In [1]:

```
import numpy as np
import pandas as pd
from astropy.timeseries import LombScargle
import matplotlib.pyplot as plt
from matplotlib import rc
```

Q1

Solving Kepler's equation using Newton's Method:

In [2]:

```
def solve_kepler(e, M):
    """
    Solve Kepler's Equation M = E - e sinE
    """
    dEs = []
    E = M + 0.85 * e * np.sign(np.sin(M))
    dE = 1e3
    while abs(E - e * np.sin(E) - M) > 1e-10:
        dE = ((E - e * np.sin(E) - M)) / (1 - e * np.cos(E))
        E -= dE
    return E
```

Verify over $0 < e < 1$ and $0 < M < 2\pi$ that Kepler's equation holds down to an error of 10^{-10}

In [3]:

```
e = np.random.random(size=10000)
M = 2 * np.pi * np.random.random(size=10000)

for e, M in zip(e, M):
    E = solve_kepler(e, M)
    assert(M - (E - e * np.sin(E)) < 1e-10)
```

None of the assertions failed, so our solver seems to do the job!

Q2

Calculate radial velocity given six parameters: $(K_{star}, P, t_p, e, \omega, \gamma)$

In [4]:

```
def true_anomaly(e, M):
    E = solve_kepler(e, M)
    return 2 * np.arctan(np.sqrt((1 + e) / (1 - e)) * np.tan(E / 2))
```

In [5]:

```
def RV_model(t, K_star, P, tp, e, omega, gamma):
    # if t is an array like object
    if np.ndim(t)!=0:
        RVs = []
        for tt in t:
            M = 2 * np.pi / P * (tt - tp)
            f = true_anomaly(e, M)
            RVs.append(K_star * (np.cos(omega) * np.cos(f)
                                 - np.sin(omega) * np.sin(f)
                                 + e * np.cos(omega))
                      + gamma)
```

```
        return RVs
    # if t is a float
    M = 2 * np.pi / P * (t - tp)
    f = true_anomaly(e, M)
    return K_star * (np.cos(omega) * np.cos(f)
                     - np.sin(omega) * np.sin(f)
                     + e * np.cos(omega)) + gamma
```

Q3

To find the period of the mystery planet we will use a Lomb Scargle Periodogram. First, loading in the data:

In [6]:

```
# Read in data
df = pd.read_csv('mystery_planet01.txt', delim_whitespace=True, header=None, names=['t', '
df['t'] = df['t'] - df['t'].min() # begin time at first obs
df.head()
```

Out[6]:

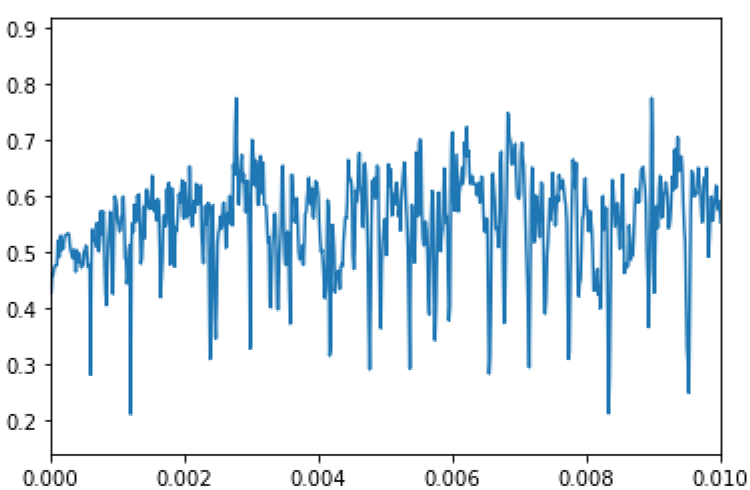
	t	rv	rv_err
0	0.000	-0.125	0.014
1	24.952	0.143	0.015
2	214.831	-0.110	0.012
3	215.842	-0.123	0.012
4	216.847	-0.129	0.012

Then some combination of arguments passed into astropy's LombScargle model seems to do the trick...

In [7]:

```
freq, power = LombScargle(df['t'], df['rv'], df['rv_err'], nterms=10).autopower(nyquist_fa

# plot frequency vs power
plt.plot(freq, power)
plt.xlim([0,.01])
```

Out[7]: (0.0, 0.01)



Getting the frequency (in $days^{-1}) with maximum power:

In [8]:

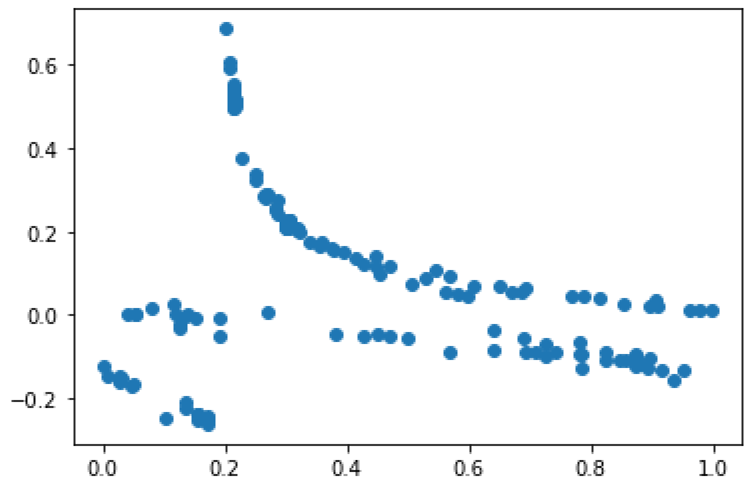
```
f = freq[np.argmax(power)]
f
```

Out[8]: 0.017948305912447 39

We plot the folded timeseries to verify if this is the correct frequency

In [9]:
```
P = 1/(f)
plt.scatter((df['t']%P)/P, df['rv'])
```

Out[9]: <matplotlib.collections.PathCollection at 0x7f80d0faa640>

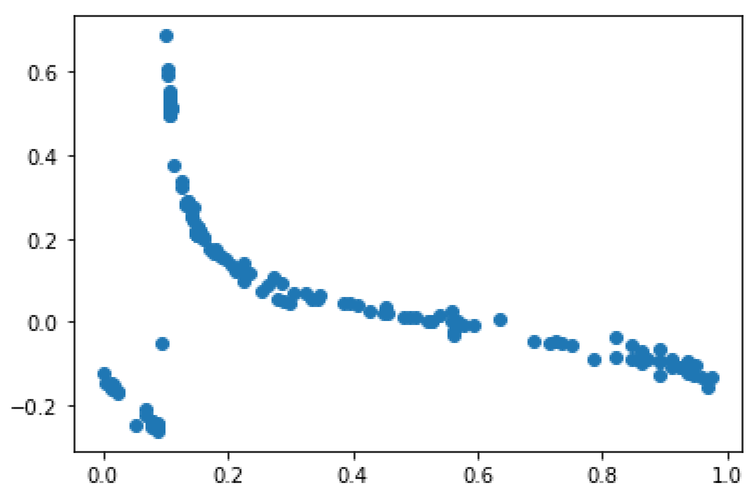


Hmmm close, but not good enough. Try the next harmonic

In [10]:
```
P = 2/(f)
```

In [11]:
```
plt.scatter((df['t']%P)/P, df['rv'])
```

Out[11]: <matplotlib.collections.PathCollection at 0x7f80c0859cd0>



In [12]:
```
P
```

Out[12]: 111.43112947573 36

Much better :) the period is approximately 111.4 days

Q4

Using an MCMC to fit the parameters: $(K_{star}, P, t_p, e, \omega, \gamma)$:

In [13]:
```python
import emcee
```

In [14]:
```python
def log_likelihood(theta, t, rv, rv_err):
    # return gaussian log likelihood of data given model params theta
    K_star, P, tp, e, omega, gamma = theta
    model = RV_model(t, K_star, P, tp, e, omega, gamma)
    sigma2 = rv_err ** 2
    return -0.5 * np.sum((rv - model) ** 2 / sigma2 + np.log(sigma2))

def log_prior(theta):
    # return -inf if params outside of priors, else return 0
    K_star, P, tp, e, omega, gamma = theta
    if .2 < K_star < .7 and 110 < P < 113 and 0 < tp < 113 and 0 < e < 1 and 0 < omega < 6
        return 0.0
    return -np.inf

def log_probability(theta, t, rv, rv_err):
    # sum of logL and log Priors
    lp = log_prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    return lp + log_likelihood(theta, t, rv, rv_err)
```

Using chi-by-eye, I determined an initial guess of parameters

In [15]:
```python
# K_star, P, tp, e, omega, gamma
init = [  0.5, 111.5,   10,  .95, 5.2,   0]
```

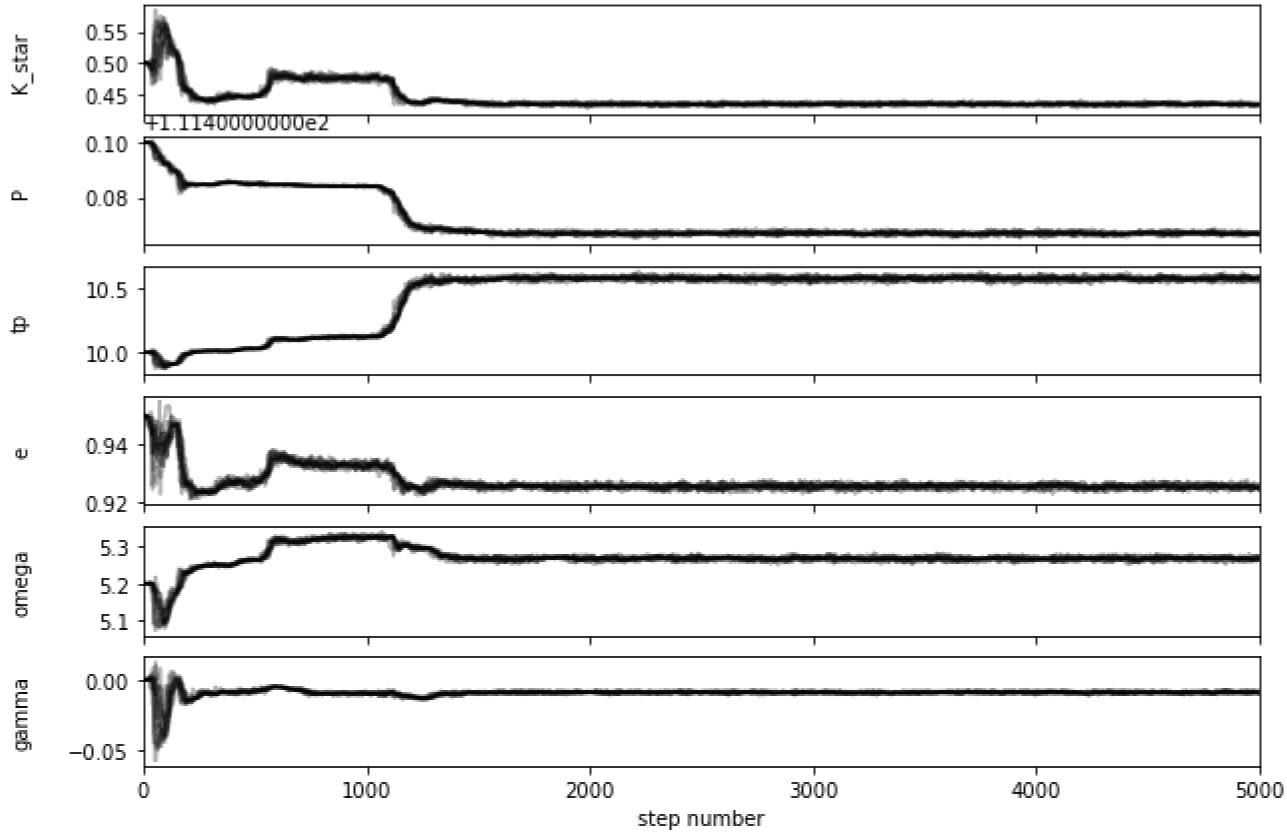
Run the MCMC

In [16]:
```python
np.random.seed(1)
pos = init + 1e-4 * np.random.randn(12, 6) # initialize starting positions of 16 walkers
nwalkers, ndim = pos.shape

# run MCMC
sampler = emcee.EnsembleSampler(
    nwalkers, ndim, log_probability,
    args=(df['t'], df['rv'], df['rv_err'])
)
sampler.run_mcmc(pos, 5000, progress=True);
```

```
100%|██████████████████████████████████████████████████████████████████████████| 5000/5000 [0
4:36<00:00, 18.10it/s]
```

Visualize traces to make sure nothing weird is happening

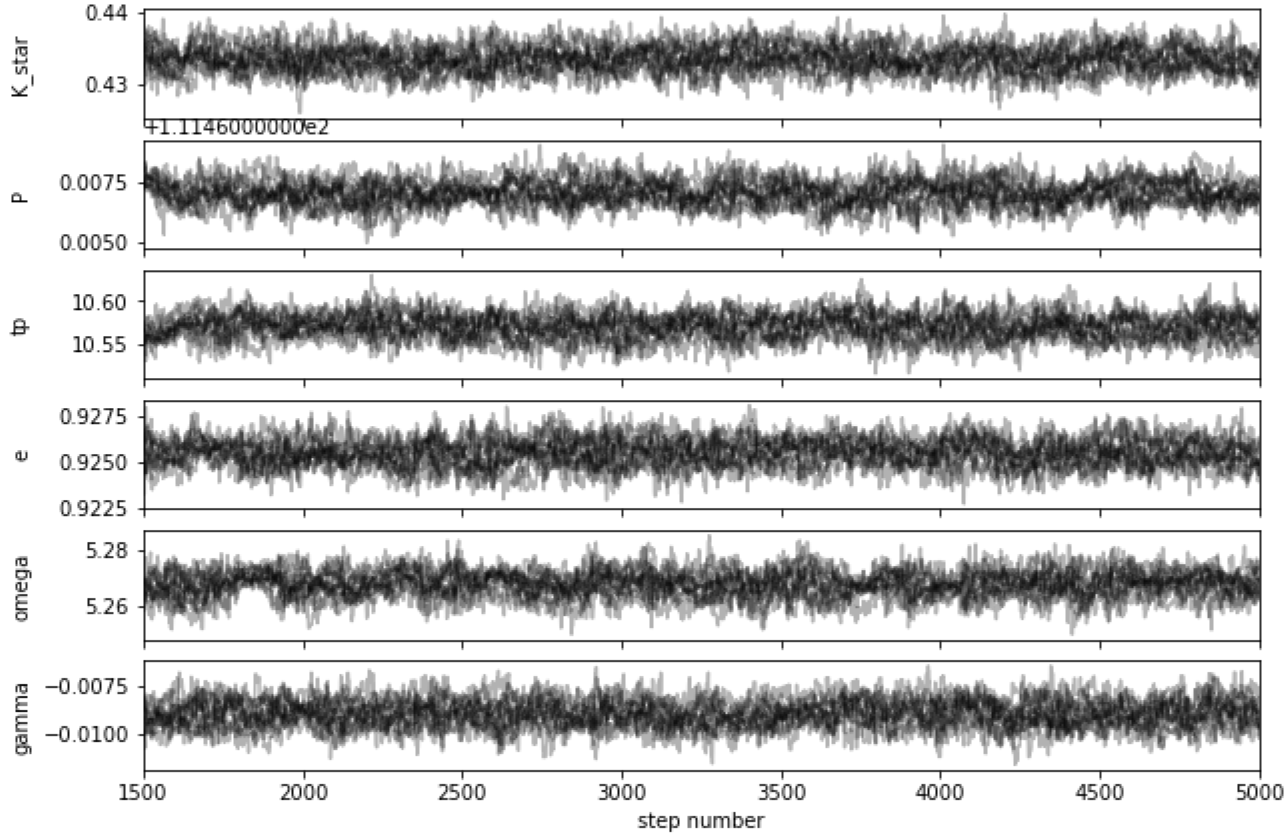
In [17]:
```python
fig, axes = plt.subplots(6, figsize=(10, 7), sharex=True)
samples = sampler.get_chain()
labels = ['K_star', 'P', 'tp', 'e', 'omega', 'gamma']
for i in range(6):
    ax = axes[i]
    ax.plot(samples[:, :, i], "k", alpha=0.3)
    ax.set_xlim(0, len(samples))
    ax.set_ylabel(labels[i])
    ax.yaxis.set_label_coords(-0.1, 0.5)

axes[-1].set_xlabel("step number");
```



Looks like it took the MCMC about 1500 steps to converge to the global minimum.

In [18]:

```
burnin = 1500
fig, axes = plt.subplots(6, figsize=(10, 7), sharex=True)
samples = sampler.get_chain(discard=burnin)
for i in range(6):
    ax = axes[i]
    ax.plot(range(burnin, burnin + len(samples)), samples[:, :, i], "k", alpha=0.3)
    ax.set_xlim(burnin, burnin + len(samples))
    ax.set_ylabel(labels[i])
    ax.yaxis.set_label_coords(-0.1, 0.5)

axes[-1].set_xlabel("step number");
```

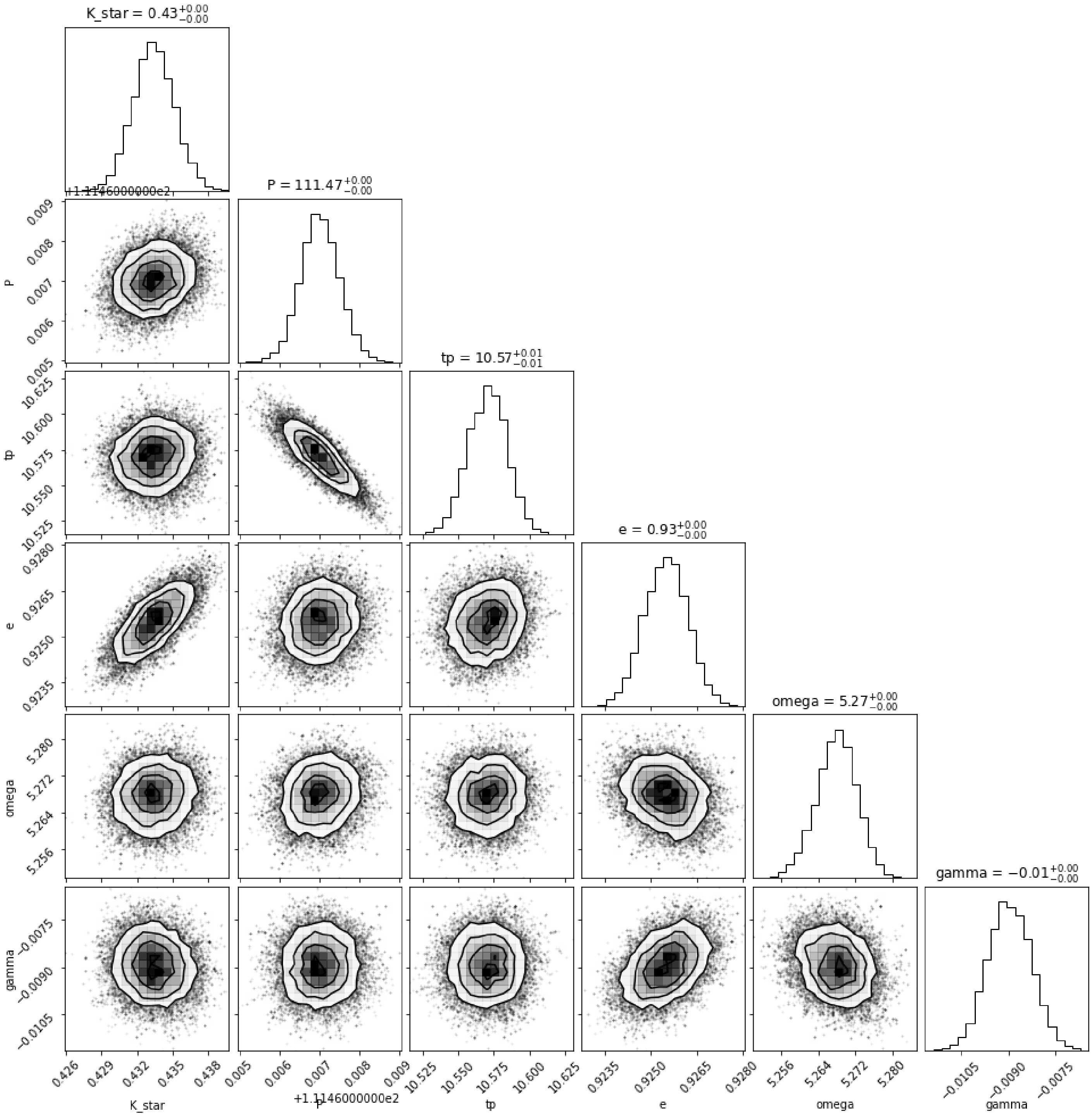


The autocorrelations seem a little high and if I were doing research I'd want to modify the step size and thin out the chain before plotting the posterior distributions. However, since this is just for class I'm going to forge ahead.

Make a corner plot:

In [19]:

```
import corner

flat_samples = sampler.get_chain(discard=burnin, flat=True)
fig = corner.corner(
    flat_samples, labels=labels, show_titles=True
);
```



Nice! The distributions look pretty smooth. Note the correlations between some of the parameters!

The fitted values of $(K_{star}, P, t_p, e, \omega, \gamma)$ are:

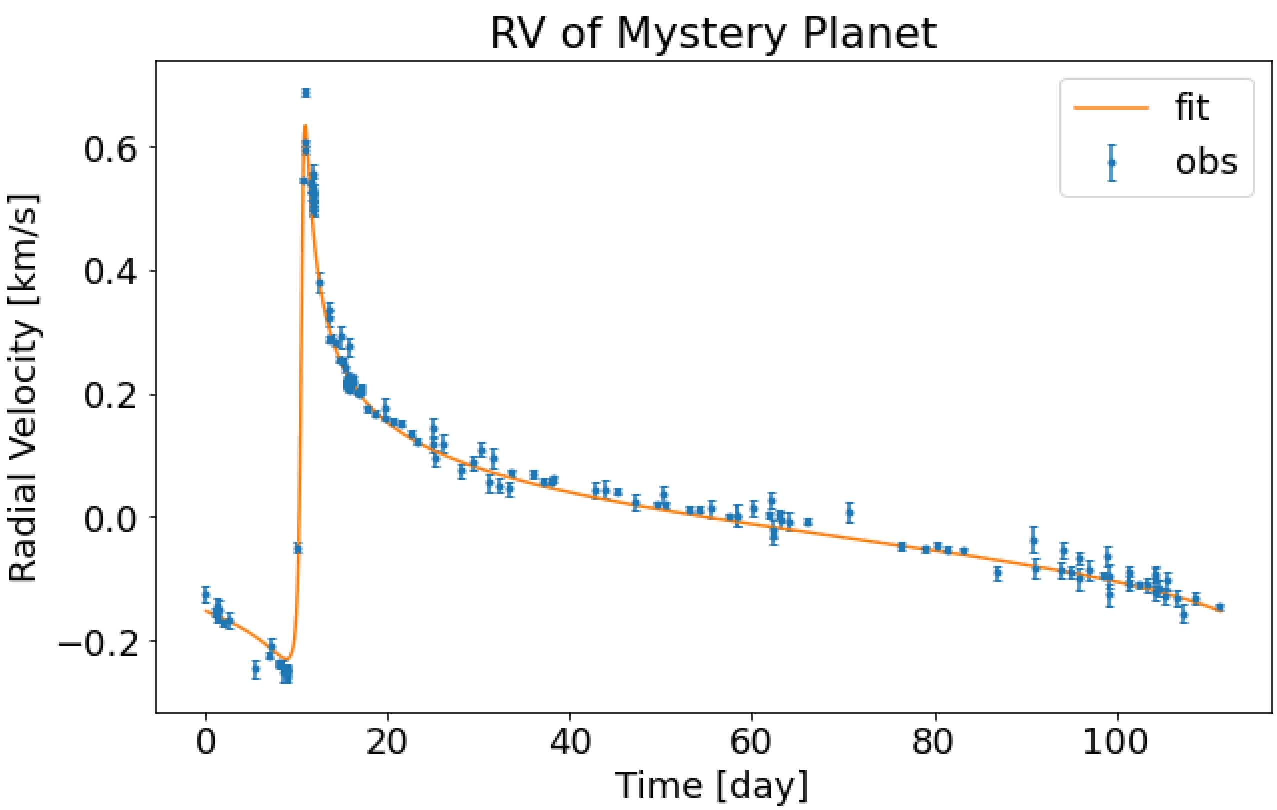
In [20]:
```
fit = flat_samples.mean(axis=0)
fit = np.median(flat_samples, axis=0)
for ii, param in enumerate(fit):
    print(labels[ii].ljust(10), round(param, 3))
```

```
K_star     0.433
P          111.467
tp         10.571
e          0.925
omega      5.268
gamma      -0.009
```

Plotting the model against the data, we see that the model does really well!

In [21]:

```
P = fit[1]
font = {'size' : 18}
rc('font', **font)

# plot fit
ts = np.linspace(0, P, 1000)
plt.figure(figsize=(10,6))
plt.title('RV of Mystery Planet')
plt.errorbar(df['t'] % P, df['rv'], df['rv_err'], label='obs', fmt='.', capsize=2)
plt.plot(ts, RV_model(ts, *fit), label='fit')
plt.xlabel('Time [day]')
plt.ylabel('Radial Velocity [km/s]')
plt.legend()
plt.show()
```



What an eccentric planetary system ;)

Q6

Note, in the write up, $\mathbf{r} = \mathbf{x}$ and $\dot{\mathbf{r}} = \mathbf{v}$

a.

Conservation of angular momentum means that cross product of the position and velocity of an object $\mathbf{r} \times \dot{\mathbf{r}}$ in orbit is conserved and equal to $|h|\hat{z}$ where $\hat{z} = \hat{r} \times \dot{\hat{r}}$ is perpendicular to the plane of orbit. Thus $\mathbf{r(t)} \times \dot{\mathbf{r(t)}} = \mathbf{r}_0 \times \dot{\mathbf{r}_0}$ for all t as long as there is no change to the orbit.

Now to prove the relation:

$$f\dot{g} - g\dot{f} = 1$$

We know that by applying the f and g equations, we can find $\mathbf{r(t)}$ and $\dot{\mathbf{r(t)}}$ as a function of $\mathbf{r}_0$ and $\dot{\mathbf{r}_0}$:

$$\mathbf{r} = f\mathbf{r}_0 + g\dot{\mathbf{r}}_0$$
$$\dot{\mathbf{r}} = \dot{f}\,\mathbf{r}_0 + \dot{g}\dot{\mathbf{r}}_0$$

Taking the cross products of each side:

$$\mathbf{r} \times \dot{\mathbf{r}} = (f\mathbf{r}_0 + g\dot{\mathbf{r}}_0) \times (\dot{f}\,\mathbf{r}_0 + \dot{g}\dot{\mathbf{r}}_0)$$

Using the following two identities for the crossproduct:

$$\mathbf{x} \times \mathbf{x} = 0$$
$$\mathbf{x} \times \mathbf{y} = -\mathbf{y} \times \mathbf{x}$$

We can reduce the RHS to:

$$\mathbf{r} \times \dot{\mathbf{r}} = (f\mathbf{r}_0 + g\dot{\mathbf{r}}_0) \times (\dot{f}\,\mathbf{r}_0 + \dot{g}\dot{\mathbf{r}}_0)$$
$$\mathbf{r} \times \dot{\mathbf{r}} = (f\mathbf{r}_0 \times \dot{g}\dot{\mathbf{r}}_0) + (g\dot{\mathbf{r}}_0 \times \dot{f}\,\mathbf{r}_0)$$
$$\mathbf{r} \times \dot{\mathbf{r}} = (f\dot{g} + -g\dot{f}\,)(\mathbf{r}_0 \times \dot{\mathbf{r}}_0)$$

However, because of conservation of momentum, $\mathbf{r} \times \dot{\mathbf{r}} = \mathbf{r}_0 \times \dot{\mathbf{r}}_0$, so we are left with:

$$\boxed{f\dot{g} - g\dot{f}\, = 1}$$

b.

The f and g functions are:

$f = \frac{a}{r_0}(\cos(E - E_0) - 1) + 1$
$g = (t - t_0) + \frac{1}{n}(\sin(E - E_0) - (E - E_0))$
$\dot{f}\, = -\frac{a^2}{rr_0} n \sin(E - E_0)$
$\dot{g} = \frac{a}{r}(\cos(E - E_0) - 1) + 1$

We will set our reference frame so that $E_0 = M_0 = 0$ and that $M = n(t - t_0)$ and also pull out factors of $\frac{a}{r_0}$ and $\frac{a}{r}$ out of f and $\dot{g}$ We can rewrite these equations:

$f = \frac{a}{r_0}(\cos E - 1 + \frac{r_0}{a})$
$g = \frac{1}{n}(M + \sin E - E)$
$\dot{f}\, = -\frac{a^2}{rr_0} n \sin E$
$\dot{g} = \frac{a}{r}(\cos E - 1) + \frac{r}{a})$

Now solving for $f\dot{g} - g\dot{f}$:

$f\dot{g} - g\dot{f}$
$= [\frac{a}{r_0}(\cos E - 1 + \frac{r_0}{a})][\frac{a}{r}(\cos E - 1) + \frac{r}{a})] - [\frac{1}{n}(M + \sin E - E)][-\frac{a^2}{rr_0} n \sin E]$
$= \frac{a^2}{rr_0}\left[(\cos E - 1 + \frac{r_0}{a})(\cos E - 1) + \frac{r}{a}) + (M + \sin E - E)\sin E\right]$

Making the following substitutions:

$M = E - e \sin E$
$r = a(1 - e \cos E)$
$r_0 = a(1 - e \cos E_0) = a(1 - e)$

We get:

$f\dot{g} - g\dot{f}$

$= \frac{a^2}{rr_0}\left[[(\cos E - 1 + 1 - e)(\cos E - 1 + 1 - e\cos E)] + [(E - e\sin E + \sin E - E)\sin E]\right]$

$= \frac{a^2}{rr_0}\left[[(\cos E - e)(\cos E - e\cos E)] + [\sin E(1 - e)\sin E]\right]$

$= \frac{a^2}{rr_0}\left[\cos^2 E - e\cos^2 E - e\cos E + e^2\cos E + \sin^2 E - e\sin^2 E\right]$

$= \frac{a^2}{rr_0}\left[\cos^2 E + \sin^2 E - e(\cos^2 E + \sin^2 E) - e\cos E + e^2\cos E\right]$

$= \frac{a^2}{rr_0}\left[1 - e - e\cos E + e^2\cos E\right]$

$= \frac{a^2}{rr_0}\left[(1 - e)(1 - e\cos E)\right]$

$= \frac{a^2}{rr_0}\frac{rr_0}{a^2}$

$\boxed{= 1}$

Thus if Kepler's equation is satisfied, $f\dot{g} - g\dot{f} = 1$ and therefore angular momentum is conserved