathcal{V}_{r,j}(t,\theta) - d_j)^2}{e_j^2 + \sigma_i^2} - \ln\sqrt{2\pi(e_j^2 + \sigma_i^2)}, \quad (6)$$

where $\mathcal{V}_{r,j}$ is the Keplerian model (Equation (4)) predicted at the time ($t$) of each RV measurement ($d_j$), $e_j$ is the measurement uncertainty associated with each $d_j$, and $\sigma_i$ is a Gaussian noise term to account for any astrophysical or instrumental noise not included in the measurement uncertainties. The $\sigma_i$ terms are unique to each instrument ($i$). For data sets containing velocities from multiple instruments, the total likelihood is the sum of the natural log of the likelihoods for each instrument:

$$\ln \mathcal{L}(\theta|\mathcal{D}) = \sum_i \ln \mathcal{L}_i(\theta|\mathcal{D}_i) \quad (7)$$

We sample the posterior probability density surface using MCMC (see Section 5.2). The natural log of the priors are applied as additional additive terms such that

$$\ln \mathcal{L}(\theta|\mathcal{D})p(\theta) = \ln \mathcal{L}(\theta|\mathcal{D}) + \sum_k \ln \mathcal{P}_k(\theta), \quad (8)$$

for each prior ($\mathcal{P}_k$). If no priors are explicitly defined by the user, then all priors are assumed to be uniform and $\sum_k \ln \mathcal{P}_k(\theta) = 0$.

## 4. Code Design

The fundamental quantity for model fitting and parameter estimation is the posterior probability density $p$, which is represented in `RadVel` as a Python object. Users create $p$ by specifying a likelihood $\mathcal{L}$, priors $\mathcal{P}$, model $\mathcal{V}_r$, and data $\mathcal{D}$, which are also implemented as objects. Here, we describe each of these objects, which are the building blocks of the `RadVel` API. Figure 2 summarizes these objects and their hierarchy.

### 4.1. Parameters

The posterior probability density $p$ is a surface in $\mathcal{R}^N$, where $N$ is number of free parameters. We specify coordinates in this parameter space using a `radvel.Parameters` object. `radvel.Parameters` is a container object that inherits from Python's ordered dictionary object, `collections.OrderedDict`. We modeled this dictionary representation after the `lmfit`[7] (Newville et al. 2014) API, which allows users to conveniently interface with variables via string keys (as opposed to integer indexes). Auxiliary attributes, such as the number of planets and the fitting basis, are also stored in the `radvel.Parameters` object outside of the dictionary representation.

Each element of the `radvel.Parameters` dictionary is represented as a `radvel.Parameter` object that contains the parameter value and a boolean attribute which specifies if the parameter is fixed or allowed to float.[8]

### 4.2. Model Object

The `radvel.RVModel` class is a callable object that computes the radial velocity curve that corresponds to the parameters stored in the `radvel.Parameters` object. Calculating the model RV curve requires solving *Kepler*'s equation (see Section 2.1), and is computationally intensive. This solver is implemented in C to maximize performance. We also provide a Python implementation so that users may run `RadVel` without compiling C code (albeit at slower speeds).

### 4.3. Likelihood Object

The primary function of the `radvel.Likelihood` object is to establish the relationship between a model and the data. It is a generic class which is meant to be inherited by objects designed for specific applications such as the `radvel.RVLikelihood` object. Most fitting packages (e.g., `emcee`, Foreman-Mackey et al. 2013) require functions that take vectors of floating-point values as inputs, and outputs a single goodness-of-fit metric. These conversions between the string-indexed `radvel.Parameters` object and ordered arrays of floats containing only the parameters that are allowed to vary are handled within the `radvel.Likelihood` object.

The `radvel.RVLikelihood` object is a container for a single radial velocity data set (usually from a single instrument) and a `radvel.RVModel` object. The `radvel.Likelihood.logprob` method returns the natural log of the likelihood of the model evaluated at the parameter values contained in the `params` attribute given the data contained in the `x`, `y`, and `yerr` attributes. The `extra_params` attribute contains additional parameters that are not needed to calculate the Keplerian model, but are needed in the calculation of the likelihood (e.g., jitter).

The `radvel.CompositeLikelihood` object is simply a container for multiple `radvel.RVLikelihood` objects

[7] https://lmfit.github.io/lmfit-py/

[8] Note that this representation is different from that used in `RadVel` versions <1.0.

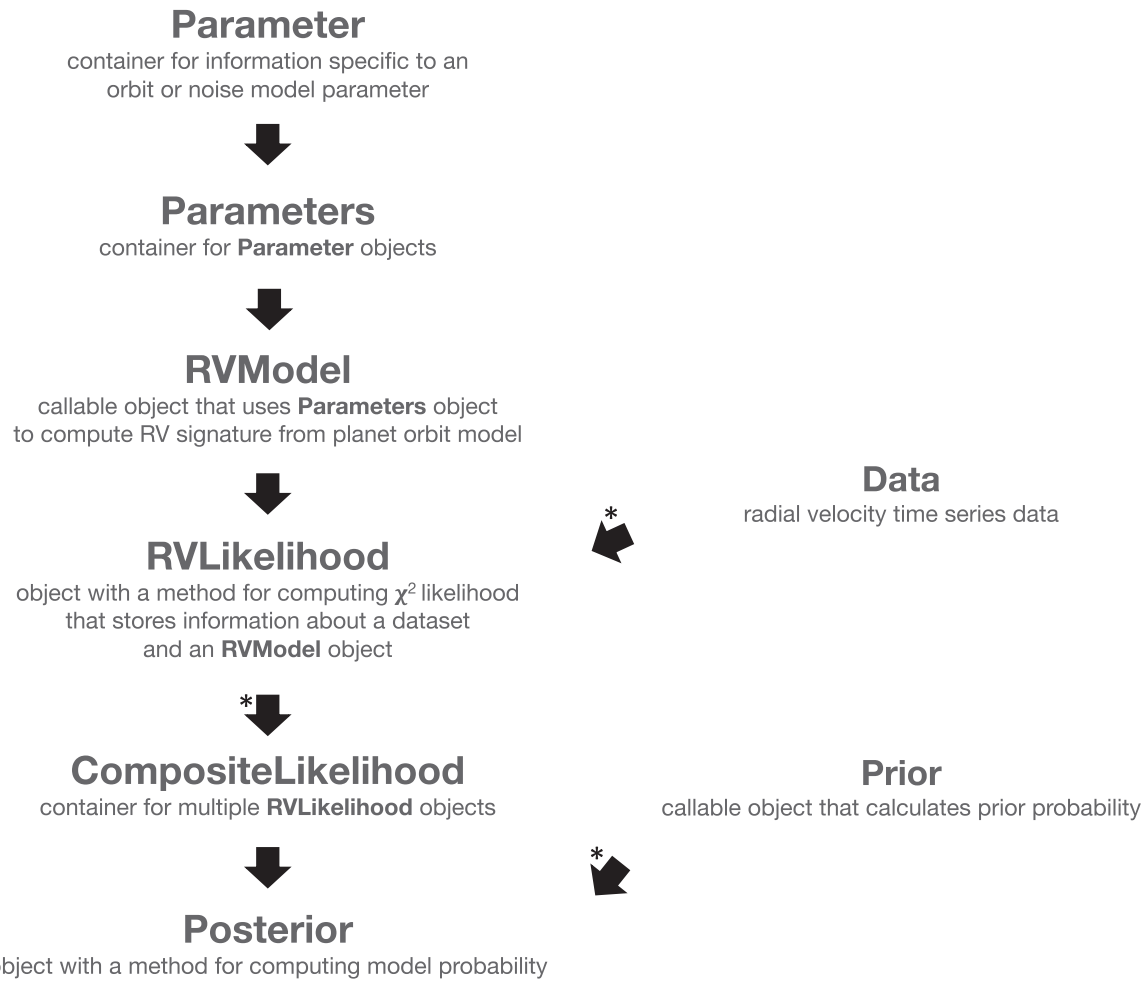


**Figure 2.** Class diagram for the `RadVel` package showing the relationships between the various objects contained within the `RadVel` package. Arrows point from attributes to their container objects (i.e., a `radvel.Parameter` object is an attribute of a `radvel.Parameters` object). Pertinent characteristics are summarized beneath each object. An asterisk next to an arrow indicates that the container object typically contains multiple attribute objects of the indicated type.

which is constructed in the case of multi-instrument data sets. The `logprob` method of the `radvel.CompositeLikelihood` adds the results of the `logprob` methods for all of the `radvel.RVLikelihood` objects contained within the `radvel.CompositeLikelihood`.

### *4.4. Posterior Object*

The `radvel.Posterior` object is very similar to the `radvel.Likelihood` object, but it also contains any user-defined priors. The `logprob` method of the `radvel.Posterior` object is then the natural log of the likelihood of the data given the model and priors.

### *4.5. Priors*

Priors are defined in the `radvel.prior` module and should be callable objects which return a single value to be multiplied by the likelihood. Several useful example priors are already implemented.

- `EccentricityPrior` can be used to set upper limits on the planet eccentricities.
- `GaussianPrior` can be used to assign a prior to a parameter value with a given center ($\mu$) and width ($\sigma$).
- `PositiveKPrior` can be used to force planet semi-amplitudes to be positive.[9]
- `HardBounds` prior is used to impose strict limits on parameter values.

Other priors are continuously being implemented and we encourage users to frequently check the API documentation on the RTD page for new priors.

[9] This should be used with extreme caution to avoid biasing results toward non-zero planet masses (Lucy & Sweeney 1971).

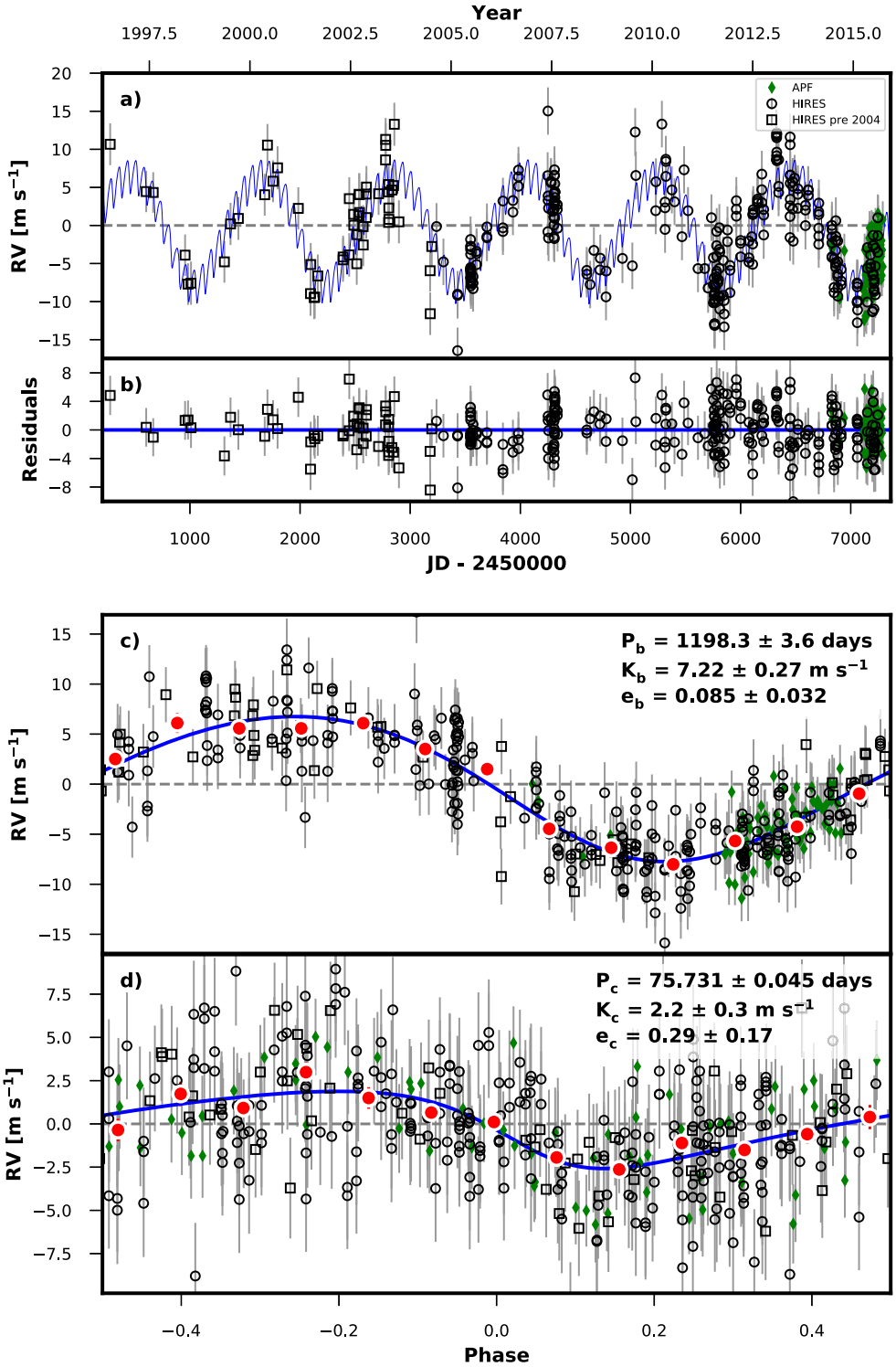


**Figure 3.** Example RV timeseries plot produced by `RadVel` for the HD 164922 fit with two planets and three spectrometers (see Section 6.2.2). This shows the MAP 2-planet Keplerian orbital model. The blue line is the two-planet model. The RV jitter terms are included in the plotted measurement uncertainties. (b) Residuals to the best fit two-planet model. (c) RVs phase-folded to the ephemeris of planet b. The Keplerian orbital models for all other planets have been subtracted. The small point colors and symbols are the same as in panel (a). Red circles are the same velocities binned in 0.08 units of orbital phase. The phase-folded model for planet b is shown as the blue line. Panel (d) is the same as panel (c), but for planet HD 164922 c.

(A color version of this figure is available in the online journal.)

## 5. Model Fitting

### 5.1. Maximum a Posteriori Fitting

The set of orbital parameters that maximizes the posterior probability (maximum a posteriori optimization, MAP) are found using Powell's method (Powell 1964) as implemented in `scipy.optimize.minimize`.[10] The code that performs the minimization can be found in the `radvel.fitting` submodule.

### 5.2. Uncertainty Estimation

The `radvel.mcmc` module handles the MCMC exploration of the posterior probability surface ($p$) to estimate parameter uncertainties. We use the MCMC package `emcee` (Foreman-Mackey et al. 2013), which employs an Affine Invariant sampler (Goodman & Weare 2010). The MCMC sampling generally explores a fairly wide range of parameter values but `RadVel` should not be treated as a planet discovery tool. The `RadVel` MCMC functionality is simply meant to determine the size and shape of the posterior probability density.

It is important to check that all of the independent walkers in the MCMC chains have adequately explored the same maximum on the posterior probability density surface and are not stuck in isolated local maxima. The Gelman-Rubin statistic (G-R, Gelman et al. 2003) is one metric to check for "convergence" by comparing the intra-chain variances to the inter-chain variances. G-R values close to unity indicate that the chains are converged.

We initially run a set of MCMC chains until the G-R statistic is less than 1.03 for all free parameters as a burn-in phase. These initial steps are the discarded and new chains are launched from the last positions. We note that this is a particularly conservative approach to burn-in that is enabled thanks to the very fast Keplerian model calculation and convergence of `RadVel`. Users may relax the G-R $< 1.03$ burn-in requirement via command-line flags or in the arguments of the `radvel.mcmc.mcmc` function. After the burn-in phase, we follow the prescription of Eastman et al. (2013) to check the MCMC chains for convergence after every 50 steps. The chains are deemed well-mixed, and the MCMC run is halted when the G-R statistic is less than 1.01 and the number of independent samples ($T_z$ statistic, Ford 2006) is greater than 1000 for all free parameters for at least five consecutive checks. We note that these statistics can not be calculated between walkers within an ensemble, so we calculate G-R and $T_z$ across completely independent ensembles of samplers. By default, we run eight independent ensembles in parallel with 50 walkers per ensemble for up to a maximum of 10000 steps per walker, or until convergence is reached.

These defaults can be customized on the command-line or in the arguments of the `radvel.mcmc.mcmc` function. Each of the independent ensembles are run on separate CPUs, so the number of ensembles should not exceed the number of CPUs available or significant slowdown will occur. Default initial step sizes for all free parameters are set to 10% of their value except period, which is set to 0.001% of the period. These initial step sizes can be customized by setting the `mcmcscale` attributes of the `radvel.Parameter` objects.

## 6. Examples

Users interact with `RadVel` either through a command-line interface (CLI) or through the Python API. Below, we run through an example RV analysis of HD 164922 from (Fulton et al. 2016) using the CLI. The Python API exposes additional functionality that may be used for special case fitting and is described in the advanced usage page on the RTD website.

### 6.1. Installation

To install `RadVel`, users must have working version of Python 2 or 3.[11] We recommend the Anaconda Python distribution.[12]

Users may then install `RadVel` from the Python Package Index (pip).[13,14]

```
1 $ pip install radvel
```

### 6.2. Command-line Interface

#### 6.2.1. Setup Files

The setup file is the central component in the command-line interface execution of `RadVel`. This file is a Python script that defines the number of planets in the system; initial parameter guesses; stellar parameters if present; priors; reads and defines data sets; initializes a model object; and defines metadata associated with the run. One setup file should be produced for each planetary system that the user attempts to model using the command-line interface. Two example setup files can be downloaded from the GitHub repo. A complete example is provided below with inline comments to describe the various components.

[10] Any of the optimization methods implemented in `scipy.optimize.minimize`, which do not require pre-calculation of derivatives (e.g., Nelder-Mead), can be swapped in with a simple modification to the code in the `radvel.fitting` module.

[11] As the community transitions from Python 2 to 3, we will likely drop support for Python 2.

[12] https://www.anaconda.com

[13] https://pypi.python.org/pypi

[14] Early `RadVel` adopters may see installation conflicts with early versions of `RadVel`. We recommend manually removing old versions and performing a fresh install with `pip`.

```
# Required packages for setup
import os
import pandas as pd
import numpy as np
import radvel as rv

# name of the star used for plots and tables
starname = 'HD164922'

# number of planets in the system
nplanets = 2

# list of instrument names. Can be whatever
# you like but should match 'tel' column in the
# input data file.
instnames = ['k', 'j', 'a']

# number of instruments with unique
# velocity zero-points
ntels = len(instnames)

# Fitting basis, see rv.basis.BASIS_NAMES
# for available basis names
fitting_basis = 'per tc secosw sesinw k'

# reference epoch for plotting purposes
bjd0 = 2450000.

# (optional) customize letters
# corresponding to planet indicies
planet_letters = {1: 'b', 2:'c'}

# Define prior centers (initial guesses) in a basis
# of your choice (need not be in the fitting basis)
# initialize Parameters object
params = rv.Parameters(nplanets,
        basis = 'per tc e w k')

# period of 1st planet
params['per1'] = rv.Parameter(value = 1206.3)
# time of inferior conjunction of 1st planet
params['tc1'] = rv.Parameter(value = 2456779.)
# eccentricity
params['e1'] = rv.Parameter(value = 0.01)
# argument of periastron of the star's orbit
params['w1'] = rv.Parameter(value = np.pi/2)
# velocity semi-amplitude
params['k1'] = rv.Parameter(value = 10.0)

# same parameters for 2nd planet ...
params['per2'] = rv.Parameter(value = 75.771)
params['tc2'] = rv.Parameter(value = 2456277.6)
params['e2'] = rv.Parameter(value = 0.01)
params['w2'] = rv.Parameter(value = np.pi/2)
params['k2'] = rv.Parameter(value = 1.0)

# slope and curvature
params['dvdt'] = rv.Parameter(value = 0.0)
params['curv'] = rv.Parameter(value = 0.0)

# zero-points and jitter terms for each instrument
params['gamma_j'] = rv.Parameter(1.0)
params['jit_j'] = rv.Parameter(value = 2.6)
```

(Continued)

```

# Convert input orbital parameters into the
# fitting basis
params = params.basis.to_any_basis(
          params,fitting_basis)

# Set the 'vary' attributes of each of the parameters
# in the fitting basis. A parameter's 'vary' attribute
# should be set to False if you wish to hold it fixed
# during the fitting process. By default, all 'vary'
# parameters are set to True.
params['dvdt'].vary = False
params['curv'].vary = False

# Load radial velocity data, in this example the
# data are contained in a csv file, the resulting
# dataframe or must have 'time', 'mnvel', 'errvel',
# and 'tel' keys the velocities are expected
# to be in m/s
path = os.path.join(rv.DATADIR, '164922_fixed.txt')
data = pd.read_csv(path, sep = ' ')

# Define prior shapes and widths here.
priors = [
   # Keeps eccentricity <1 for all planets
   rv.prior.EccentricityPrior(nplanets ),
   # Keeps K > 0 for all planets
   rv.prior.PositiveKPrior(nplanets ),
   # Hard limits on jitter parameters
   rv.prior.HardBounds('jit_j', 0.0, 10.0),
   rv.prior.HardBounds('jit_k', 0.0, 10.0),
   rv.prior.HardBounds('jit_a', 0.0, 10.0)
]

# abscissa for slope and curvature terms
# (should be near mid-point of time baseline)
time_base = 0.5*(data.time.min() + data.time.max())

# optional argument that can contain stellar mass
# in solar units (mstar) and uncertainty (mstar_err).
# If not set, mstar will be set to nan.
stellar = dict(mstar = 0.874, mstar_err = 0.012)
```

### 6.2.2. Workflow

The `RadVel` CLI is provided for the convenience of the user and is the standard operating mode of `RadVel`. It acts as a wrapper for much of the underlying API. Most users will likely find this to be the easiest way to run `RadVel` fits and produce the standard outputs.

Here, we provide a walkthrough for a basic `RadVel` fit for a multi-planet system with RV data collected using three different instruments. The data for this example is taken from Fulton et al. (2016). The first step is to create a setup file for the fit. In this case, we have provided a setup file in the GitHub

repo (example_planets/HD164922.py). Once we have a setup file, we always need to run a MAP fit first. This is done using the radvel fit command:

```
$ radvel fit -s /path/to/HD164922.py
```

This command will produce some text output summarizing the result which shows the final parameter values after the MAP fit. A new directory will be created in the current working directory with the same name of the setup file. In this case, a directory named HD164922 was created and it contains a HD164922_post_obj.pkl file and a HD164922_radvel.stat file. Both of these files are used internally by RadVel to keep track of the components of the fit that have already been run. All of the output associated with this RadVel fit will be put in this directory.

It is useful to inspect the MAP fit using the radvel plot command:

```
$ radvel plot -t rv -s /path/to/HD164922.py
```

This will produce a new PDF file in the output directory called HD164922_rv_multipanel.pdf that looks very similar to Figure 3. The annotated parameter uncertainties will only be printed if MCMC has already been run. The -t rv flag tells RadVel to produce the standard RV timeseries plot. Several other types of plots can also be produced using the radvel plot command.

If the fit looks good, then the next step is to run an MCMC exploration to estimate parameter uncertainties:

```
$ radvel mcmc -s /path/to/HD164922.py
```

Once the MCMC chains have converged or the maximum step limit is reached (see Section 5.2), two additional output files will be produced: HD164922_chains.csv.tar.bz2 and HD164922_post_summary.csv. The chains.csv.tar.bz2 file contains each step in the MCMC chains. All of the independent ensembles and walkers have been combined such that there is one column per free parameter. The post_summary.csv file contains the median and 68th percentile credible intervals for all free parameters.

Now that the MCMC has finished we can make some additional plots:

```
$ radvel plot -t rv -s /path/to/HD164922.py
$ radvel plot -t corner -s /path/to/HD164922.py
$ radvel plot -t trend -s /path/to/HD164922.py
```

The RV timeseries plot (Figure 3) now includes the parameter uncertainties in the annotations. In addition, a "corner" or triangle plot showing all parameter covariances produced by the corner Python package (Figure 4, Foreman-Mackey et al. 2016), and a "trend" plot showing the evolution of the parameter values as they step through the MCMC chains will be produced (Figure 5).

In this case, we have also defined the stellar mass (mstar) and uncertainty (mstar_err) in the stellar dictionary in the setup file. This allows us to use the radvel derive command to convert the velocity semi-amplitude ($K$), orbital period ($P$), and eccentricity ($e$) into a minimum planet mass ($M \sin i$):

```
$ radvel derive -s /path/to/HD164922.py
$ radvel plot -t derived -s /path/to/HD164922.py
```

This produces the file HD164922_derived.csv.tar.bz2 in the output directory which contains columns for each of the derived parameters. For the case of transiting planets, planetary radii can also be specified (see the epic203771098.py example file) to allow the computation of planet densities. Synthetic Gaussian posterior distributions are created for these stellar parameters and these synthetic posteriors are multiplied by the real posterior distributions in order to properly account for the uncertainties in both the stellar and orbital parameters. A corner plot for the derived parameters is created using the radvel plot -t derived command.

An optional model comparison table (Table 3) can be created using the radvel bic command:

```
$ radvel bic -t nplanets -s /path/to/HD164922.py
```

This produces a table summarizing model comparisons with 0 to $N_{\rm pl}$ planets where $N_{\rm pl}$ is the number of planets in the system specified in the setup file. This allows the user to compare models with fewer planets to ensure that their adopted model is statistically favored. The comparisons are performed by fixing the jitter parameters to their MAP values from the full $N_{\rm pl}$ planet fit.

RadVel can also produce publication-quality plots, tables, and a summary report in LaTEX and PDF form. The functionality contained in the radvel.RadvelReport and radvel.TexTable objects depend on the existence of a setup file (see Section 6.2.1), which is usually only present when utilizing the CLI. RadVel reports contain LaTEX tables showing the MAP values and credible intervals for all orbital parameters, a summary of non-uniform priors, and a model comparison table.[15] A RV timeseries plot and a corner plot are also included. If stellar and/or planetary parameters are specified in the setup file (see Section 6.2.1), then a second corner plot is included with the derived parameters including planet masses ($M \sin i$) and densities (if transiting and planet radii given).

[15] only if the radvel bic command has been run.

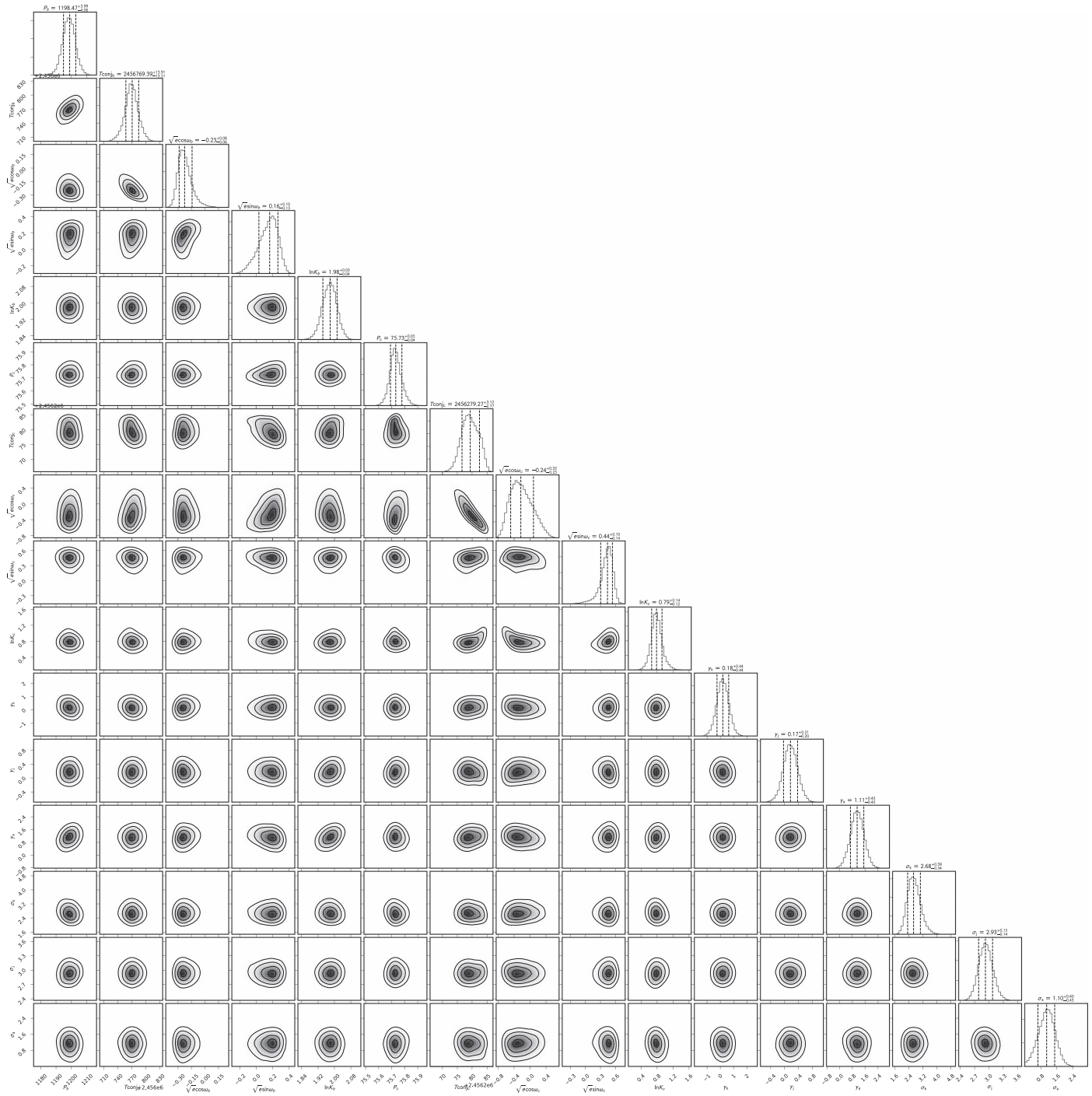

**Figure 4.** Corner plot showing all joint posterior distributions derived from the MCMC sampling. Histograms showing the marginalized posterior distributions for each parameter are also included. This plot is produced as part of the CLI example for HD 164922 as discussed in Section 6.2.2.

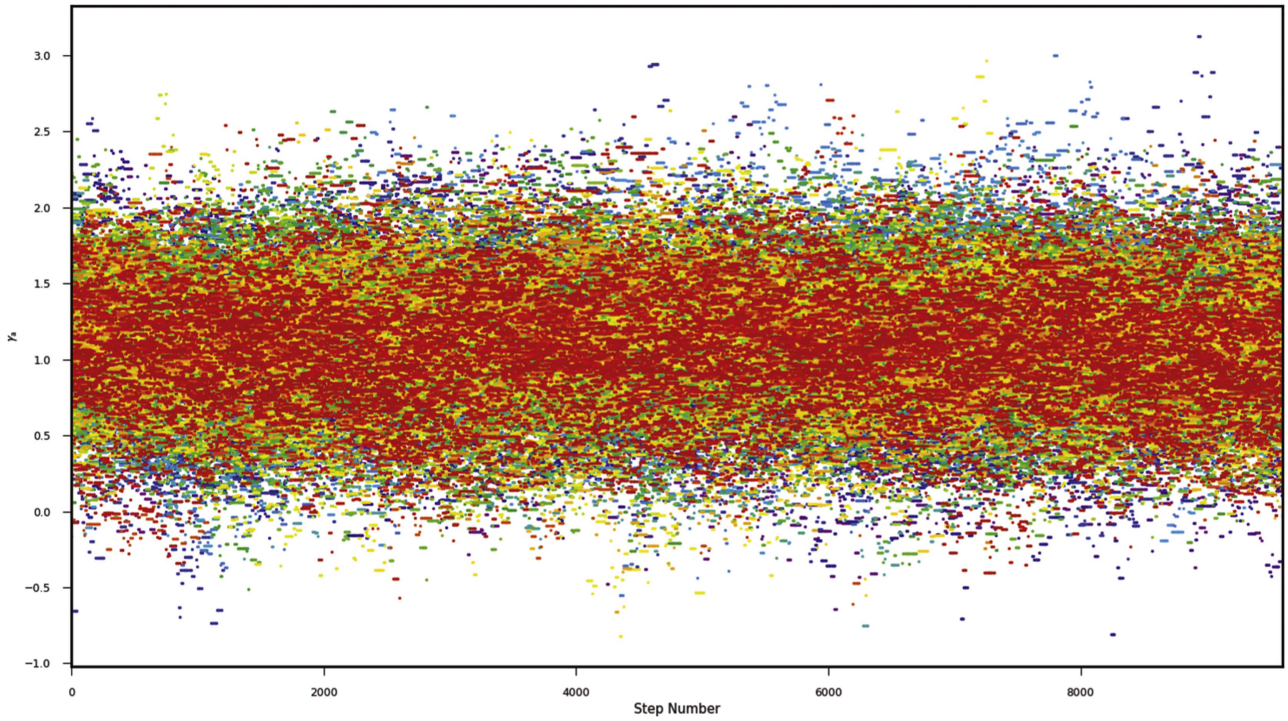


**Figure 5.** Trend plot produced by `RadVel` showing parameter evolution during the MCMC exploration. This plot is produced as part of the CLI example for HD 164922 as discussed in Section 6.2.2. As an example, we show only a single page of the PDF output. In practice, there will be one plot like this for each free parameter. Each independent ensemble and walker is plotted in different colors. If the MCMC runs have successfully converged, these plots should look like white noise will all colors mixed together. Large-scale systematics and/or single chains (colors) isolated from the rest likely indicate problems with the initial parameter guesses and/or choices of priors.
(A color version of this figure is available in the online journal.)

**Table 3**
Model Comparison

| Statistic | 0 Planets | 1 Planet | 2 Planets (adopted) |
|---|---|---|---|
| $N_{\rm data}$ (number of measurements) | 401 | 401 | 401 |
| $N_{\rm free}$ (number of free parameters) | 3 | 8 | 13 |
| rms (rms of residuals in m s$^{-1}$) | 4.97 | 3.01 | 2.91 |
| $\chi^2$ (jitter fixed) | 1125.56 | 431.7 | 397.29 |
| $\chi^2_\nu$ (jitter fixed) | 2.83 | 1.1 | 1.02 |
| $\ln \mathcal{L}$ (natural log of the likelihood) | −1356.54 | −1009.61 | −992.41 |
| BIC (Bayesian information criterion) | 2731.06 | 2067.17 | 2062.74 |

The command

```
1 $ radvel report -s /path/to/HD164922.py
```

creates the LaTEX file `HD164922_results.tex` in the output directory and compiles it into a PDF using the `pdflatex` command.[16] The summary PDF is a great way to send `RadVel` results to colleagues, and because the LaTEX code for the report is also saved the tables can be copied directly into a manuscript to avoid transcription errors.

## 7. RadVel and the Community

### *7.1. Support*

Please report any bugs or problems to the issues tracker on the GitHub repo page.[17] Bugfixes in response to issue reports will be included in the next version release and will be

[16] The `pdflatex` binary is assumed to be in the system's path by default, but the full path to the binary may be specified using the `-latex-compiler` flag if it installed in a non-standard location.

[17] https://github.com/California-Planet-Search/radvel/issues

available by download from PyPI (via `pip install radvel --upgrade`) at that time.

### 7.2. Contributing Code

The `RadVel` codebase is open-source and we encourage contributions from the community. All development will take place on GitHub. Developers should fork the repo and submit pull requests into the `next-release` base branch. These pull requests will be reviewed by the maintainers and merged into the `master` branch to be included in the next tagged release. We ask that developers follow these simple guidelines when contributing code:

- Do not break any existing functionality. This will automatically be checked when a pull request is created via Travis Continuous Integration.[18]
- Develop with support for Python 3 and backward-compatibility with Python 2 where possible.
- Code according to PEP8 standards.[19]
- Document your code using Napolean-compatible docstrings,[20] following the Google Python Style Guide.[21]
- Include unit tests that touch any new code using the `nosetests` framework.[22] Example unit tests can be found in the `radvel/radvel/tests` subdirectory of the repo.

### 7.3. Future Work

`RadVel` is currently under active development and will grow to incorporate the modeling needs of the exoplanet community. Specific areas for future include:

- *Gaussian Process noise modeling*. Implement likelihoods that incorporate Gaussian process descriptions of RV variability.

The authors have several potential improvements to `RadVel` in mind or currently under development. We encourage the community to suggest other wish list items or contribute insight and/or code to ongoing development efforts using the GitHub issue tracker. Our current wish list is as follows:

- Improve performance of the dictionary key-based string indexing.
- Include other types of model comparisons in the `radvel bic` command (e.g., eccentric versus circular fits).
- Add ability to simultaneously fit other data sets (e.g., transit photometry, transit timing variations, astrometry).

[18] https://travis-ci.org
[19] https://www.python.org/dev/peps/pep-0008/
[20] https://sphinxcontrib-napoleon.readthedocs.io
[21] http://google.github.io/styleguide/pyguide.html
[22] http://pythontesting.net/framework/nose

## 8. Conclusion

We have provided a flexible, open-source toolkit for modeling RV data written in object-oriented Python. The package is designed to model the RV orbits of systems with multiple planets and data collected from multiple instruments. It features a convenient command-line interface and a scriptable API. `RadVel` utilizes a fast Keplerian solver written in C and robust, real-time convergence checking of the MCMC chains. It supports multiple parameterizations of the RV orbit and contains convenience functions for converting between parameterizations. `RadVel` has already been used to model RV orbits in at least nine refereed publications.[23] In addition, Teske et al. (2017) demonstrated that `RadVel` produces results consistent with the results of the `Systemic` RV fitting package (Meschiari et al. 2009; Meschiari & Laughlin 2010). We encourage the community to continue using `RadVel` for their RV modeling needs and to contribute to its future development.

E.A.P. acknowledges support from Hubble Fellowship grant *HST*-HF2-51365.001-A awarded by the Space Telescope Science Institute, which is operated by the Association of Universities for Research in Astronomy, Inc. for NASA under contract NAS 5-26555. E.S. is supported by a post-graduate scholarship from the Natural Sciences and Engineering Research Council of Canada.

## ORCID iDs

Benjamin J. Fulton https://orcid.org/0000-0003-3504-5316
Erik A. Petigura https://orcid.org/0000-0003-0967-2893
Sarah Blunt https://orcid.org/0000-0002-3199-2888
Evan Sinukoff https://orcid.org/0000-0002-5658-0601

## References

Akeson, R. L., Chen, X., Ciardi, D., et al. 2013, PASP, 125, 989
Borucki, W. J., Koch, D., Basri, G., et al. 2010, Sci, 327, 977
Campbell, B., Walker, G. A. H., & Yang, S. 1988, ApJ, 331, 902
Christiansen, J. L., Vanderburg, A., Burt, J., et al. 2017, AJ, 154, 122
Crossfield, I. J. M., Ciardi, D. R., Isaacson, H., et al. 2017, AJ, 153, 255
Danby, J. M. A. 1988, Fundamentals of Celestial Mechanics
Deck, K. M., Agol, E., Holman, M. J., & Nesvorný, D. 2014, ApJ, 787, 132
Eastman, J., Gaudi, B. S., & Agol, E. 2013, PASP, 125, 83
Ford, E. B. 2006, ApJ, 642, 505
Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, PASP, 125, 306
Foreman-Mackey, D., Vousden, W., Price-Whelan, A., et al. 2016, corner.py: corner.py v2.0.0, doi:10.5281/zenodo.53155
Fulton, B., & Petigura, E. 2017, RadVel: Radial Velocity Fitting Toolkit, doi:10.5281/zenodo.580821
Fulton, B. J., Howard, A. W., Weiss, L. M., et al. 2016, ApJ, 830, 46

[23] (Sinukoff et al. 2017a, 2017b; Petigura et al. 2017; Crossfield et al. 2017; Weiss et al. 2017; Grunblatt et al. 2017; Christiansen et al. 2017; Teske et al. 2017).

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. 2003, Bayesian Data Analysis (2nd edn.; Chapman and Hall)
Goodman, J., & Weare, J. 2010, Comm. App. Math. Comp. Sci., 5, 65
Grunblatt, S. K., Huber, D., Gaidos, E., et al. 2017, AJ, 154, 254
Iglesias-Marzoa, R., López-Morales, M., & Jesús Arévalo Morales, M. 2015, PASP, 127, 567
Latham, D. W., Stefanik, R. P., Mazeh, T., Mayor, M., & Burki, G. 1989, Natur, 339, 38
Lucy, L. B., & Sweeney, M. A. 1971, AJ, 76, 544
Marcy, G. W., & Butler, R. P. 1996, ApJL, 464, L147
Mayor, M., & Queloz, D. 1995, Natur, 378, 355
Mede, K., & Brandt, T. D. 2017, AJ, 153, 135
Meschiari, S., & Laughlin, G. P. 2010, ApJ, 718, 543
Meschiari, S., Wolf, A. S., Rivera, E., et al. 2009, PASP, 121, 1016
Murray, C. D., & Correia, A. C. M. 2010, in Keplerian Orbits and Dynamics of Exoplanets, ed. S. Seager (Tucson, AZ: Univ. Arizona Press), 15
Murray, C. D., & Dermott, S. F. 1999, in Solar System Dynamics, ed. C. D. Murray (Cambridge: Cambridge Univ. Press)
Newville, M., Stensitzki, T., Allen, D. B., & Ingargiola, A. 2014, LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python, doi:10.5281/zenodo.11813
Petigura, E. A., Sinukoff, E., Lopez, E. D., et al. 2017, AJ, 153, 142
Powell, M. J. D. 1964 Comp. J, 7, 155
Sinukoff, E., Howard, A. W., Petigura, E. A., et al. 2017a, AJ, 153, 70
Sinukoff, E., Howard, A. W., Petigura, E. A., et al. 2017b, AJ, 153, 271
Teske, J. K., Wang, S. X., Wolfgang, A., et al. 2017, arXiv:1711.01359
Weiss, L. M., Deck, K. M., Sinukoff, E., et al. 2017, AJ, 153, 265
Wright, J. T., & Howard, A. W. 2009, ApJS, 182, 205