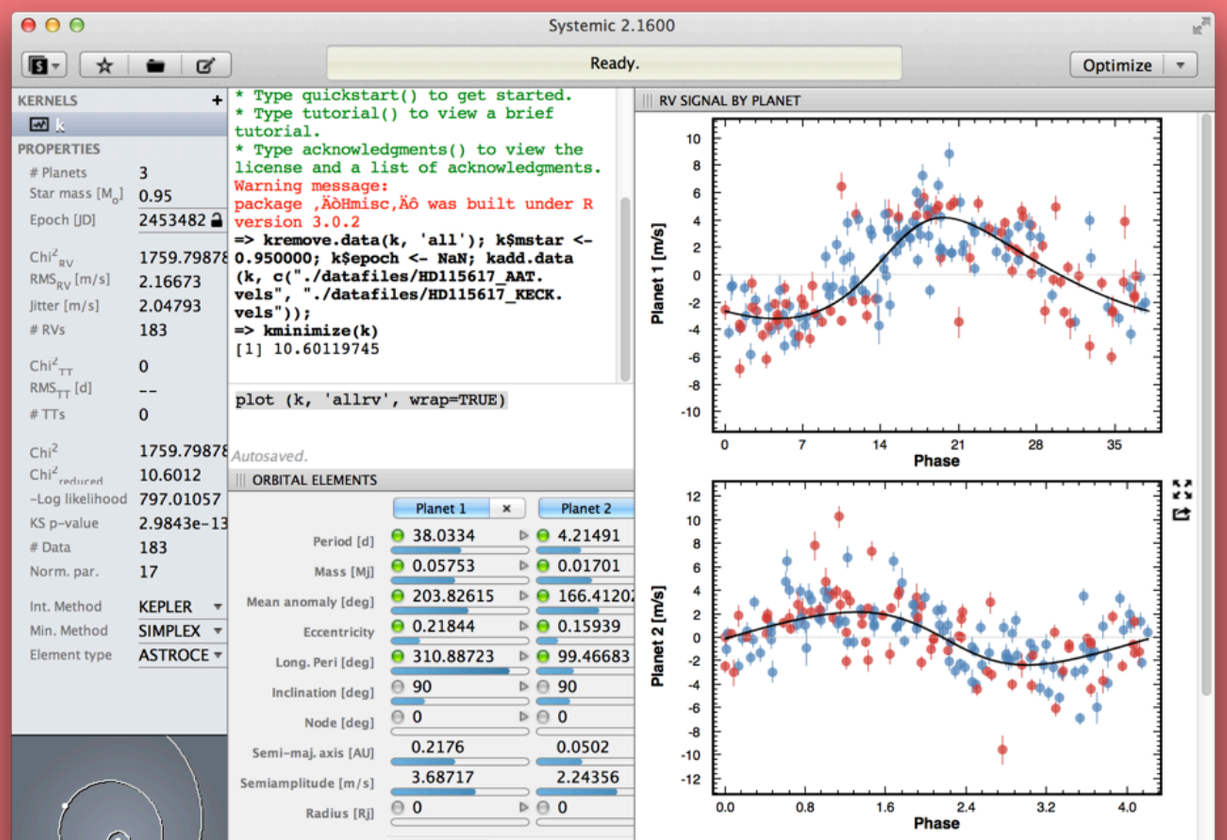
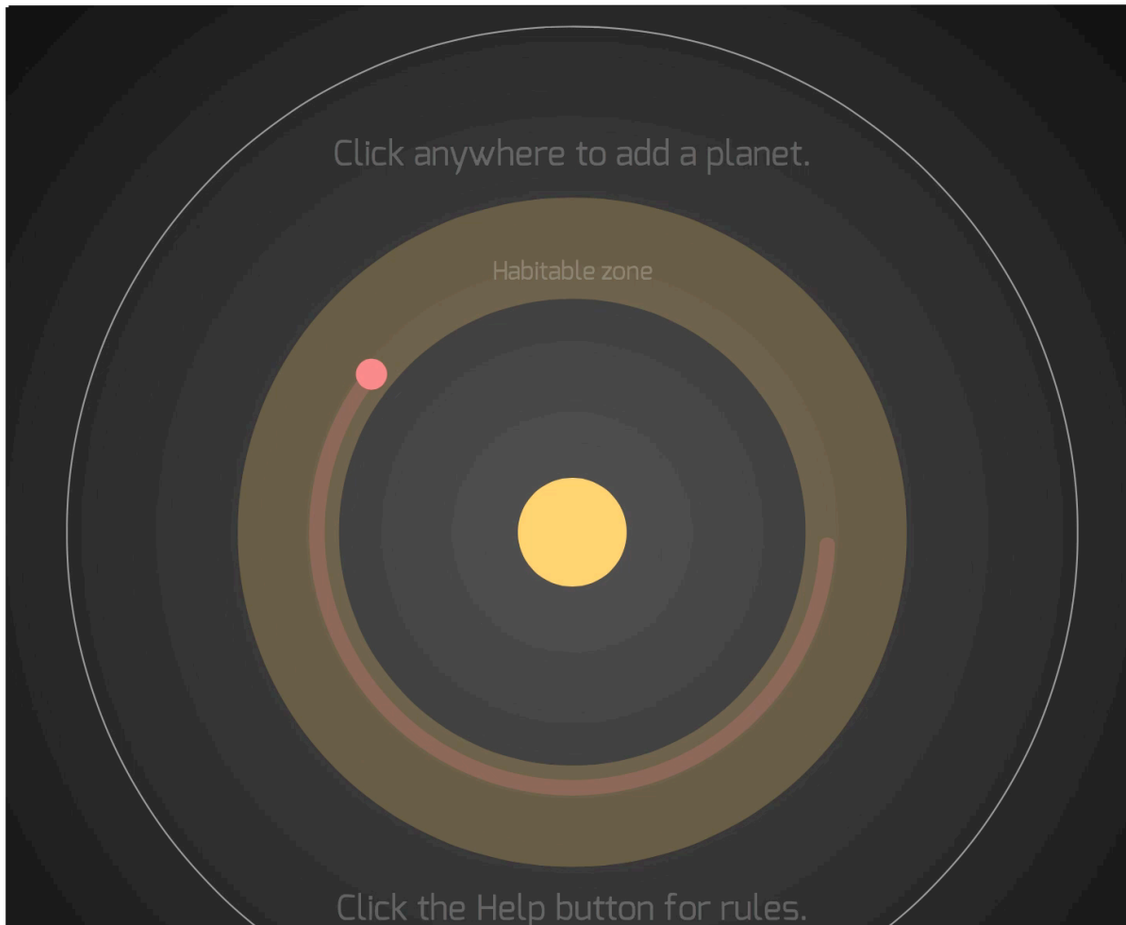
You, Too, Can Make Spiffy Online Web Apps for Outreach and $$$

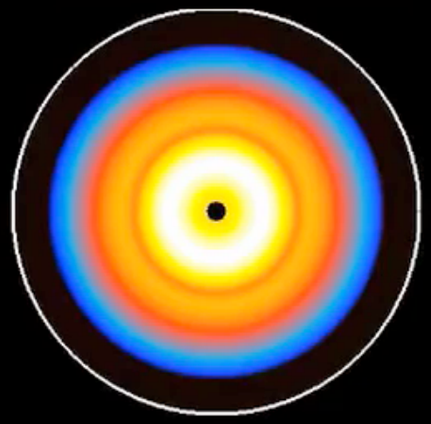
Stefano Meschiari
UT Austin, W.J. McDonald Fellow

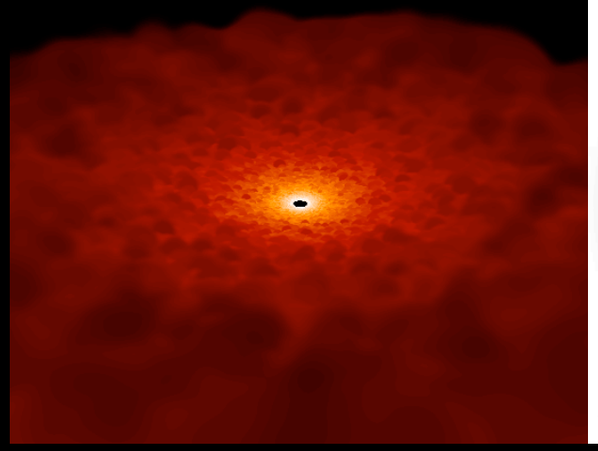
UT Austin GSPS
March 21, 2014

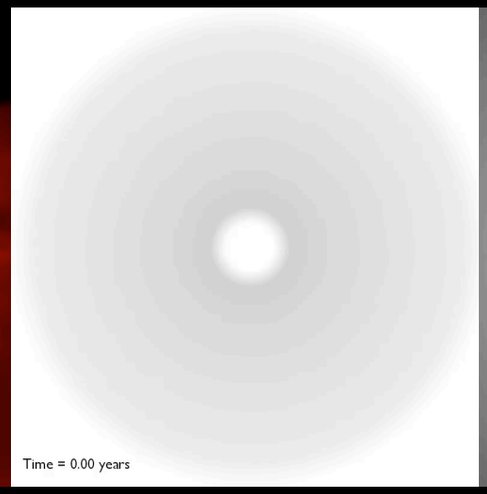


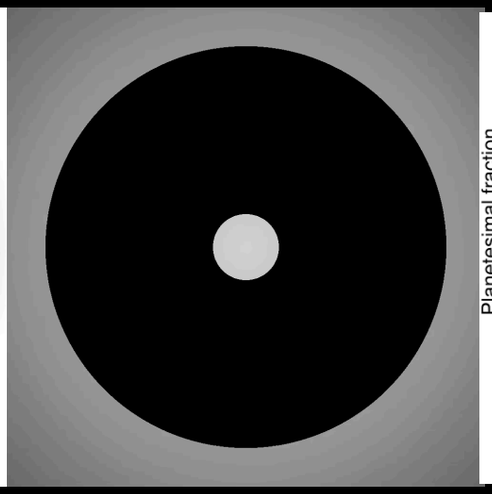


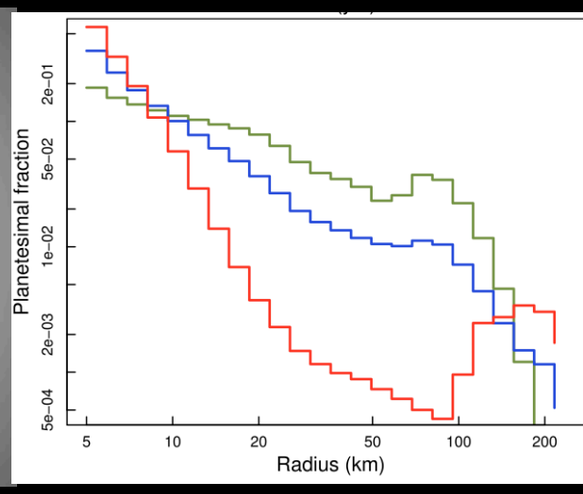
If you don’t know who I am:



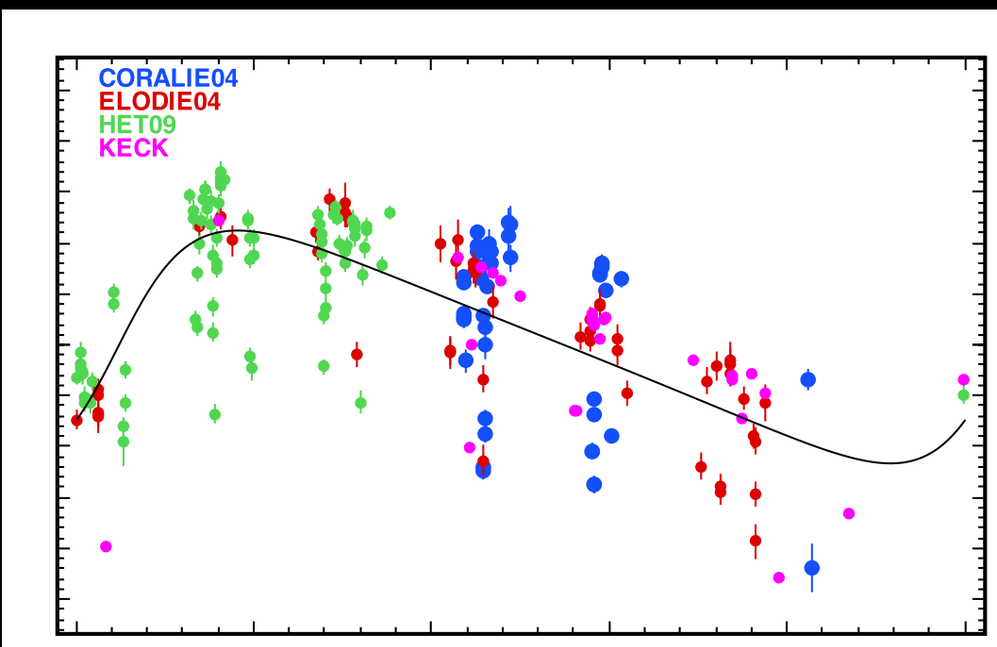


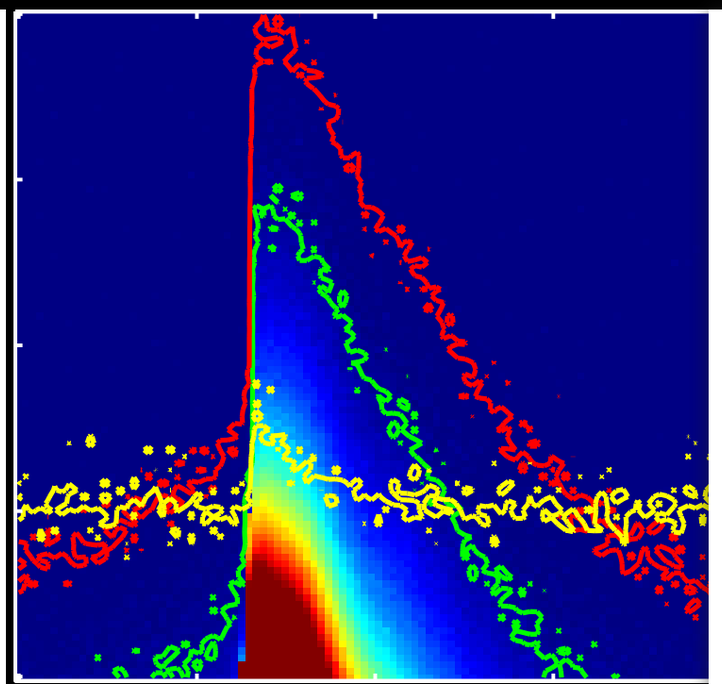


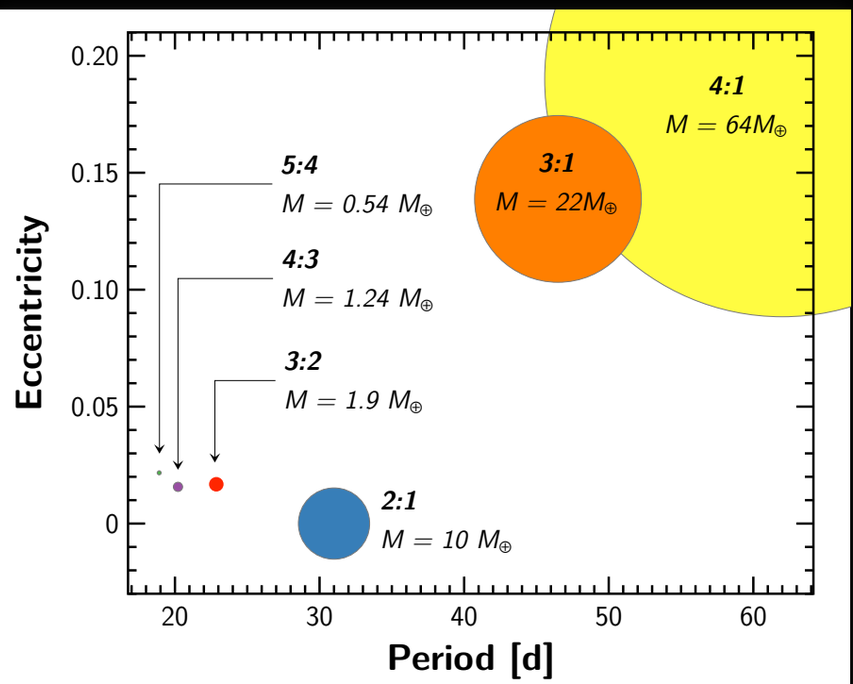




My science shtick is planet formation & exoplanet detection.







I'm also very interested in outreach, especially when it doesn't involve me physically standing in front of an audience.

Shy panda

Systemic:

One Software Package to Rule Them All

What is Systemic?

Systemic is an open-source software package for analyzing and modelling exoplanetary time series (primarily Radial Velocities and transit timing)

Systemic:

One Software Package to Rule Them All

What is Systemic?

Systemic is an open-source software package for analyzing and modelling exoplanetary time series (primarily Radial Velocities and transit timing)

What is “All”?

Systemic:

One Software Package to Rule Them All

What is Systemic?

Systemic **is an open-source software package for analyzing and modelling exoplanetary time series (primarily Radial Velocities and transit timing)**

What is "All"?

- **Science**
- **Teaching & outreach**
- **A fun treat**

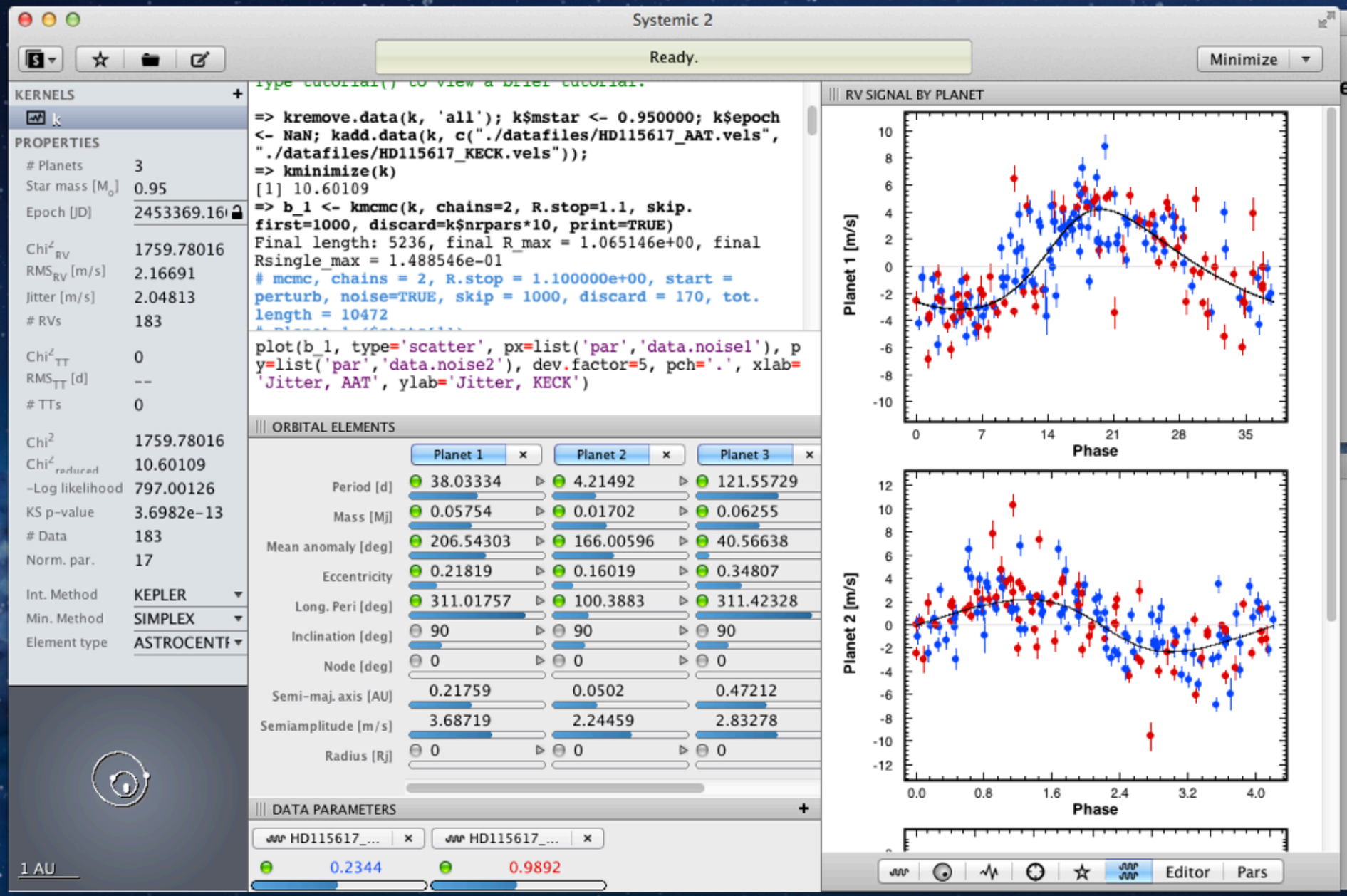
Science

Collaborators:

Greg Laughlin, Russell Hanson,
Jenn Burt, Steve Vogt (UCSC),
Paul Butler (Carnegie), Joel Green (UT)

Systemic 2

http://www.stefanom.org/systemic



Meschiari+ ‘09, ‘10, ‘11, Meschiari+ ‘14 (in prep.)

Systemic 2

Ready.

Minimize

KERNELS

k

PROPERTIES

# Planets	3
Star mass [M_o]	0.95
Epoch [JD]	2453369.16
Chi^2_{RV}	1759.78016
RMS_{RV} [m/s]	2.16691
Jitter [m/s]	2.04813
# RVs	183
Chi^2_{TT}	0
RMS_{TT} [d]	--
# TTs	0
Chi^2	1759.78016
$Chi^2_{reduced}$	10.60109
-Log likelihood	797.00126
KS p-value	3.6982e-13
# Data	183
Norm. par.	17
Int. Method	KEPLER
Min. Method	SIMPLEX
Element type	ASTROCENTR

1 AU

```
Type tutorial() to view a brief tutorial.

=> kremove.data(k, 'all'); k$mstar <- 0.950000; k$epoch
<- NaN; kadd.data(k, c("./datafiles/HD115617_AAT.vels",
"./datafiles/HD115617_KECK.vels"));
=> kminimize(k)
[1] 10.60109
=> b_1 <- kmcmc(k, chains=2, R.stop=1.1, skip.
first=1000, discard=k$nrpars*10, print=TRUE)
Final length: 5236, final R_max = 1.065146e+00, final
Rsingle_max = 1.488546e-01
# mcmc, chains = 2, R.stop = 1.100000e+00, start =
perturb, noise=TRUE, skip = 1000, discard = 170, tot.
length = 10472
```

```
plot(b_1, type='scatter', px=list('par','data.noise1'), p
y=list('par','data.noise2'), dev.factor=5, pch='.', xlab=
'Jitter, AAT', ylab='Jitter, KECK')
```

ORBITAL ELEMENTS

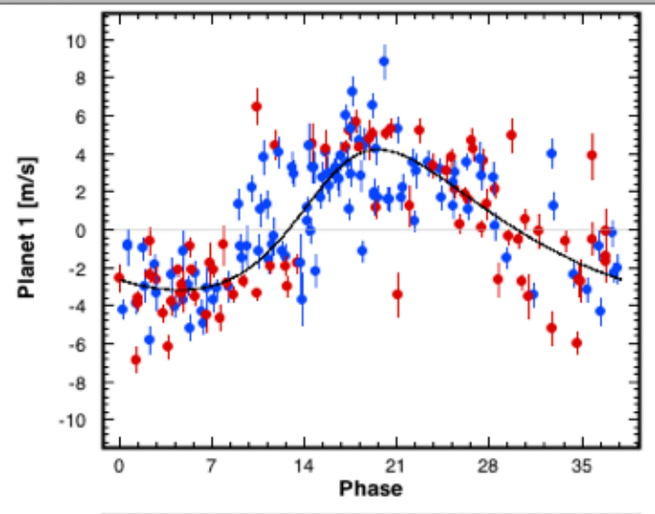
	Planet 1	Planet 2	Planet 3
Period [d]	38.03334	4.21492	121.55729
Mass [Mj]	0.05754	0.01702	0.06255
Mean anomaly [deg]	206.54303	166.00596	40.56638
Eccentricity	0.21819	0.16019	0.34807
Long. Peri [deg]	311.01757	100.3883	311.42328
Inclination [deg]	90	90	90
Node [deg]	0	0	0
Semi-maj. axis [AU]	0.21759	0.0502	0.47212
Semiamplitude [m/s]	3.68719	2.24459	2.83278
Radius [Rj]	0	0	0

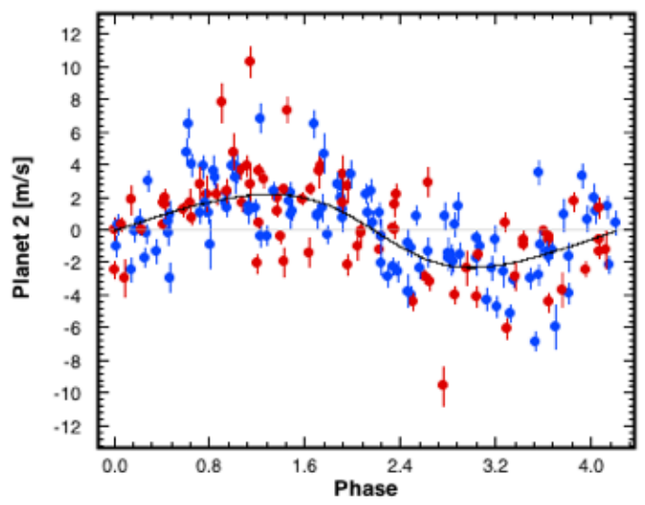
DATA PARAMETERS

HD115617_... | HD115617_...

0.2344 | 0.9892

RV SIGNAL BY PLANET

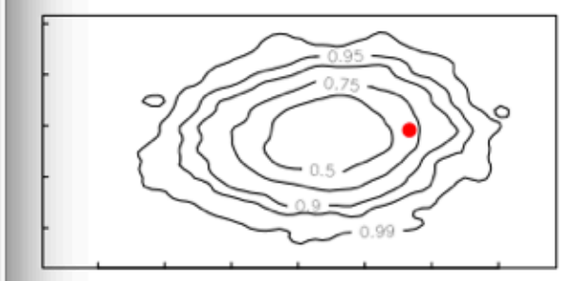




Editor | Pars

Quartz 2

eriod of planet 1 vs Period of planet 2

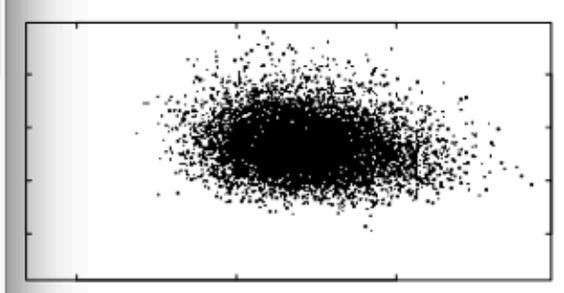


37.80 37.90 38.00 38.10

Period of planet 1

Quartz 3 [*]

Jitter, AAT vs Jitter, KECK

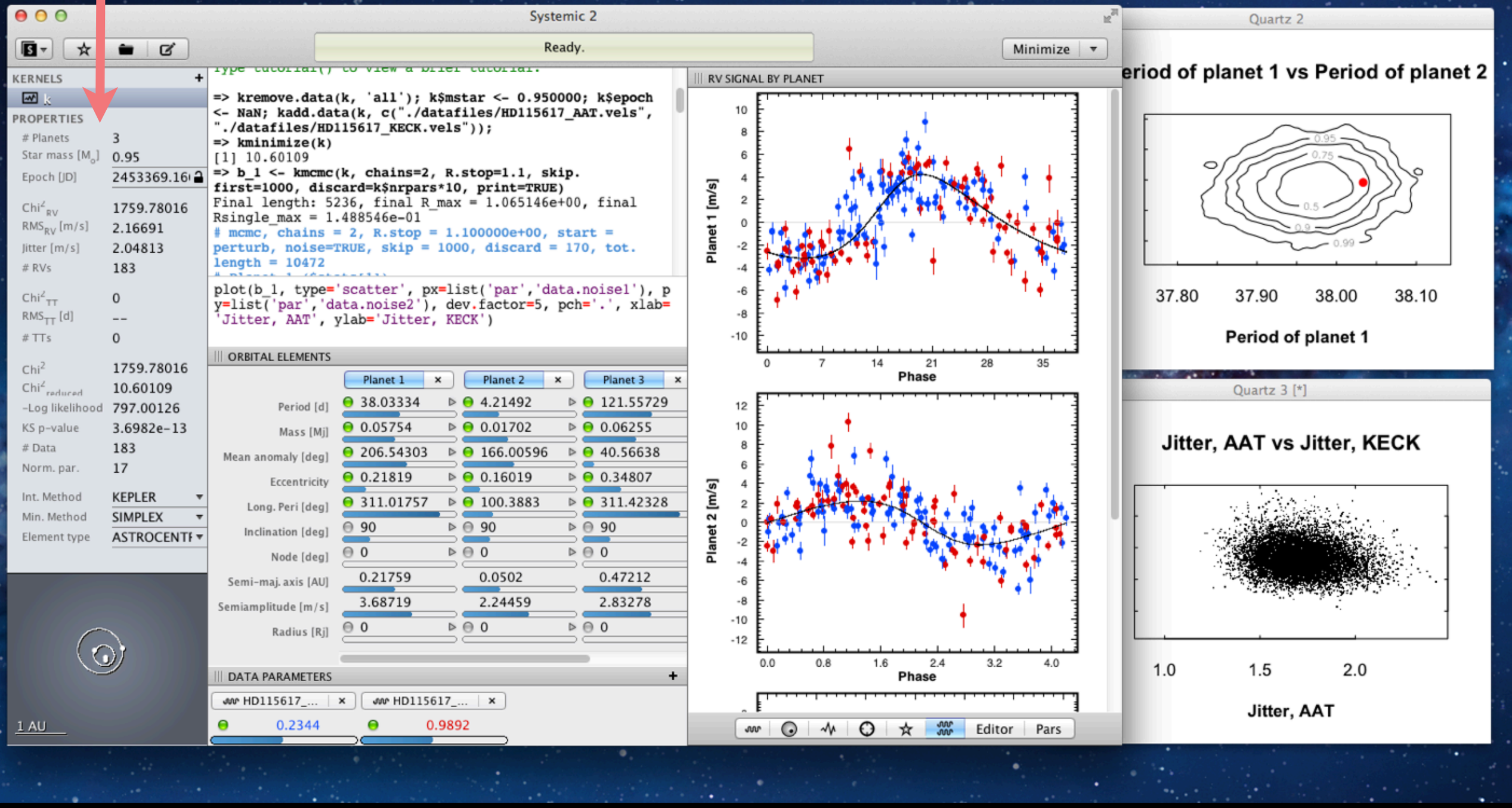


1.0 1.5 2.0

Jitter, AAT

Model statistics

(Chi², log likelihood, etc.)



Model statistics
(Chi², log likelihood, etc.)

Command line

Systemic

Ready.

Minimize

KERNELS

k

PROPERTIES

# Planets	3
Star mass [M_o]	0.95
Epoch [JD]	2453369.16
Chi^2_{RV}	1759.78016
RMS_{RV} [m/s]	2.16691
Jitter [m/s]	2.04813
# RVs	183
Chi^2_{TT}	0
RMS_{TT} [d]	--
# TTs	0
Chi^2	1759.78016
$Chi^2_{reduced}$	10.60109
-Log likelihood	797.00126
KS p-value	3.6982e-13
# Data	183
Norm. par.	17
Int. Method	KEPLER
Min. Method	SIMPLEX
Element type	ASTROCENTR

1 AU

```
=> kremove.data(k, 'all'); k$mstar <- 0.950000 ; k$epoch
<- NaN; kadd.data(k, c("./datafiles/HD115617_AAT.vels",
"./datafiles/HD115617_KECK.vels"));
=> kminimize(k)
[1] 10.60109
=> b_1 <- kmcmc(k, chains=2, R.stop=1.1, skip.
first=1000, discard=k$nrpars*10, print=TRUE)
Final length: 5236, final R_max = 1.065146e+00, final
Rsingle_max = 1.488546e-01
# mcmc, chains = 2, R.stop = 1.100000e+00, start =
perturb, noise=TRUE, skip = 1000, discard = 170, tot.
length = 10472
```

```
plot(b_1, type='scatter', px=list('par','data.noise1'), p
y=list('par','data.noise2'), dev.factor=5, pch='.', xlab=
'Jitter, AAT', ylab='Jitter, KECK')
```

ORBITAL ELEMENTS

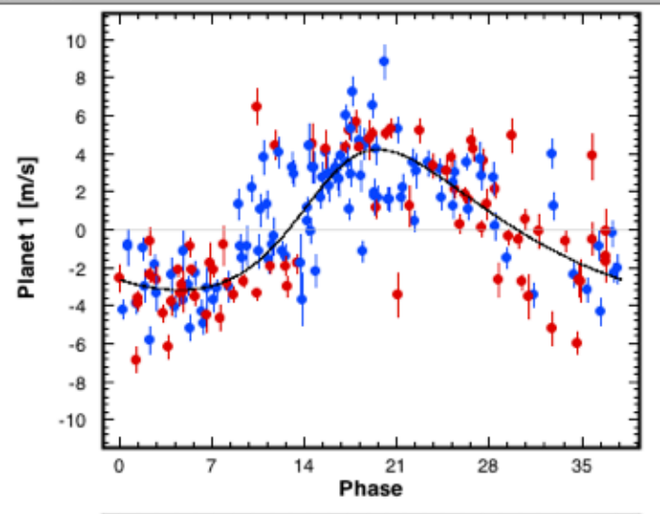
	Planet 1	Planet 2	Planet 3
Period [d]	38.03334	4.21492	121.55729
Mass [Mj]	0.05754	0.01702	0.06255
Mean anomaly [deg]	206.54303	166.00596	40.56638
Eccentricity	0.21819	0.16019	0.34807
Long. Peri [deg]	311.01757	100.3883	311.42328
Inclination [deg]	90	90	90
Node [deg]	0	0	0
Semi-maj. axis [AU]	0.21759	0.0502	0.47212
Semiamplitude [m/s]	3.68719	2.24459	2.83278
Radius [Rj]	0	0	0

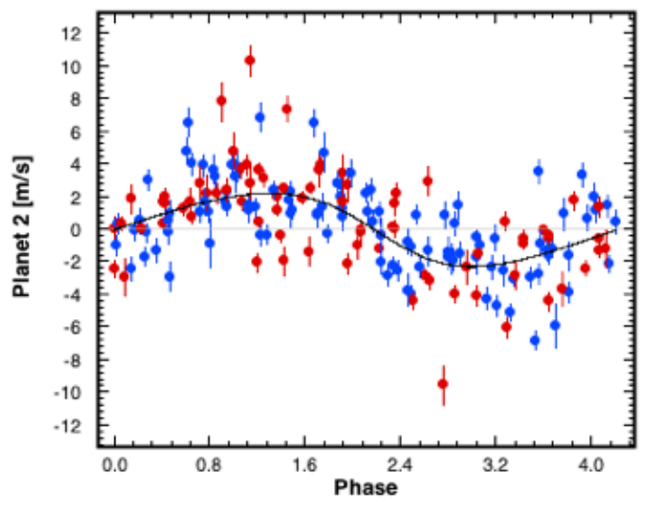
DATA PARAMETERS

HD115617_... | HD115617_...

0.2344 | 0.9892

RV SIGNAL BY PLANET

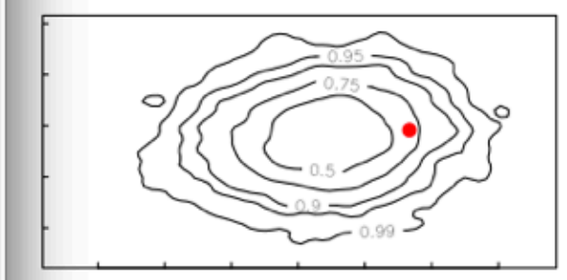




Editor | Pars

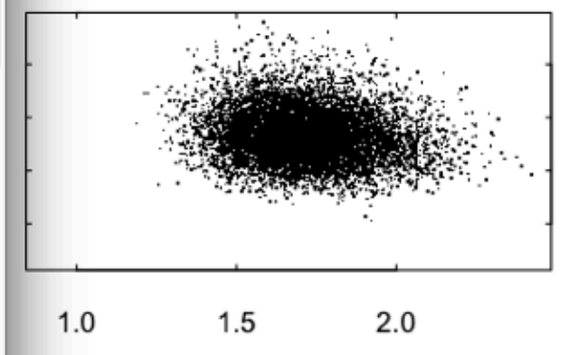
Quartz 2

Period of planet 1 vs Period of planet 2



Quartz 3 [*]

Jitter, AAT vs Jitter, KECK



Model statistics
(Chi², log likelihood, etc.)

Command line

Plots
(interactively updated)

Systemic

Ready.

Minimize

KERNELS

k

PROPERTIES

# Planets	3
Star mass [M_o]	0.95
Epoch [JD]	2453369.16
Chi^2_{RV}	1759.78016
RMS_{RV} [m/s]	2.16691
Jitter [m/s]	2.04813
# RVs	183
Chi^2_{TT}	0
RMS_{TT} [d]	--
# TTs	0
Chi^2	1759.78016
$Chi^2_{reduced}$	10.60109
-Log likelihood	797.00126
KS p-value	3.6982e-13
# Data	183
Norm. par.	17
Int. Method	KEPLER
Min. Method	SIMPLEX
Element type	ASTROCENTR

1 AU

```
=> kremove.data(k, 'all'); k$mstar <- 0.950000 ; k$epoch
<- NaN; kadd.data(k, c("./datafiles/HD115617_AAT.vels",
"./datafiles/HD115617_KECK.vels"));
=> kminimize(k)
[1] 10.60109
=> b_1 <- kmcmc(k, chains=2, R.stop=1.1, skip.
first=1000, discard=k$nrpars*10, print=TRUE)
Final length: 5236, final R_max = 1.065146e+00, final
Rsingle_max = 1.488546e-01
# mcmc, chains = 2, R.stop = 1.100000e+00, start =
perturb, noise=TRUE, skip = 1000, discard = 170, tot.
length = 10472
```

```
plot(b_1, type='scatter', px=list('par','data.noise1'), p
y=list('par','data.noise2'), dev.factor=5, pch='.', xlab=
'Jitter, AAT', ylab='Jitter, KECK')
```

ORBITAL ELEMENTS

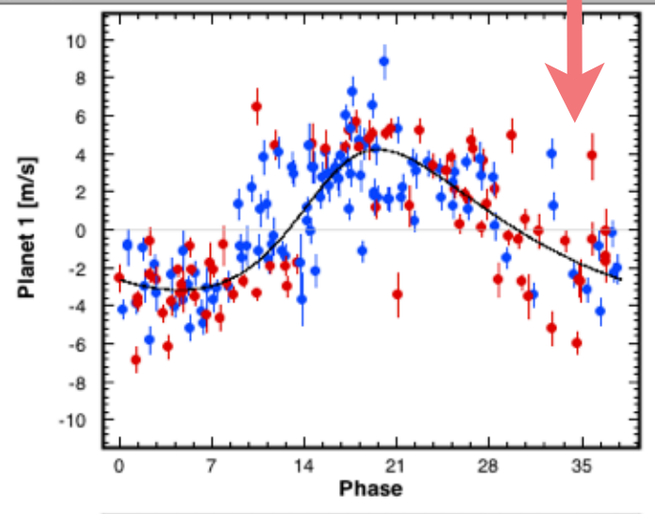
	Planet 1	Planet 2	Planet 3
Period [d]	38.03334	4.21492	121.55729
Mass [Mj]	0.05754	0.01702	0.06255
Mean anomaly [deg]	206.54303	166.00596	40.56638
Eccentricity	0.21819	0.16019	0.34807
Long. Peri [deg]	311.01757	100.3883	311.42328
Inclination [deg]	90	90	90
Node [deg]	0	0	0
Semi-maj. axis [AU]	0.21759	0.0502	0.47212
Semiamplitude [m/s]	3.68719	2.24459	2.83278
Radius [Rj]	0	0	0

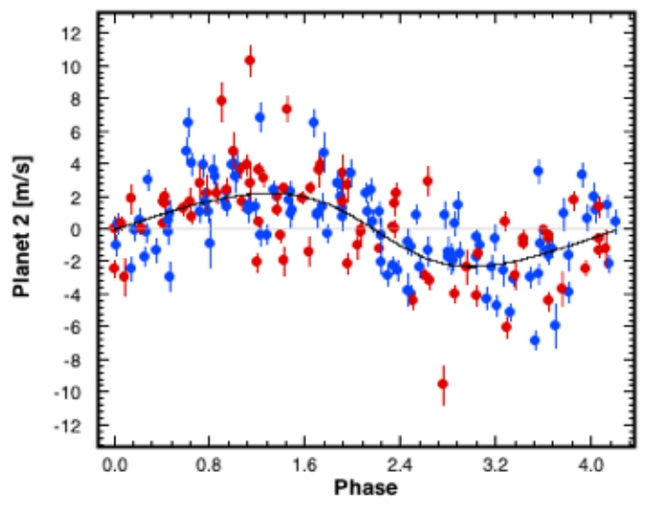
DATA PARAMETERS

HD115617_... | HD115617_...

0.2344 | 0.9892

RV SIGNAL BY PLANET

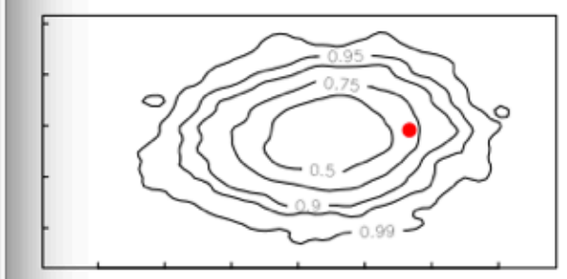




Editor | Pars

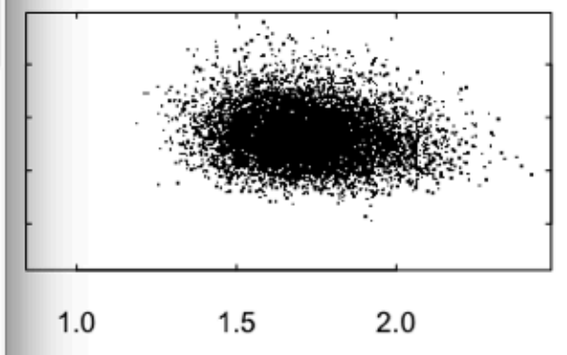
Quartz 2

Period of planet 1 vs Period of planet 2



Quartz 3 [*]

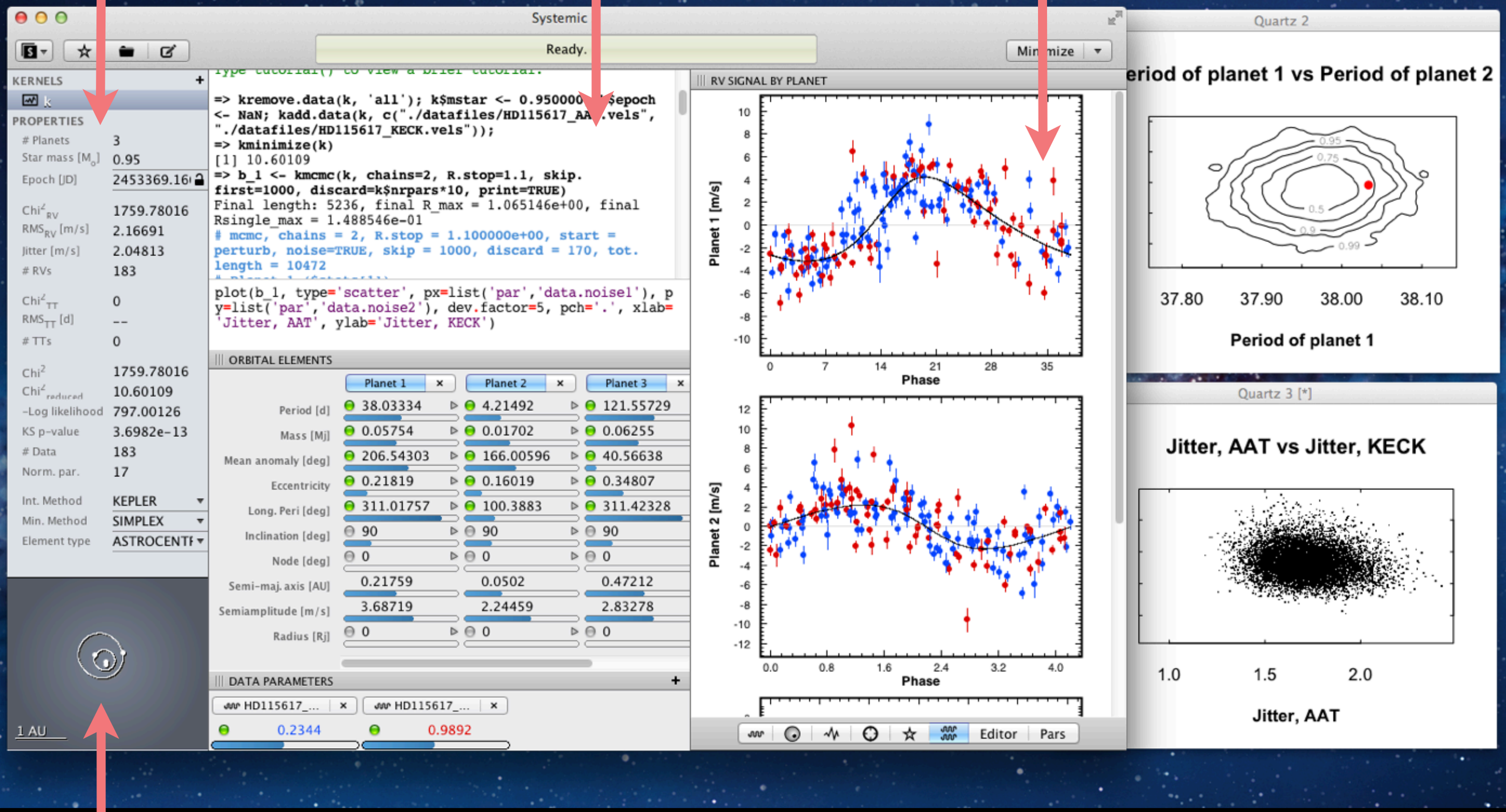
Jitter, AAT vs Jitter, KECK



Model statistics
(Chi², log likelihood, etc.)

Command line

Plots
(interactively updated)

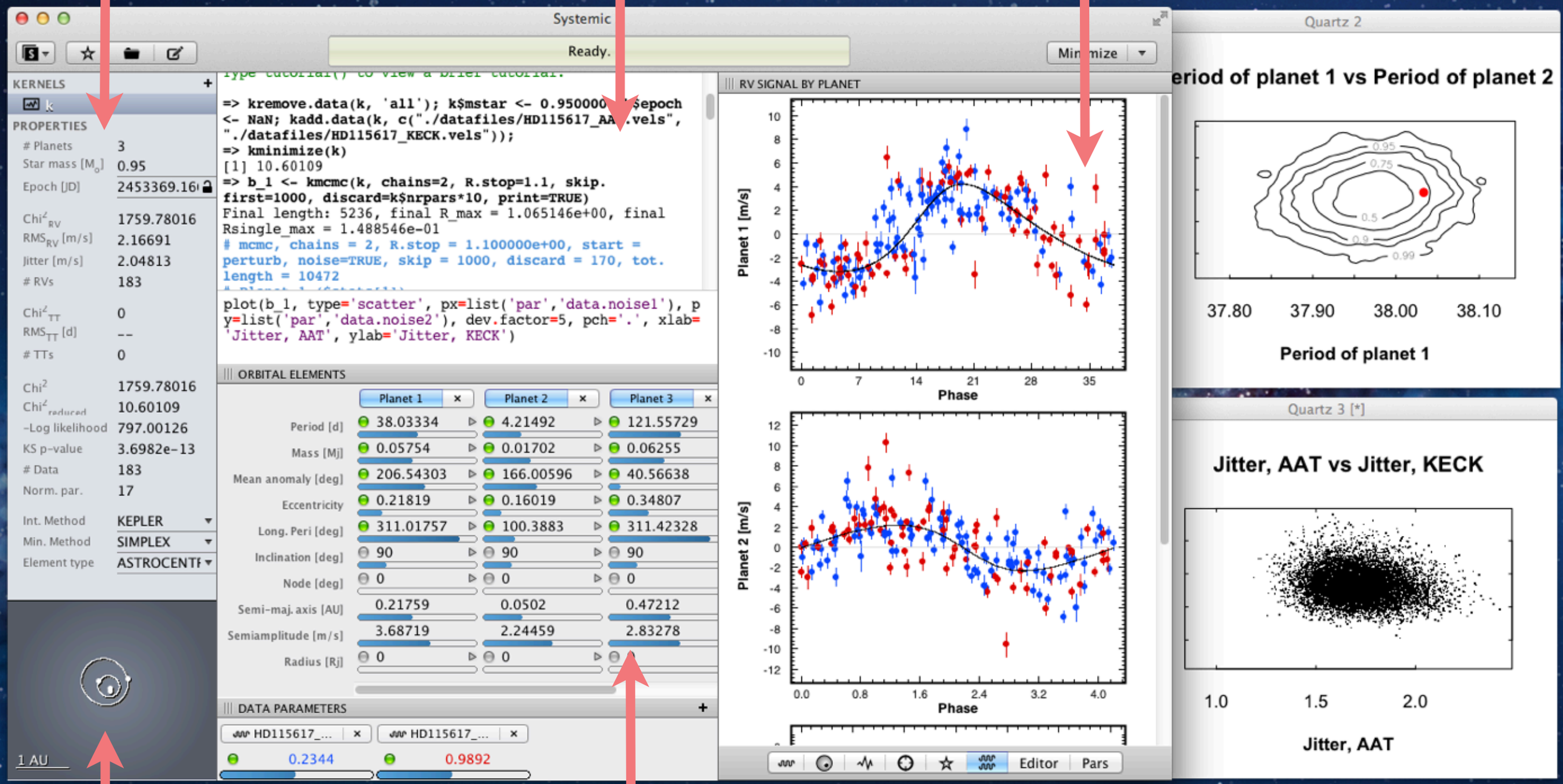


Orbital plot

Model statistics
(Chi², log likelihood, etc.)

Command line

Plots
(interactively updated)



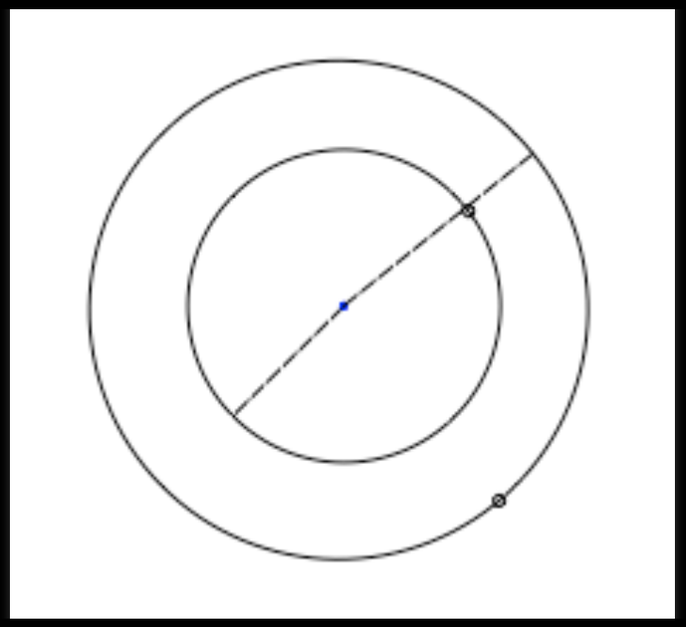
Orbital plot

Model parameters

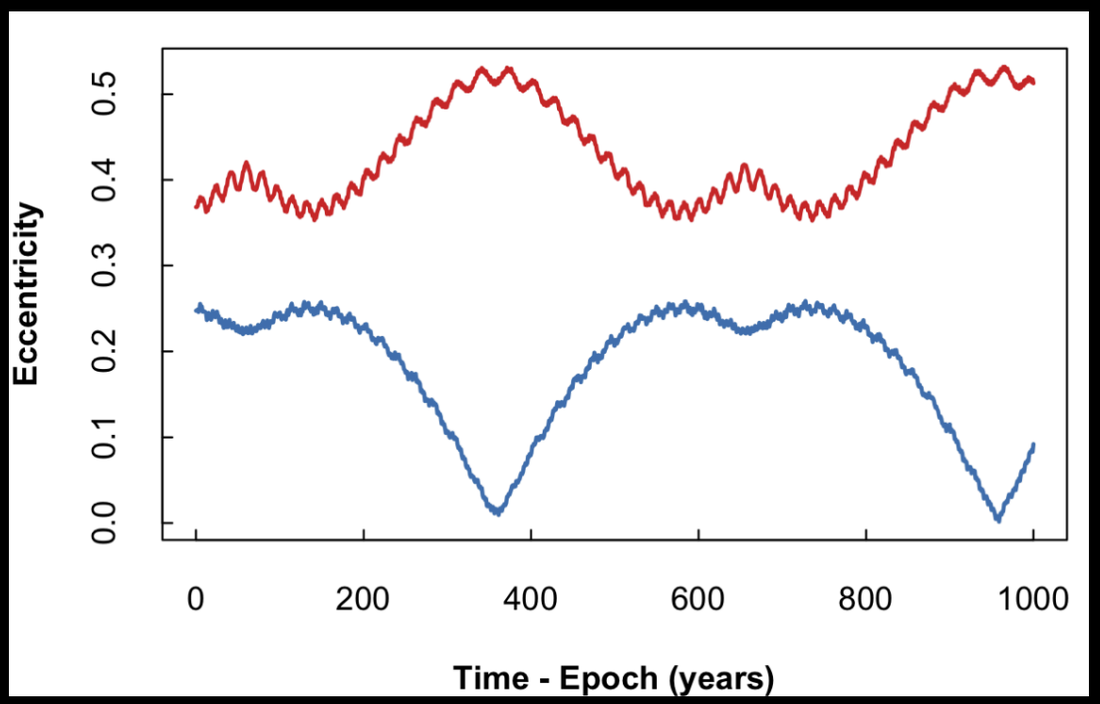
Dynamical fitting

Models can optionally include gravitational interactions between bodies:

(1) Fit strongly interacting/resonant systems (e.g. GJ876, HD128311, etc.)

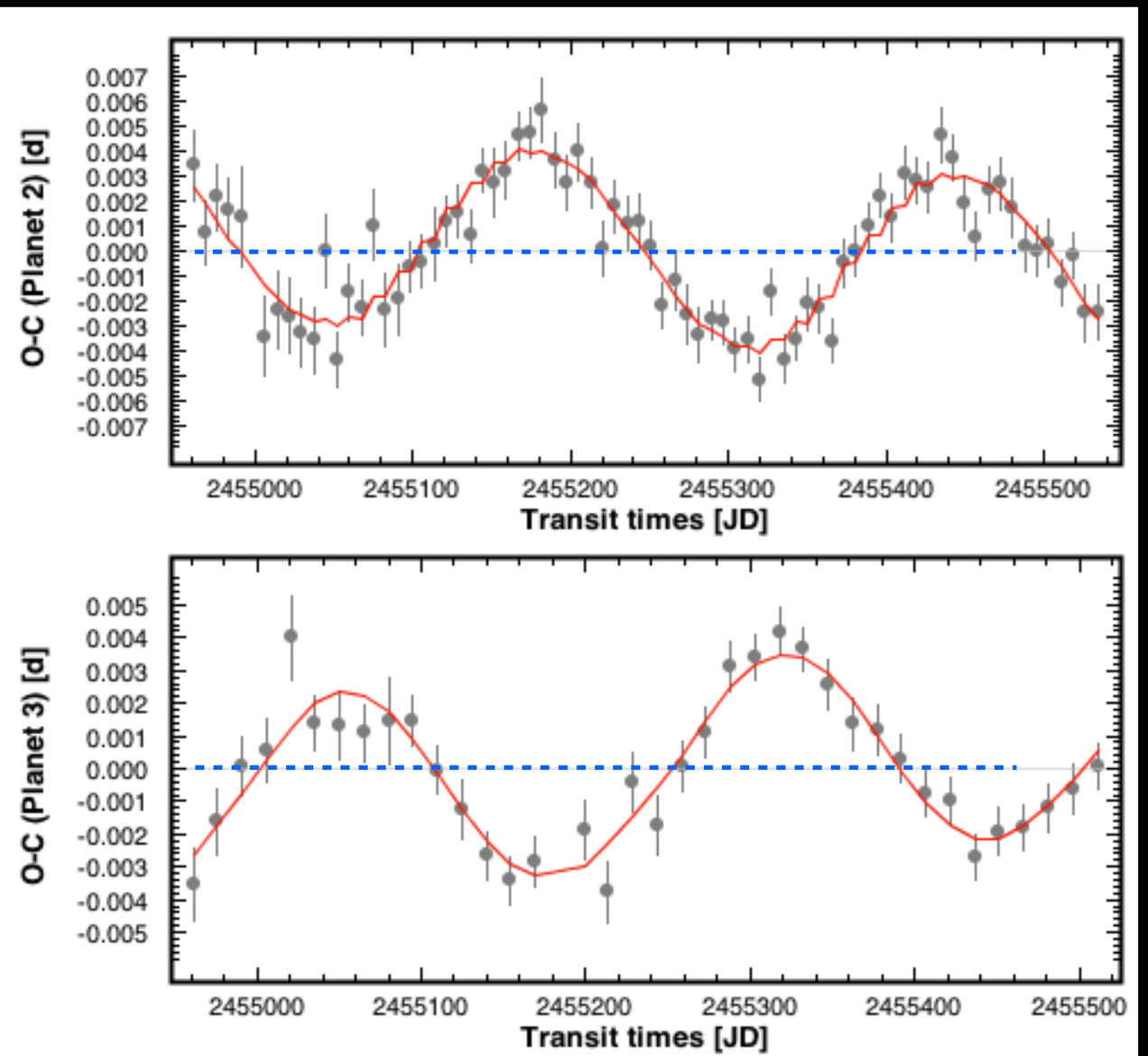


(2) Check for the long-term stability of a planetary system and create stability maps

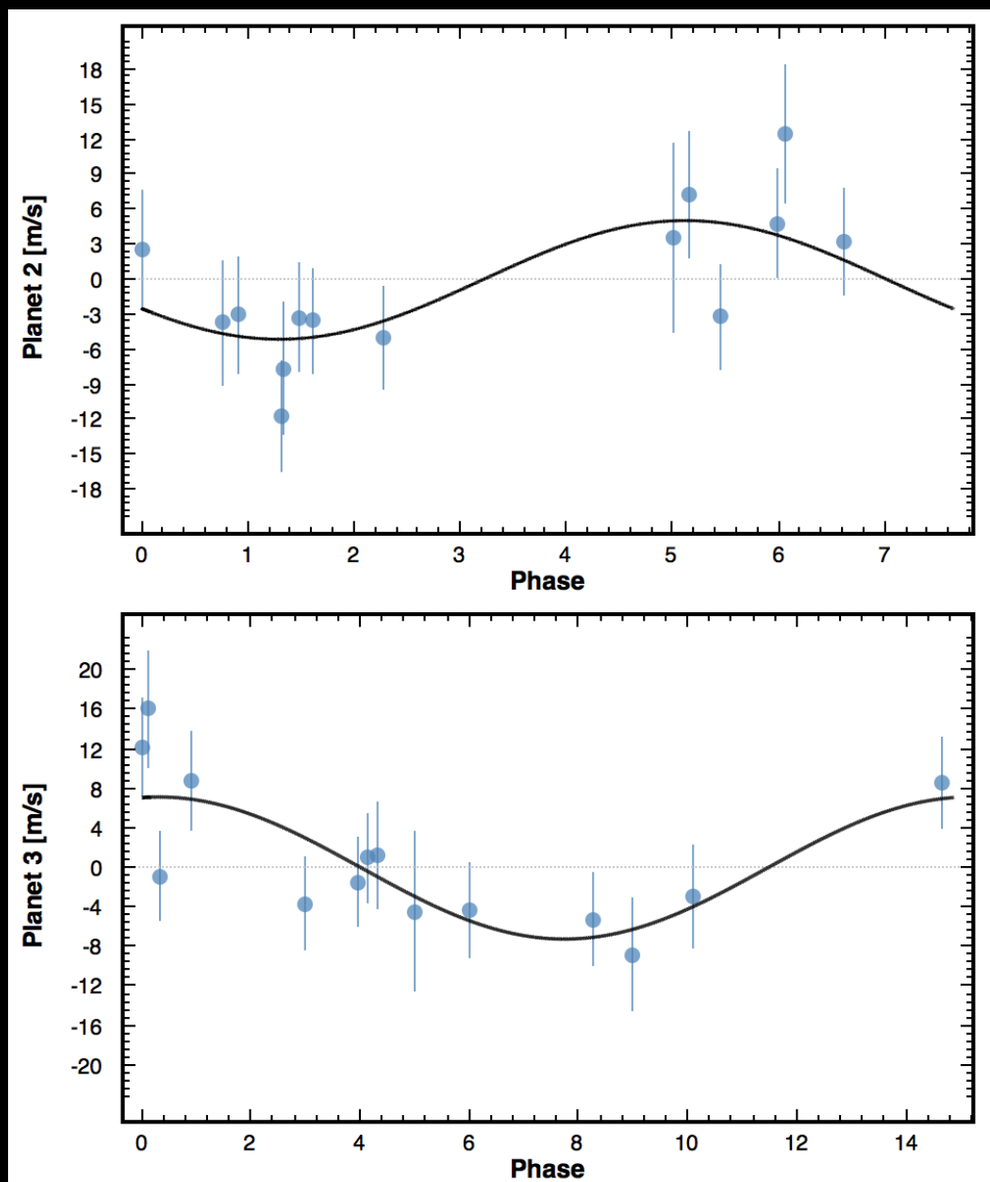


Dynamical fitting

(3) Fit transit timing datasets combined with radial velocity datasets and take advantage of transit timing variations to constrain orbital elements.



+



- Lomb-Scargle and bootstrapped periodogram
- Keplerian and self-consistent fitting
- Long-term integration using SWIFT
- Optimization using Simplex, Levenberg-Marquardt, Simulated Annealing or Differential Evolution
- Error estimation using Markov-Chain Monte Carlo or bootstrap
- Model cross-validation using jack-knife
- Completely customizable models (e.g., add new parameters to the model)
- Algorithms are automatically parallelized to run across multiple cores; some algorithms can run across computing clusters
- And more!

Demo

It'll be quick, I promise!

Systemic is also an R package.

This means that you can write full-fledged scripts to analyze your data, and interface with literally thousands of sophisticated statistical packages.

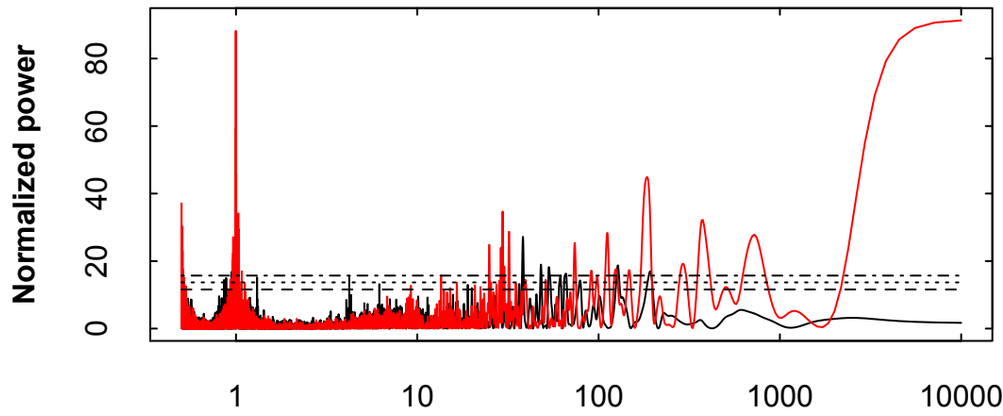
Computations are parallelized and can be run across clusters.

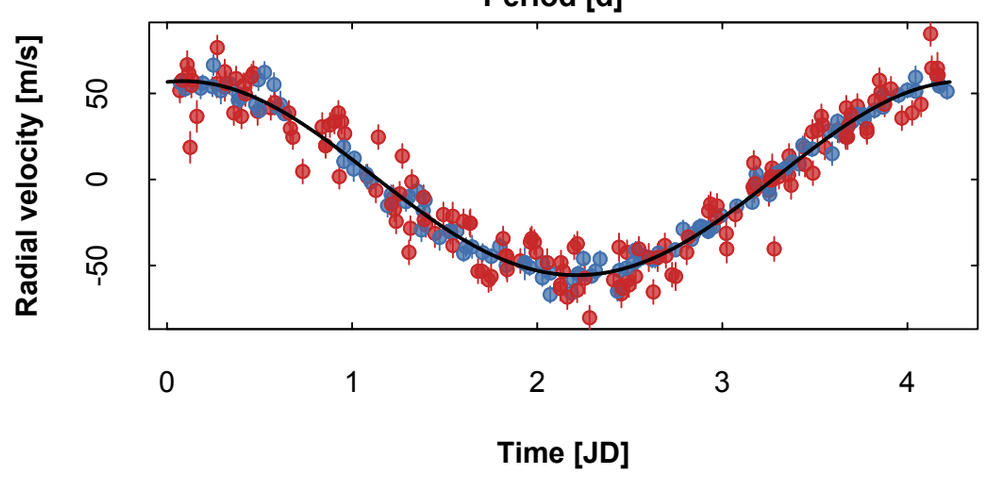
Load data, add a planet and run a Markov-Chain Monte Carlo algorithm.

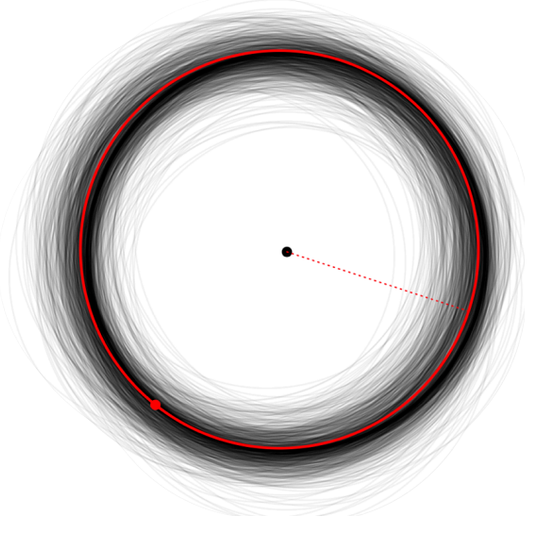
```
# Creates a new model object
k <- knew()
# Load new data
kadd.data(k, "1pl.vels")
# Calculate the power spectrum of the data
p <- kperiodogram(k)

# Add a planet at the period corresponding
# to the highest peak
kadd.planet(k, c(period = p[1, 'period']))
kminimize(k)
plot(k)

# Run a Markov-Chain Monte Carlo analysis
# (with default parameters)
kmcmc(k, chains=5)
```







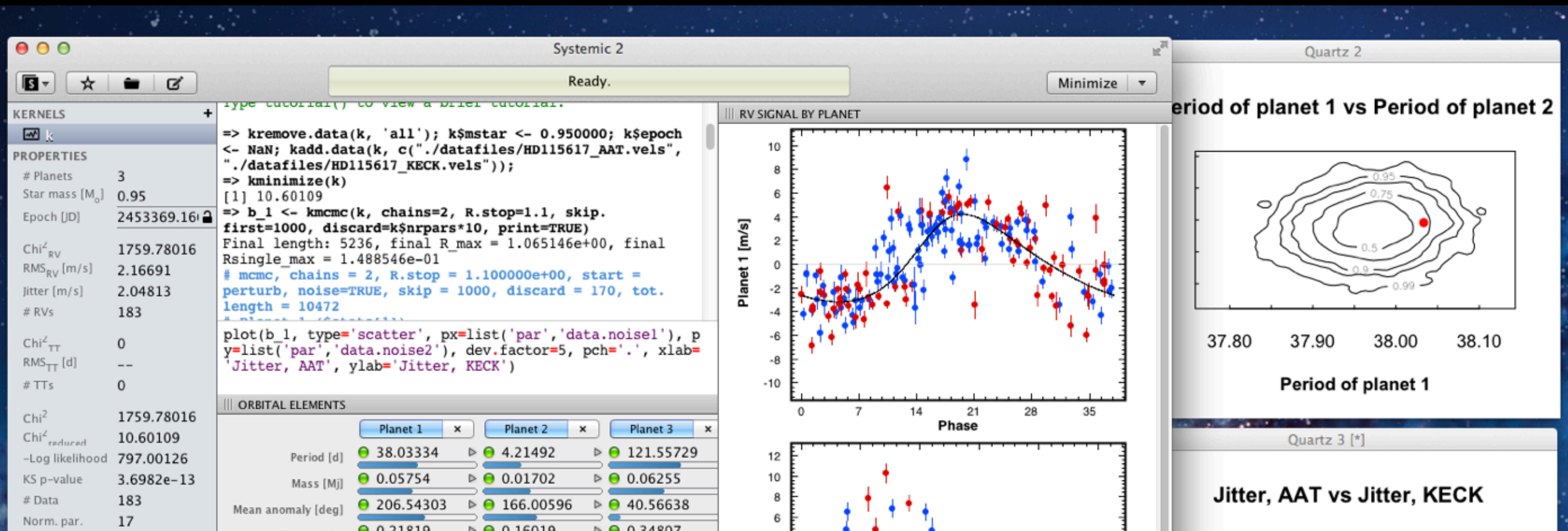
Systemic 2

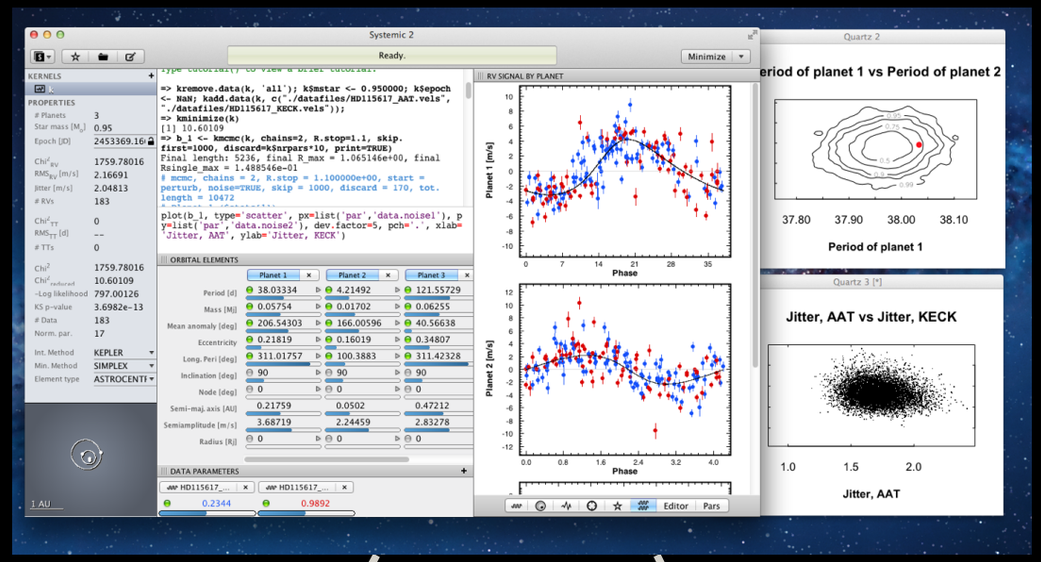
This package and its source code is free and available on GitHub: anyone can download it and modify it freely...

http://github.com/stefano-meschiari/Systemic2

Teaching & Outreach

One could use the "full" Systemic to let students analyze exoplanetary data, but its interface can be overwhelming...





(Python)

R + packages

C library (libsystemic)

Gnu Scientific Library (GSL)

ODEX.f

SWIFT.f

Mercury.f

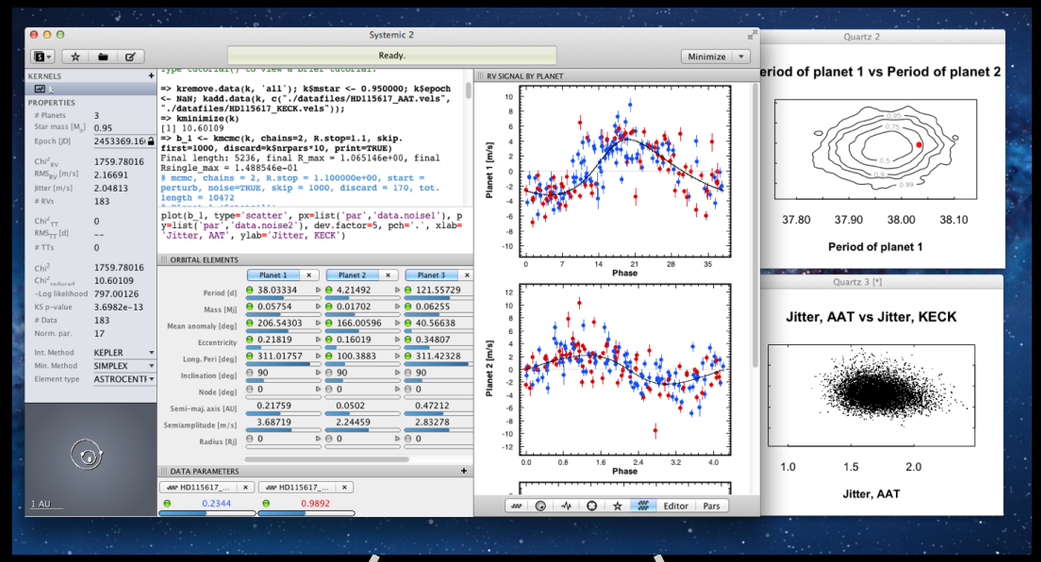
User interface (Java)

High-level language (R and packages)

Core code:

C library

+ some Fortran code



(Python)

R + packages

C library (libsystemic)

Gnu Scientific Library (GSL)

ODEX.f

SWIFT.f

Mercury.f

User interface (Java)

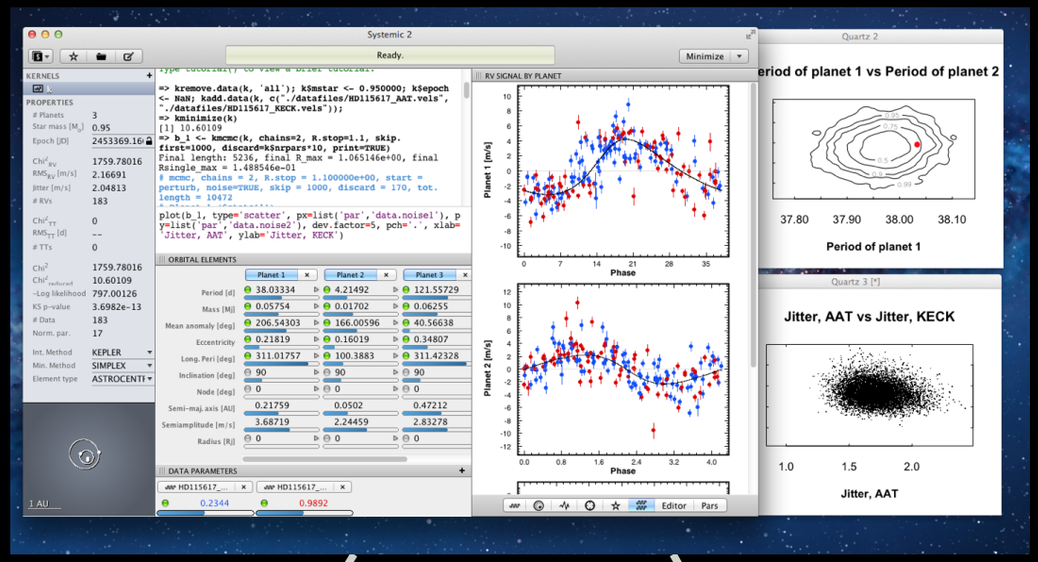
High-level language (R and packages)

Core code:

C library

+ some Fortran code

Best installation experience is no installation.



(Python)

R + packages

C library (libsystemic)

Gnu Scientific Library (GSL)

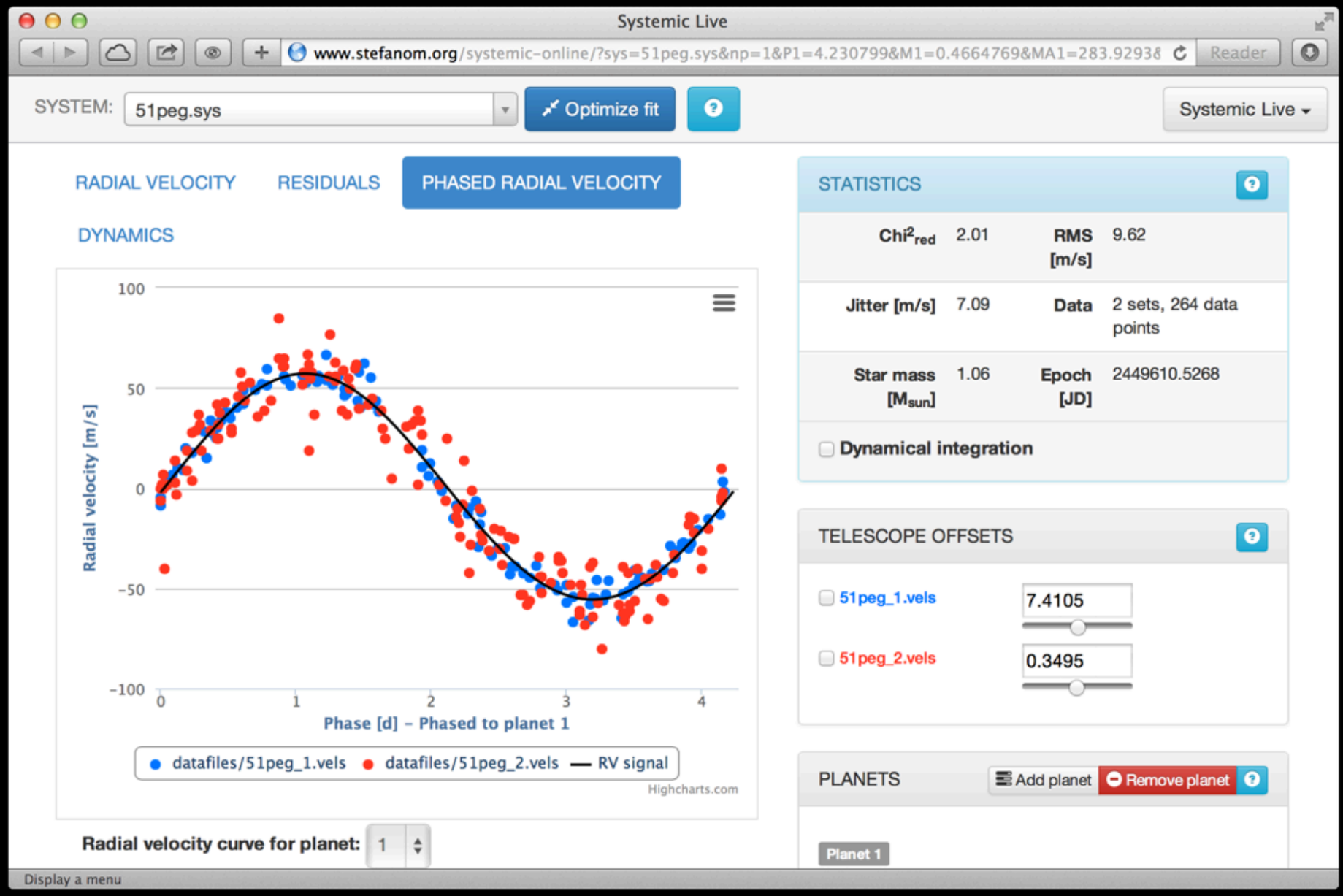
ODEX.f

SWIFT.f

Mercury.f

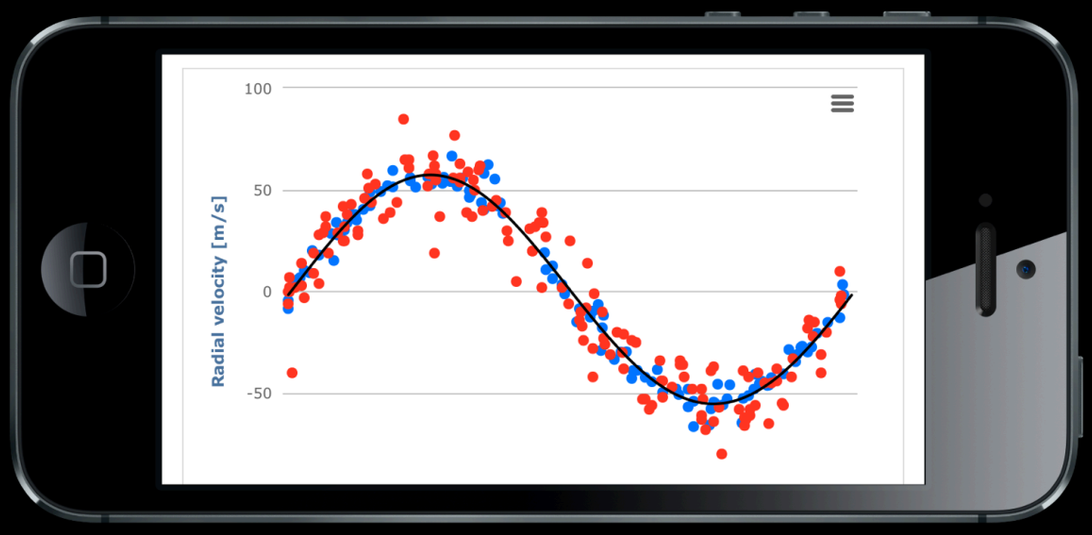
The Web

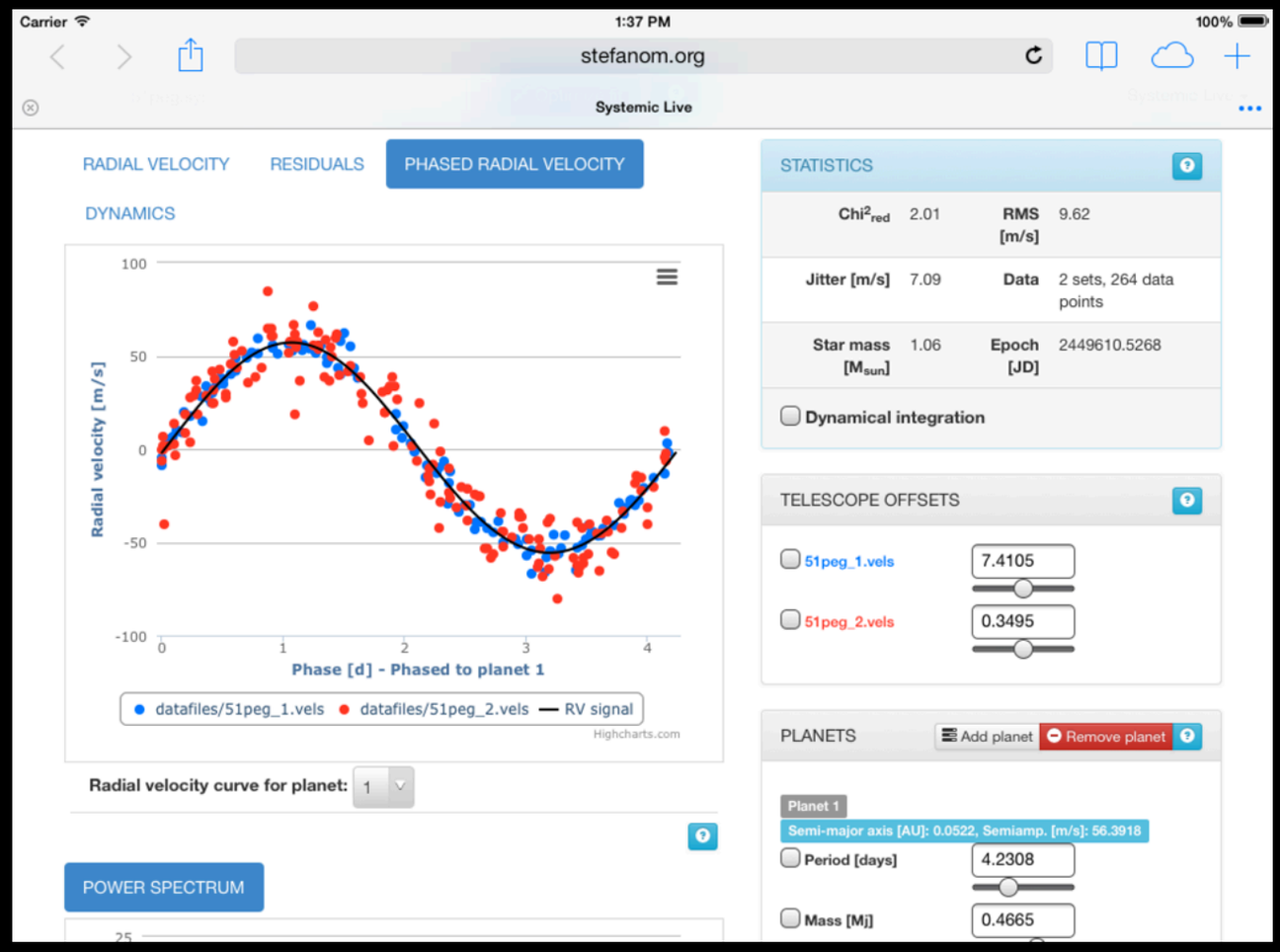
Systemic Live



A simplified web app for modelling exoplanetary data, at the just the right level for high school & undergraduate classes.

Systemic Live





Works on smartphones and tablets, too.

Demo

http://www.stefanom.org/systemic-online/?sys=51peg.sys

Systemic Live

It's easy to share a planetary model using just the current web address, like so:

http://www.stefanom.org/systemic-online/?sys=51peg.sys&np=1&P1=4.230799&M1=0.4664769&MA1=283.9293&E1=0.0140892&L1=344.9533&o0=7.410511&o1=0.3495096&im=0

"The online Systemic Console is a real gift to the community. The online console distills years of work to optimize the modeling real radial velocity data. Students can run bootstrap Monte Carlo codes to determine measurement errors and numerical integrations to determine the dynamical stability of multi-planet systems. I use this site to train both undergraduate and graduate students – they love the power of this program."

Debra Fischer, Yale University

"Systemic is simple enough to use that it can provide a hand-on 'virtual lab' for a large general education class, [...] students can get a taste of the scientific process even before they learn to program" –

Eric Ford, Penn State

"I have used Systemic for several years in my class for advanced undergraduate physics majors. The students favorite problem set uses Systemic to explore real radial velocity data sets and compare their solutions to orbital parameters for published systems. Systemic is extremely sophisticated, but easy to use, so it allows students to get a feeling for the tools used in real exoplanet research."

Jonathan Fortney, UC Santa Cruz

Tutorials & labs

You can find tutorials and labs on the webpage:

http://www.stefanom.org/systemic-live/

- **“51 Pegged: Rediscovering the First Exoplanet”**
- **“A Fish in a Barrel — HD 4208b”**
- **“The Ups and Downs of Ups And”**
- **...and others**

I’m also working to connect existing exoplanet “databases” (exoplanet.eu, exoplanet.org, etc.) with Systemic, so that with a click you can access and analyze the data associated with a system.

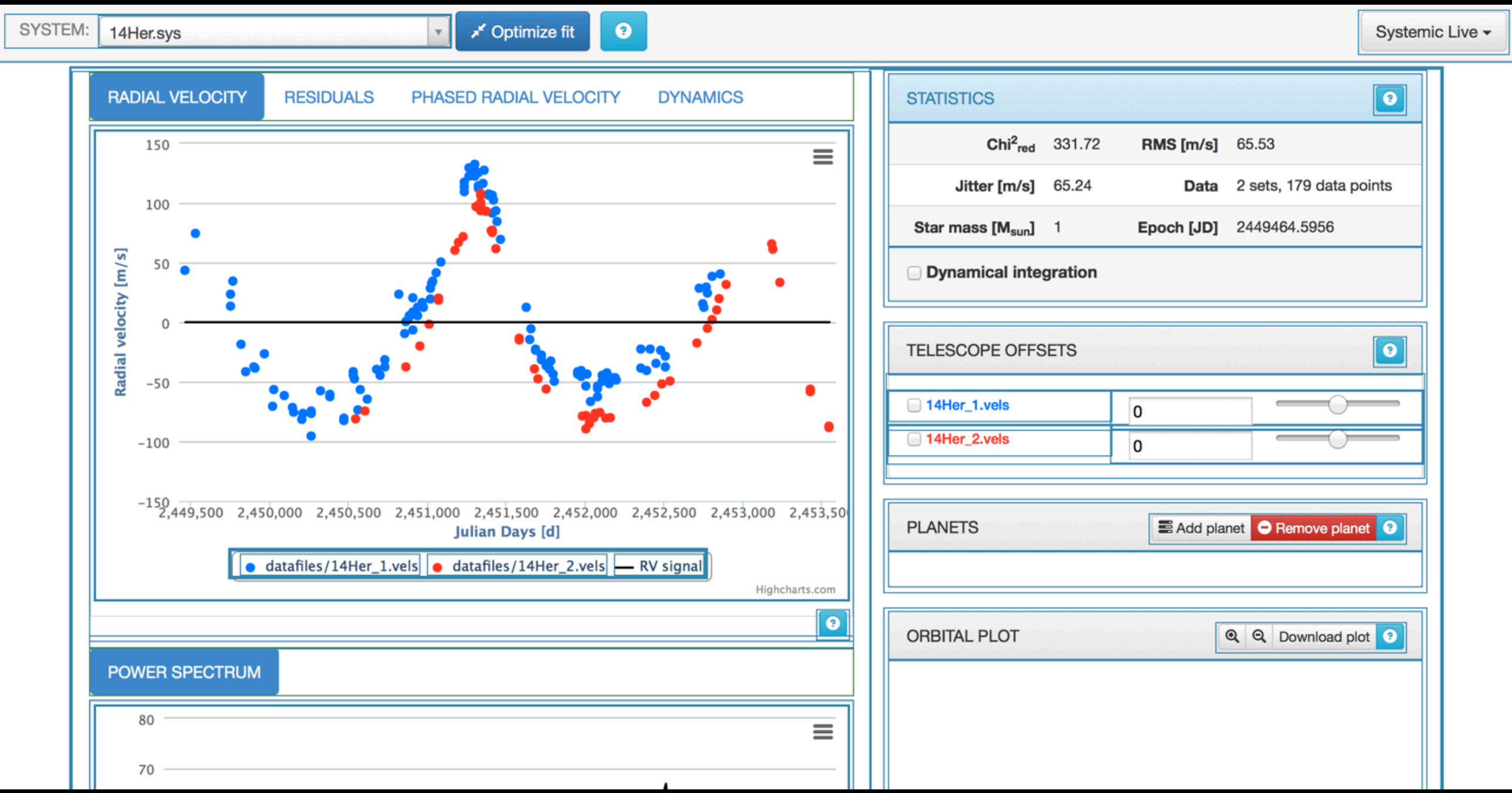
HOWTO

My biased recommendations on getting it done, as a busy astronomer who's eager to learn new valuable skills and do some outreach in the process.

https://github.com/stefano-meschiari/Notes

Making online web apps

Presentational part: Structure, Layout and Appearance



Making online web apps

Presentational part:

- **Learn the *very* basics of HTML5 and CSS.**
  You should start grokking this stuff anyway if you're making your own webpage (see Chalence's talk). HTML defines the structure of the page, CSS its appearance (roughly).

  Recommendation: the **Mozilla Developer Network (MDN)** is really great place to start. Unfortunately, a lot of bad/misleading resources bubble up Google searches (*cough* W3Schools *cough*), so beware.

Making online web apps

Presentational part:

- **Make a framework do most of the work.**

  A framework takes care of a lot of things that are objectively complicated even in modern browsers, like complex layouts, components, smoothing over browser differences, and more. They are usually pretty quick to learn and give your projects a professional look.

  Recommendation: **Bootstrap** or **UIKit.**

Making online web apps

Interactivity/computations

systemic.js (/Users/sm52286/Projects/SystemicLive/js/systemic.js)

```
944
945     var optimize = function() {
946         var active = false;
947         for (var i = 1; i <= K_getNplanets(k); i++) {
948             for (var j = PER; j <= LOP; j++) {
                    if ($("#elementSel_" + i + "_" + j).is(":chec
950                     K_setElementFlag(k, i, j, MINIMIZE+ACTIVE
951                     active = true;
952                 }
953                 else
954                     K_setElementFlag(k, i, j, ACTIVE);
955             }
956         }
957         for (var i = 0; i < MAX_SETS; i++)
958             if ($("#offsetSel_" + i).is(":checked")) {
959                 K_setParFlag(k, i, MINIMIZE+ACTIVE);
960                 active = true;
961             }
962         else
963             K_setParFlag(k, i, ACTIVE);
964
965         if (!active) {
966             alert("You need to have at least one parameter se
    checkboxes next to each parameter.)");
967             return;
968         }
969
970
971         var chi2 = K_getChi2(k);
```

Making online web apps

Interactivity/computations:

- **Learn JavaScript.**

  JavaScript superficially looks very similar to C or Java.

```
function square(x) {
    return x * x;
}

var x = 2;
alert(“The square of” + x + “ is “ + sqr(x));
```

In reality, very different conceptually and functionally.

Making online web apps

Interactivity/computations:

- **Learn JavaScript.**

  In many ways, it's an evil, evil language. At the same time, *only* language allowed on browsers (no C/Fortran/IDL/Python/Perl/Ruby/anything), so it's an incredibly valuable skill.

  This is probably the hardest component to learn correctly.

Making online web apps

Interactivity/computations:

Recommendations:

Making online web apps

Interactivity/computations:

Recommendations:

- **JavaScript: the Good Parts** is brief, clear and warns you about warts & pitfalls of the language.

Making online web apps

Interactivity/computations:

Recommendations:

- **JavaScript: the Good Parts** is brief, clear and warns you about warts & pitfalls of the language.
- **Mozilla Developer Network** is a useful reference for JavaScript as well.

Making online web apps

Interactivity/computations:

Recommendations:

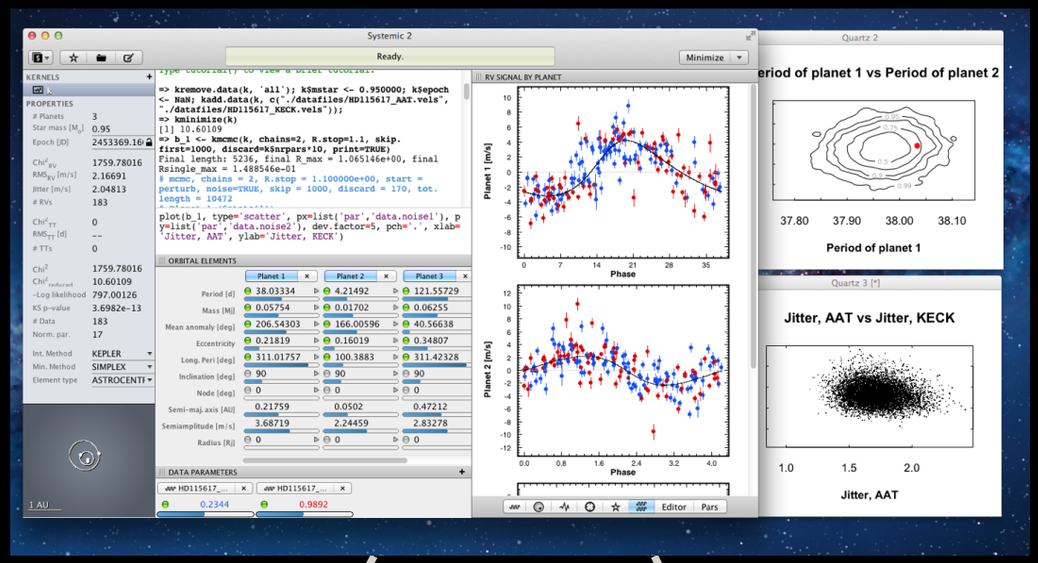
- **JavaScript: the Good Parts** is brief, clear and warns you about warts & pitfalls of the language.
- **Mozilla Developer Network** is a useful reference for JavaScript as well.
- Again, frameworks and libraries can lessen the pain and make you more productive. **JQuery** and **Underscore.js** have taken a lot of the friction out of interacting with the webpage elements for me.

Making online web apps

Interactivity/computations:

Recommendations:

- **JavaScript: the Good Parts** is brief, clear and warns you about warts & pitfalls of the language.
- **Mozilla Developer Network** is a useful reference for JavaScript as well.
- Again, frameworks and libraries can lessen the pain and make you more productive. **JQuery** and **Underscore.js** have taken a lot of the friction out of interacting with the webpage elements for me.
- **Node.js** is a way to run and test your JavaScript *outside* your browser.



(Python)

R + packages

C library (libsystemic)

Gnu Scientific Library (GSL)

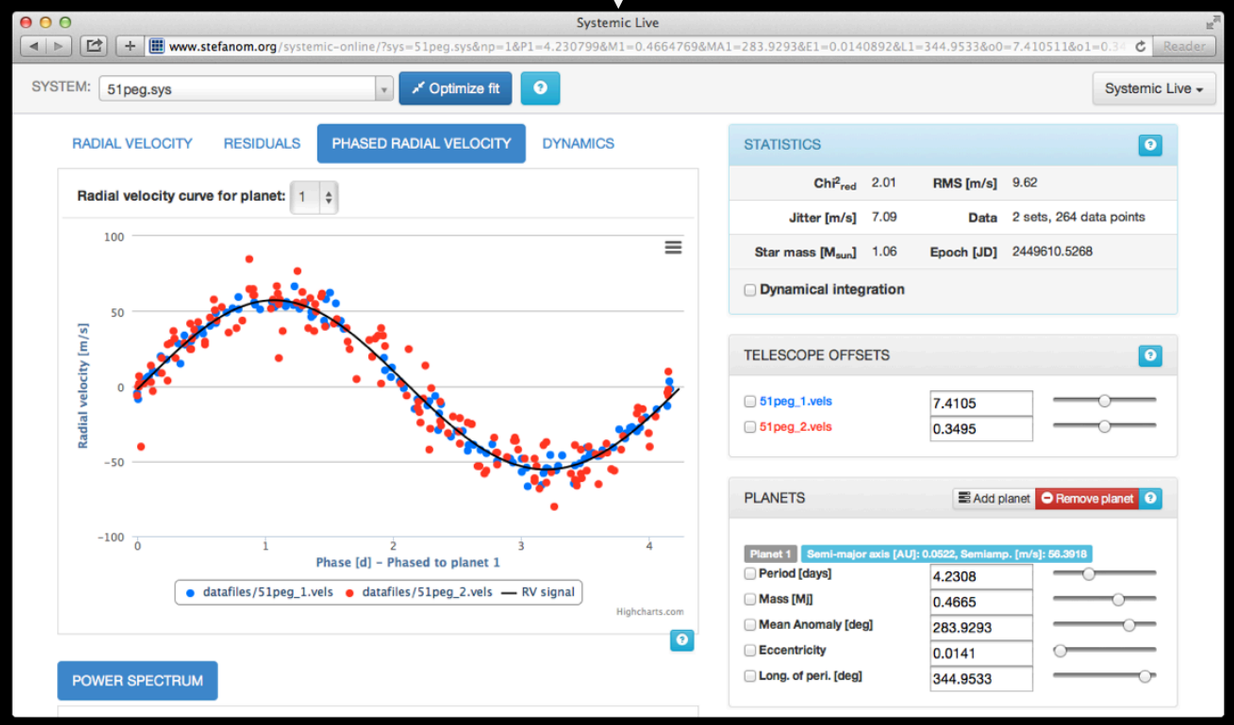
ODEX.f

Mercury.f

SWIFT.f

emscripten

converts C code into Javascript



Making online web apps

Interactivity/computations:

Recommendations, part deux:

- **Emscripten** is a fabulous way to translate complex but trusty C (Fortran) code into Javascript code. You literally could just change this command:

```
make MyProject
```

into this:

```
emmake MyProject
```

and you would get Javascript (instead of machine code) as the output.

Systemic Live

Science

Teaching & outreach

Any improvement to the scientific software goes directly into the teaching & outreach code.

Making online web apps

Plotting:

- **Lots of very different approaches.**

  I tend to prefer more limited libraries that are "turn-key", i.e. do not require to learn a whole different paradigm just for plotting some data. E.g. just specify that you want a scatterplot, provide the data and go.

  Recommendation: **Highcharts** **(free for edu)**

One more outreach thingy

How do you reach people that are very enthusiastic about exoplanets, but don’t have the technical skills, or patience, or interest in looking at real data?

How do you reach people that are very enthusiastic about exoplanets, but don't have the technical skills, or patience, or interest in looking at real data?

We have all this really good code and a way to port it on the Web, so what do you do?

How do you reach people that are very enthusiastic about exoplanets, but don't have the technical skills, or patience, or interest in looking at real data?

We have all this really good code and a way to port it on the Web, so what do you do?

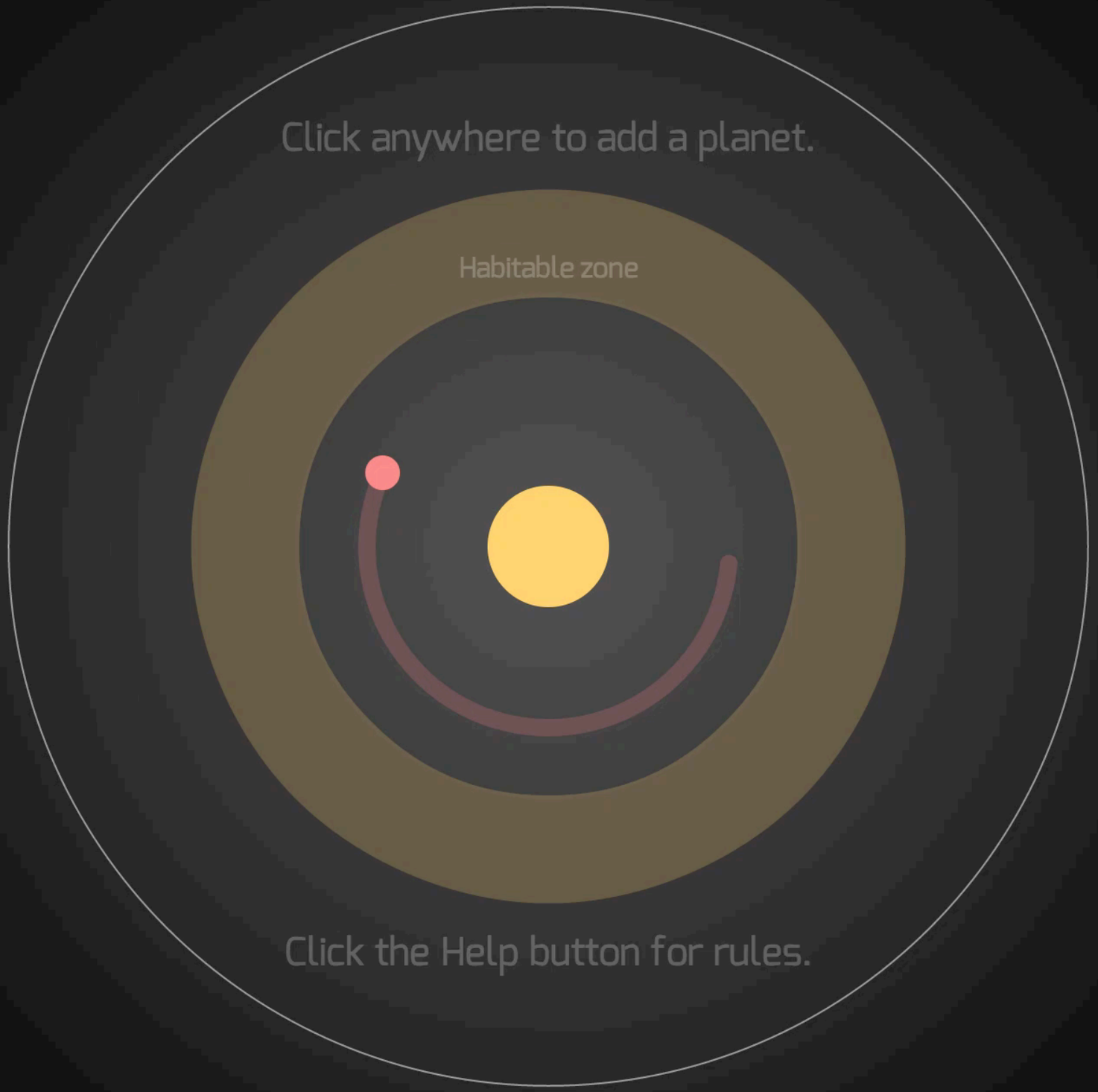
Make a game!

Super Planet Crush

Click on the type of body to add next:

- Earth 1x
- Super-Earth 5x
- Ice giant 15x
- Giant planet 300x
- Brown dwarf 5,000x
- Dwarf star 30,000x

Help | End Game



Years: 0.0/500
Points: 0
2 / 12 bodies
Crowdedness bonus: 1.0x
Habitability bonus: 1.0x
Central star: 1.0x
Speed: 3x

Planet 1 (1.00 M_{earth})

	Name	High score
1.	Ben	51,256,990
2.	Rachael	31,585,152
3.	Ben	28,164,173
4.	Augusto	23,419,920
5.	Ben	22,753,808
6.	Augusto	20,448,158
7.	Ben	17,863,445
8.	Mike P	17,768,522
9.	Mike P	17,675,722
10.	Rachael	17,439,645

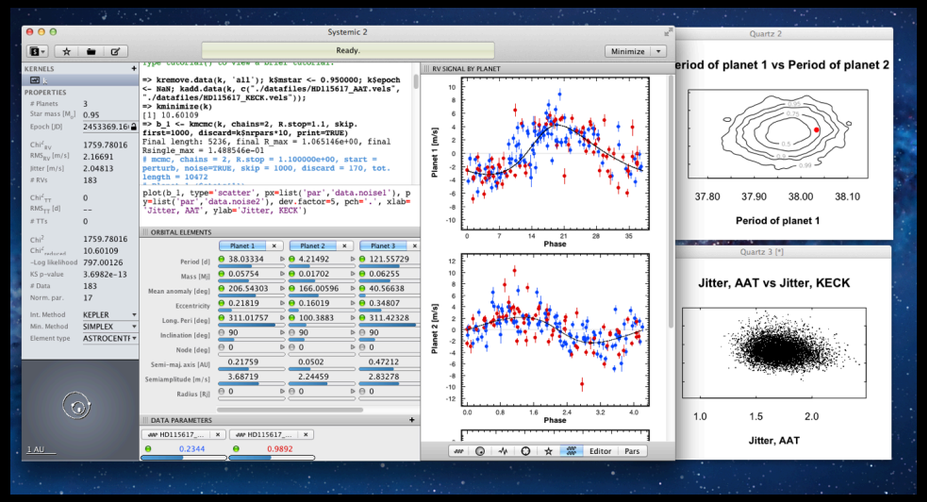
http://www.stefanom.org/spc

http://www.stefanom.org/spc

user: baesm
password: baesm

My next goal is for the game to be able to use interesting compact multi-planet systems (e.g. Kepler-11) as the starting templates, so the player can mercilessly destabilize them by adding planets.

Thank you!



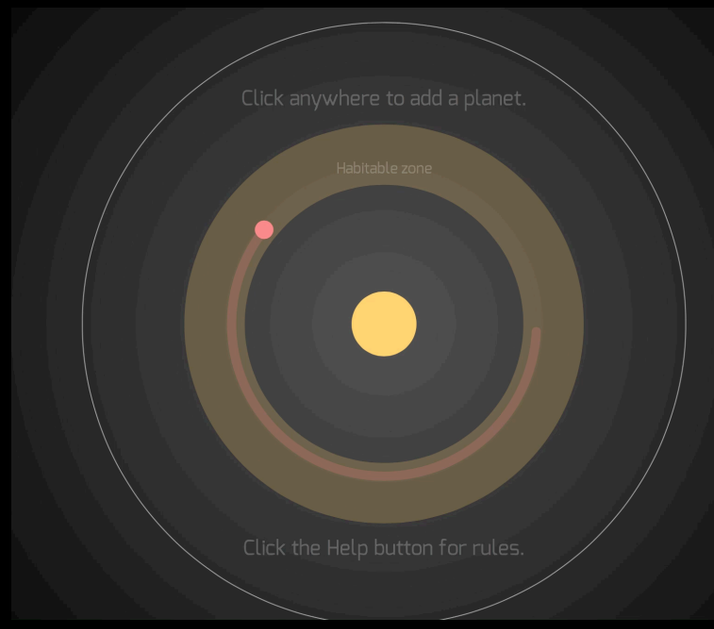
All things Systemic:

http://www.stefanom.org/systemic

Play with this game and beat your fellow astronomers:

http://www.stefanom.org/spc

user/password: baesm



Here is a list of all the tools I mentioned in this talk (with links):

https://github.com/stefano-meschiari/Notes