In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import emcee
import celerite

from celerite import terms
from scipy.optimize import minimize
c2='#f15a22'
```

In [2]:
```
def log_like(params,y,gp,method):
    gp.set_parameter_vector(params)
    if method=='MAP':
        return -gp.log_likelihood(y)+0.5*params[0]-params[1]*0.5
    else: return -gp.log_likelihood(y)

def log_probability(params,y,gp):
        gp.set_parameter_vector(params)
        lp=gp.log_prior()
        log_a=gp.get_parameter_dict().get('kernel:log_a')
        log_c=gp.get_parameter_dict().get('kernel:log_c')
        if not np.isfinite(lp):
            return -np.inf
        return gp.log_likelihood(y)+lp-0.5*log_a+log_c*0.5


def DRW_fit(t,s,err,method='MAX',tl=[1,5000],sl=[0.02,0.7]):
    # Set up the GP model
    bounds=dict(log_a=(2*np.log(sl[0]),2*np.log(sl[1])),
                log_c=(-np.log(tl[1]),-np.log(tl[0])))
    kernel=terms.RealTerm(log_a=np.log(0.1414),log_c=-np.log(400),bounds=bounds)
    gp=celerite.GP(kernel,mean=np.mean(s),fit_mean=True)
    gp.compute(t,err)
    # Fit for the maximum likelihood parameters
    initial_params = gp.get_parameter_vector()
    soln=minimize(log_like, initial_params,method="L-BFGS-B",args=(s,gp,method))

    gp.set_parameter_vector(soln.x)
    rt=np.exp(-gp.get_parameter_dict().get('kernel:log_c'))#MLE or MAP
    rs=np.exp(gp.get_parameter_dict().get('kernel:log_a')/2)
    if method=='MLE' or method=='MAP':
        return np.array([rt,rs])
    #MCMC
    initial=np.array(soln.x)
    ndim, nwalkers=len(initial),16
    sampler=emcee.EnsembleSampler(nwalkers,ndim,log_probability,args=(s,gp))
    #Running burn-in
    p0=initial+1e-4*np.random.randn(nwalkers,ndim)
    p0,lp,_=sampler.run_mcmc(p0,125)
    #Running production
    sampler.reset()
    sampler.run_mcmc(p0,500)
    #Markov chains
    t_chain=np.exp(np.sort(-sampler.flatchain[:,1]))
    s_chain=np.exp(np.sort(sampler.flatchain[:,0]/2))
    return t_chain
```

In [4]:
```
y=np.loadtxt('141',usecols=[3,4,5])
n=int(np.sqrt(len(y)))
print(max(y[:,-1]))
y[:,0:2]=np.power(10,y[:,0:2])
y[:,2]=y[:,2]-500
y[:,2]=np.exp(y[:,2])

tl=np.logspace(np.log10(0.03),np.log10(0.03)+6,n)
sl=np.logspace(np.log10(0.01),np.log10(30),n)

tau=np.array([])
for i in range(n):
    tau=np.append(tau,np.trapz(y[n*i:(i+1)*n,2]/y[n*i,0]/y[n*i:(i+1)*n,1],y[n*i:(i+1)*n,1]))
norm=np.trapz(tau,tl)

print(norm)
```

```
1066.74
6.292325039805142e+245
```

In [5]:
```
tau=tau/norm
```

In [6]:
```
x=np.loadtxt('lc01')
```

In [7]:
```
chain=DRW_fit(x[:,0],x[:,1],x[:,2])
```

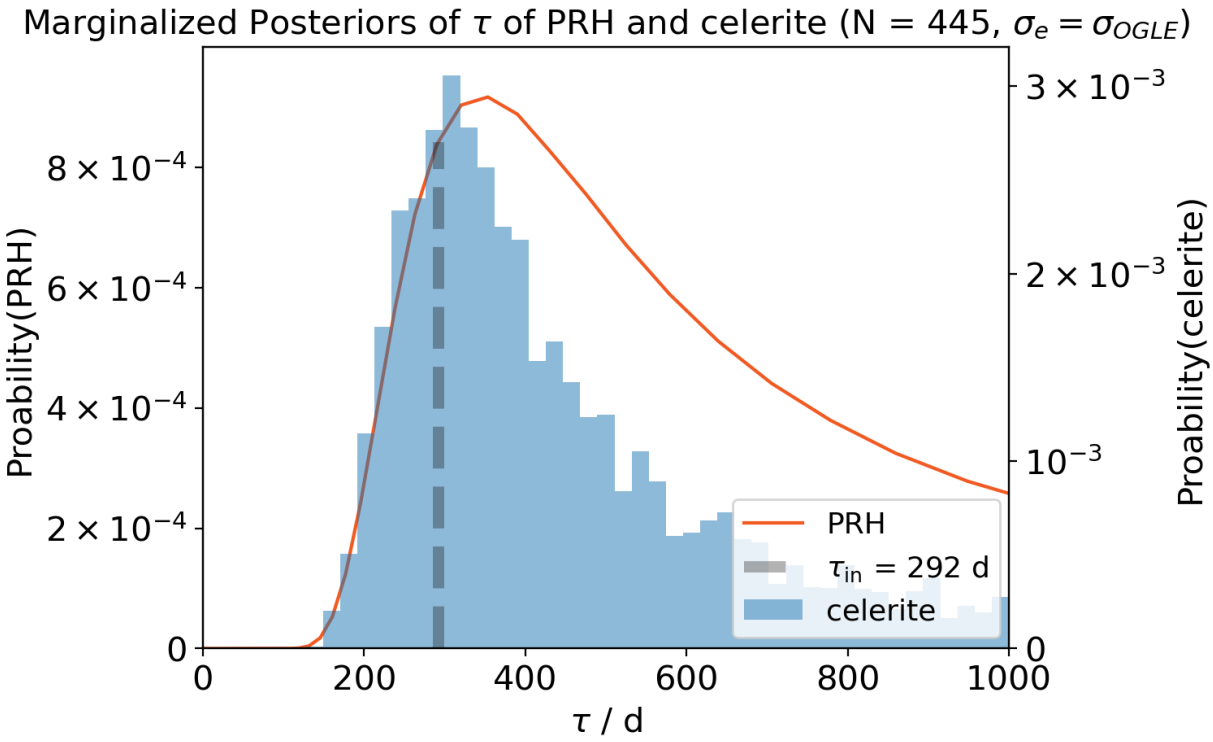
In [15]:
```
fig = plt.figure(dpi=200)
ax = fig.add_subplot(111)
ax.plot(tl,tau,color=c2)
ax.set_xlim(0,1000)
ax.set_ylim(0,1e-3)
ax.set_xlabel(r"$\tau$ / d",fontsize=15)
ax.set_xticklabels(range(0,1200,200),fontsize=15)
ax.set_ylabel(r"Proability(PRH)",fontsize=15)
ax.set_yticks([0,2e-4,4e-4,6e-4,8e-4],
              ['0',r'$2\times10^{-4}$',r'$4\times10^{-4}$',r'$6\times10^{-4}$',r'$8\times10^{-4}$'],
              fontsize=15)

ax2 = ax.twinx()
ax2.hist(chain,density=True,bins=1410,range=(0.03,30000),alpha=0.5)
ax2.plot(np.full(10,292),np.linspace(0,2.7e-3,10),'k--',lw=5,alpha=0.3)
ax2.set_ylabel(r"Proability(celerite)",fontsize=15)
ax2.set_xlim(0,1000)
ax2.set_yticks([0,1e-3,2e-3,3e-3],
               ['0',r'$10^{-3}$',r'$2\times10^{-3}$',r'$3\times10^{-3}$'],fontsize=15)
#ax2.set_xticks(range(0,1200,200),fontsize=15)

fig.legend(['PRH',r'$\tau_{\rm in}$ = 292 d','celerite'],loc=2,bbox_to_anchor=(0.62,0.32),fontsize=13)

ax.set_title(r'Marginalized Posteriors of $\tau$ of PRH and celerite (N = 445, $\sigma_e=\sigma_{OGLE}$)',
             fontsize=15)
#ax2.set_xticks(range(0,1200,200),fontsize=15)
plt.show()
```

```
/tmp/ipykernel_5469/3001663790.py:7: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(range(0,1200,200),fontsize=15)
```



In [ ]: