# Time delay estimation

(case of gravitationally-lensed quasars)

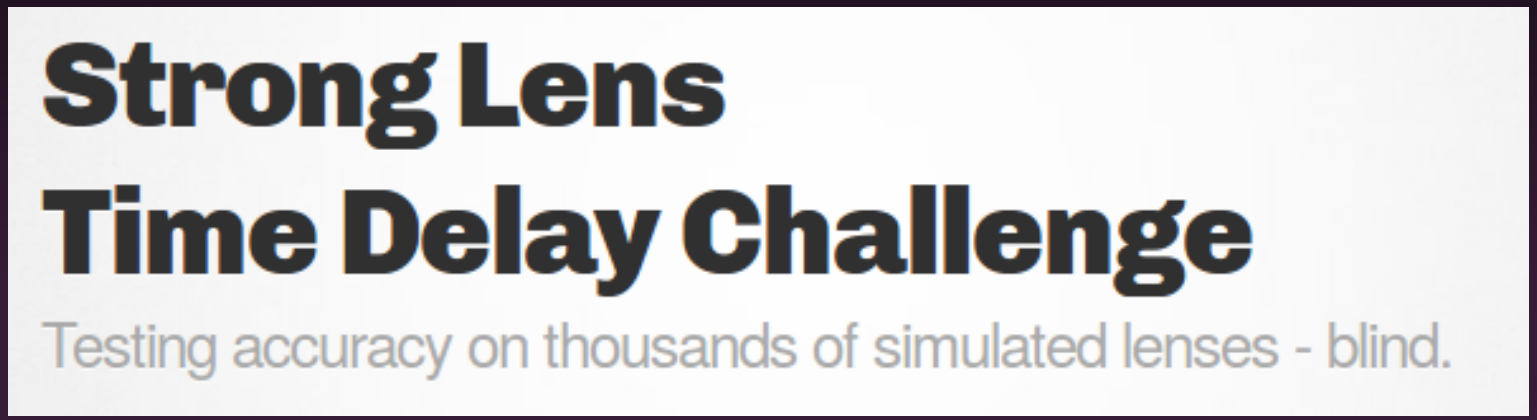
Saghar Asadi

May 17, 2016

# Outline

- Question
- Dataset
- Main idea
- What we tried
  - Unsuitable methods
  - (probably) Suitable methods
- Current state
- Way to improvement

# Question - Original




TimeDelayChallenge.org

# Question - Machine learning approach

**How much do the ground truth information mean in terms of accuracy?**

**OR:**
How limiting are the intrinsic uncertainties?

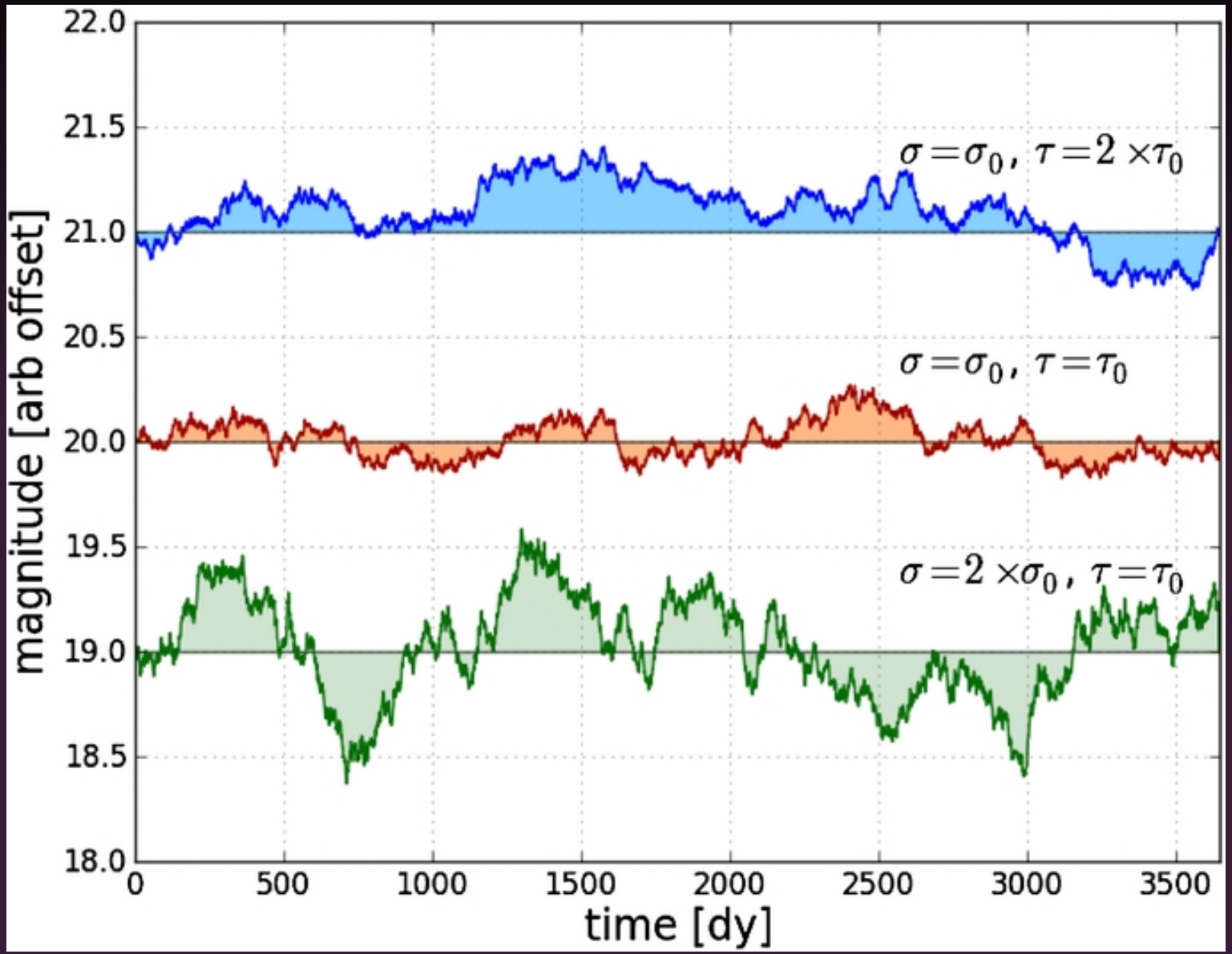
**Meta questions:**
What does this mean in terms of the Hubble constant?
Does the LSST observing strategy need to be modified?

# Dataset - Inroduction

Realistic mock observed lensed quasar light curves

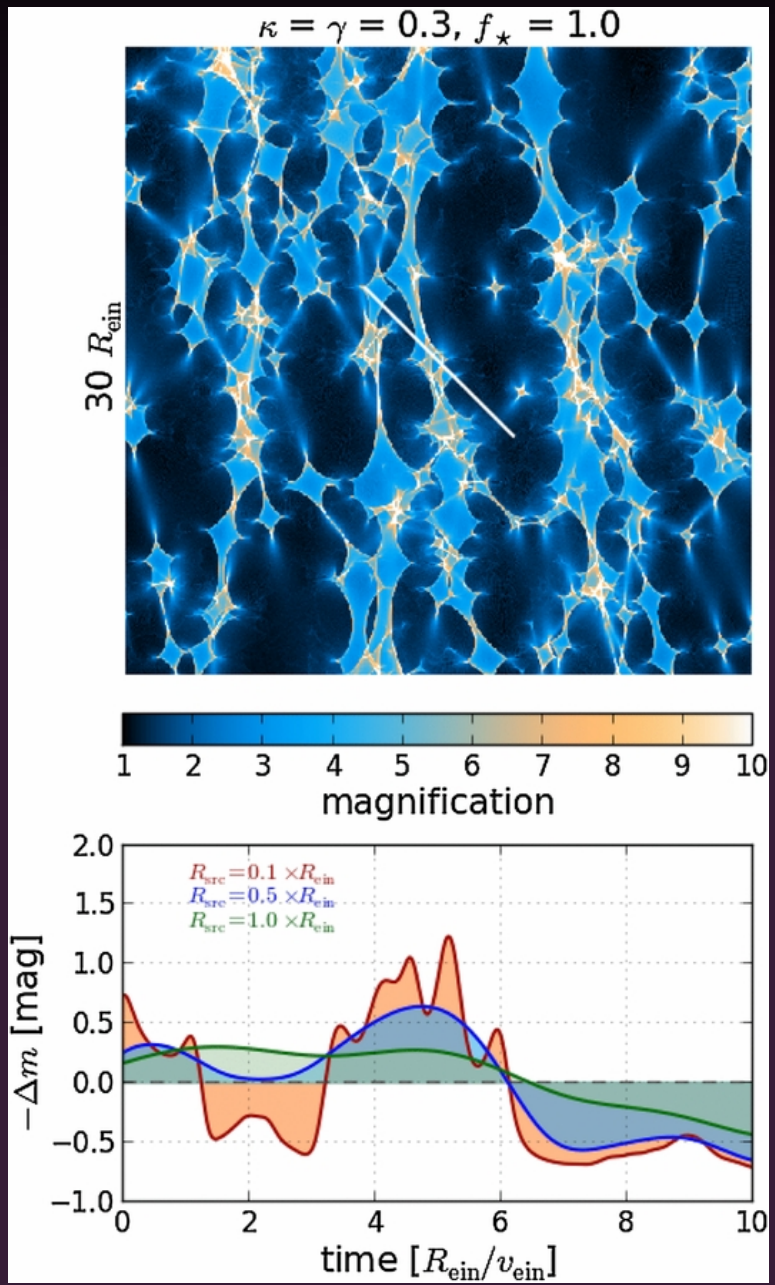
# Dataset - AGN variability

Dobler+2015



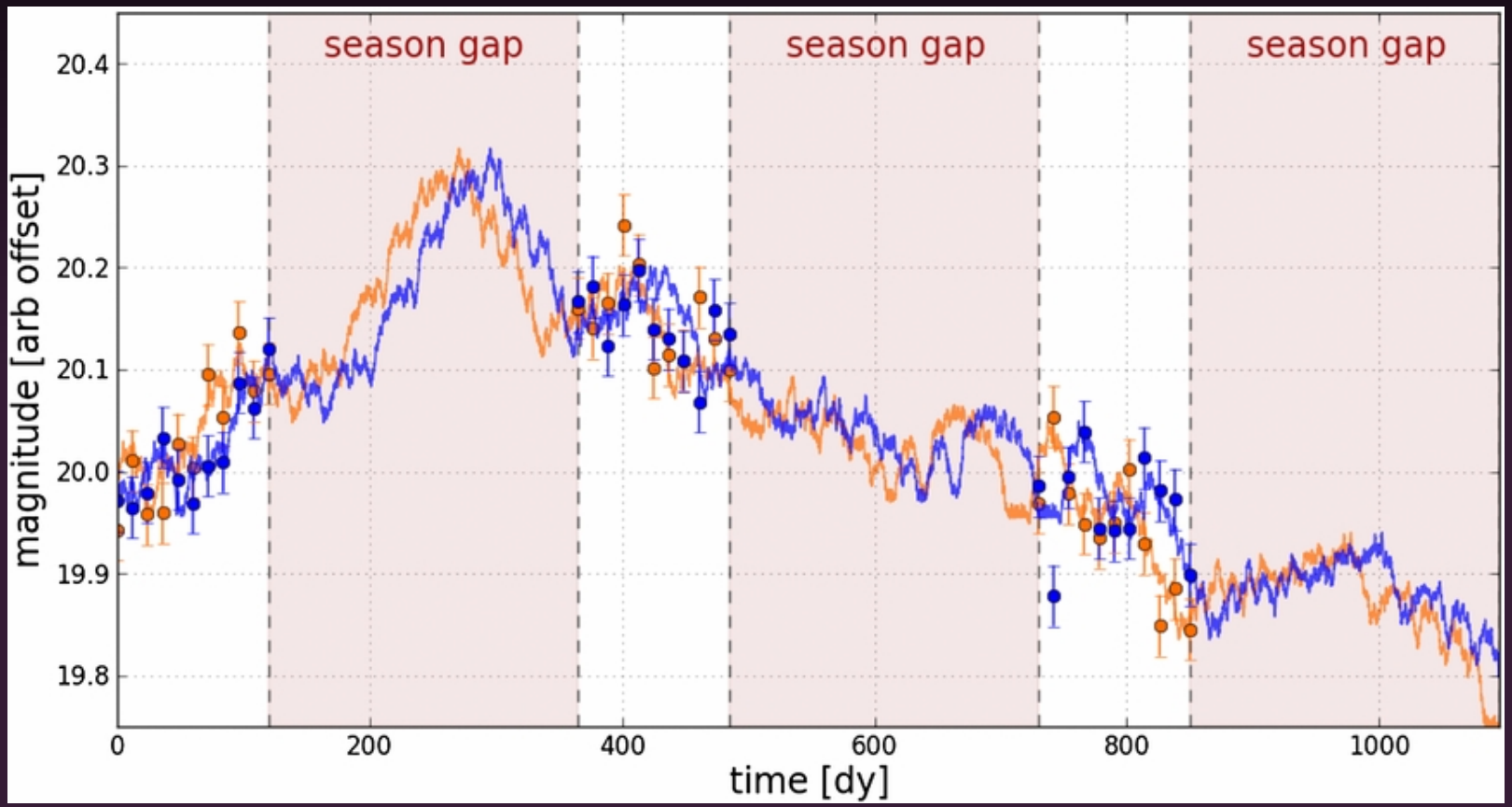
# Dataset - Microlensing effects

Dobler+2015



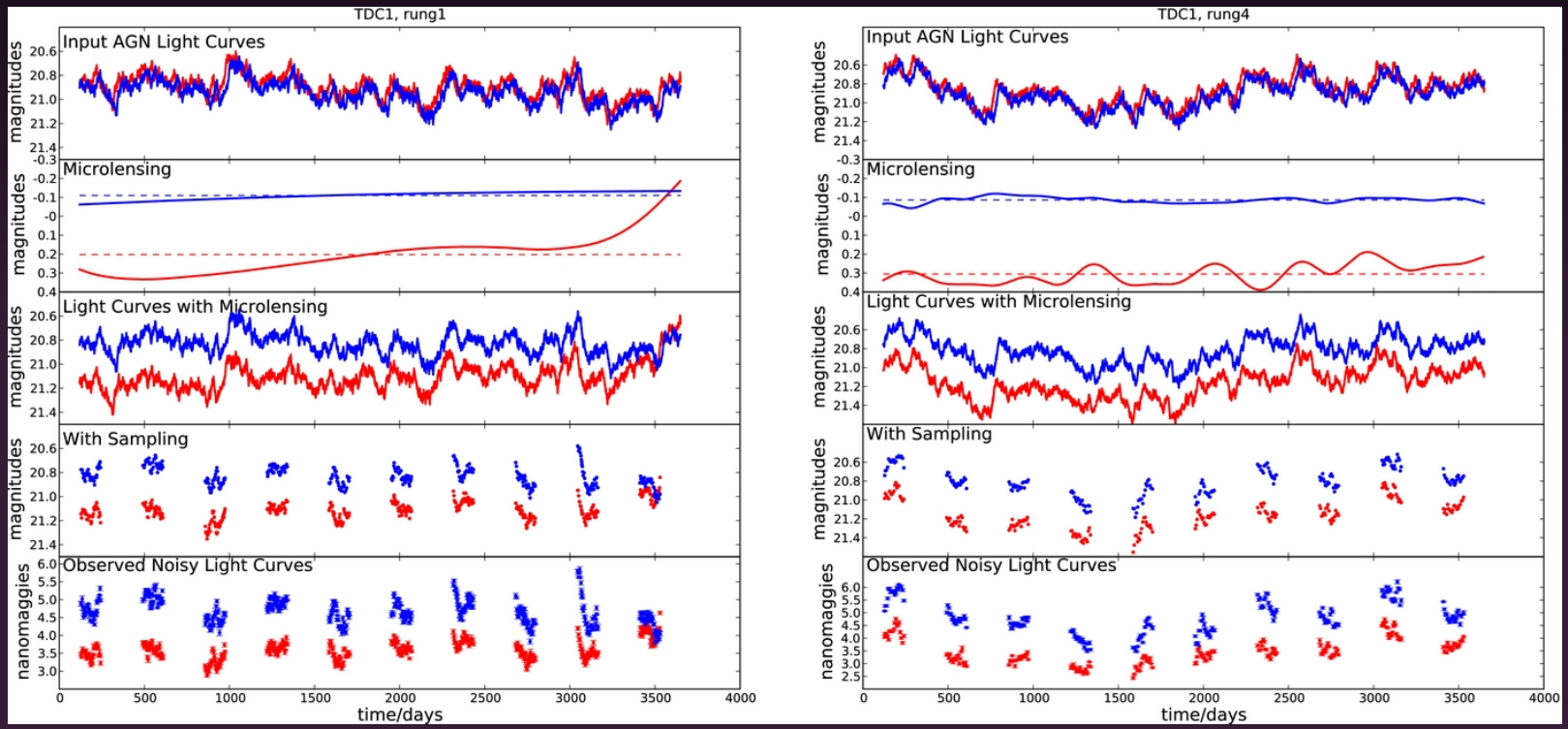
# Dataset - Observational effects

Dobler+2015



# Dataset

Liao+2015



# Dataset - for each pair

```
## Time Delay Challenge light curves
##
## [time]=days, [lc]=[err]=flux in nanomaggies
##
##       time       lc_A      err_A       lc_B      err_B
##-------------------------------------------------------
      1.87000    1.02659    0.03866    2.50732    0.07133
      5.99000    0.91101    0.05243    2.52054    0.02973
      7.32000    1.04086    0.06210    2.54686    0.06845
      9.83000    1.07427    0.03612    2.44915    0.04517
     13.14000    1.19277    0.08868    2.66788    0.05020
```

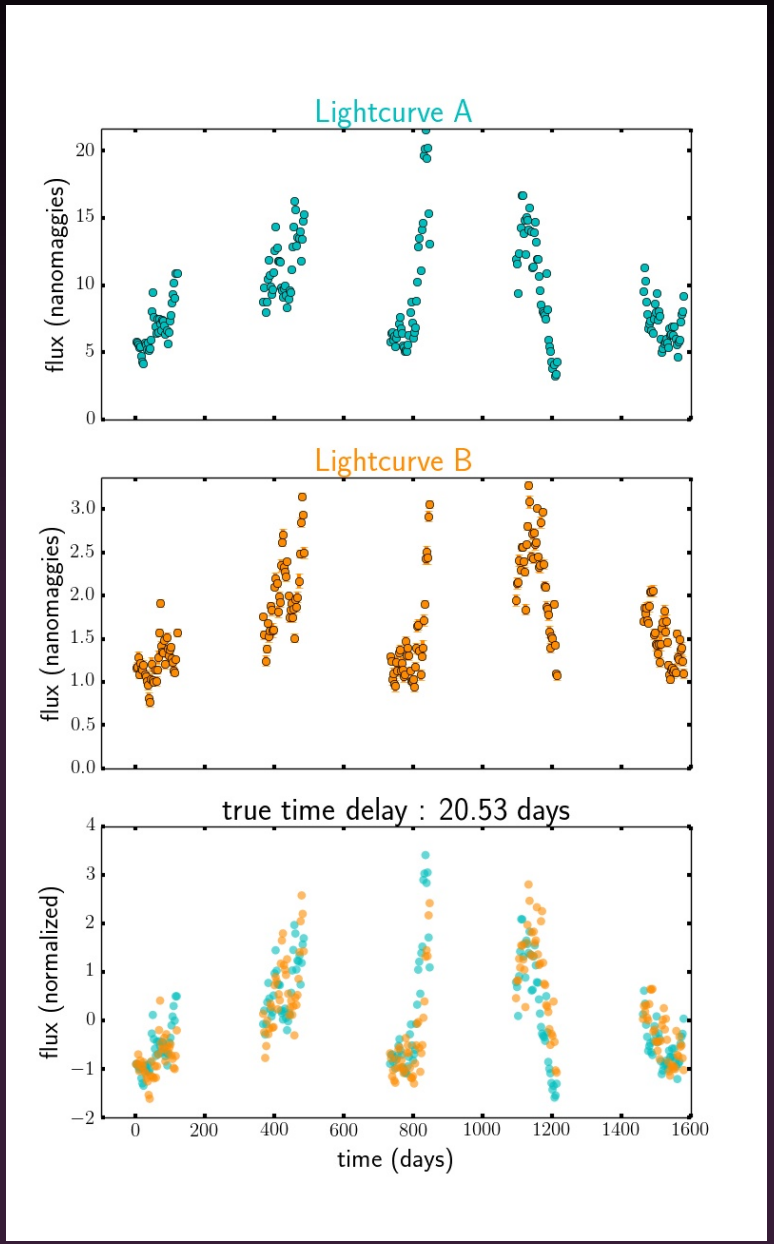
# Dataset - ground truth

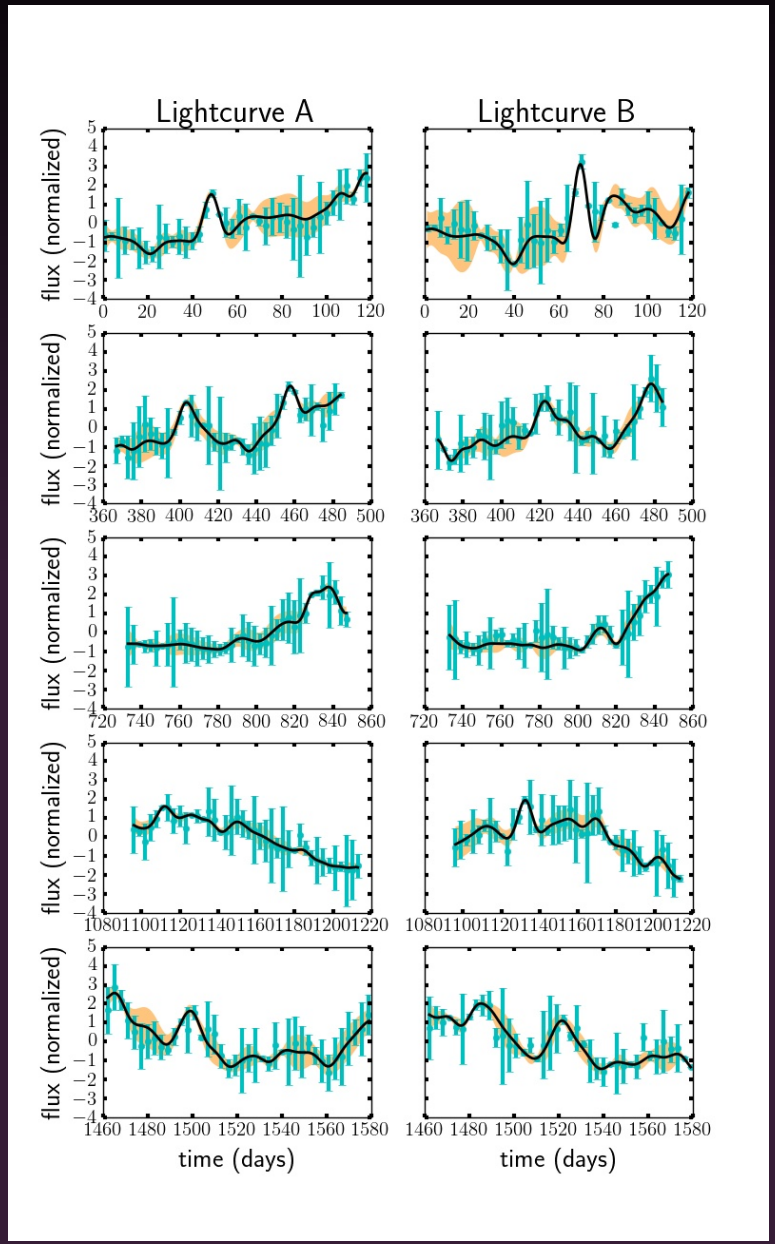
| #name | dt | m1 | m2 | zl | zs | id | tau | sig |
|---|---|---|---|---|---|---|---|---|
| tdc1_rung0_double_pair1.txt | -70.89 | 21.56 | 22.64 | 0.654 | 3.0 | 10879293 | 273.0681 | 0.01511 |
| tdc1_rung0_double_pair2.txt | 104.22 | 19.6 | 20.48 | 1.168 | 3.61 | 42276315 | 485.5498 | 0.02137 |
| tdc1_rung0_double_pair3.txt | 56.59 | 21.62 | 22.96 | 0.728 | 4.41 | 14204145 | 691.05658 | 0.00502 |
| tdc1_rung0_double_pair4.txt | -79.61 | 20.87 | 23.13 | 0.792 | 1.9 | 17432214 | 32.75536 | 0.02914 |
| tdc1_rung0_double_pair5.txt | -5.78 | 21.51 | 22.6 | 0.388 | 3.54 | 2736515 | 53.4707 | 0.01867 |
| tdc1_rung0_double_pair6.txt | 33.85 | 20.63 | 21.3 | 0.27 | 1.54 | 1012286 | 387.21437 | 0.00975 |
| tdc1_rung0_double_pair7.txt | -71.07 | 21.33 | 20.46 | 0.616 | 3.04 | 9364230 | 36.49722 | 0.02042 |
| tdc1_rung0_double_pair8.txt | 47.25 | 21.22 | 22.14 | 0.236 | 3.01 | 694868 | 278.76035 | 0.02772 |
| tdc1_rung0_double_pair9.txt | 21.57 | 23.21 | 22.64 | 1.722 | 4.71 | 91690964 | 259.56163 | 0.02477 |
| tdc1_rung0_double_pair10.txt | -39.05 | 22.27 | 22.63 | 1.496 | 3.98 | 70257822 | 786.49613 | 0.01069 |

# Main idea

- **Smooth** and **interpolate** evenly sampled data
- **Compare** two light curves of each window for dt between 0 and window length
  - `scipy.signal.correlate`
  - MSE
- Find the **best timeshift** for each window
  - `max(correlate)`
  - `min(MSE)`
- **Compare** estimated best dt of different windows of the same pair
  - `np.median(dt[window])`
  - `np.mean(dt[window])`
  - `np.std(dt[window])`
  - weighted mean (based on absolute correlate value)
- Use a **clustering** algorithm to reduce the noise
- Apply a **regression** method to the clustered values

# Dataset - smooth (Gaussian processes)



# Dataset - smooth (Gaussian processes)

```
gp = GaussianProcess(corr = "squared_exponential",
                     regr = "quadratic",
                     theta0 = sigma,
                     thetaL = 1e-2,
                     thetaU = tau,
                     nugget = (dy / y) ** 2,
                     random_start=500)


gp.fit(X, y)

y_pred, MSE = gp.predict(x, eval_MSE=True)
```

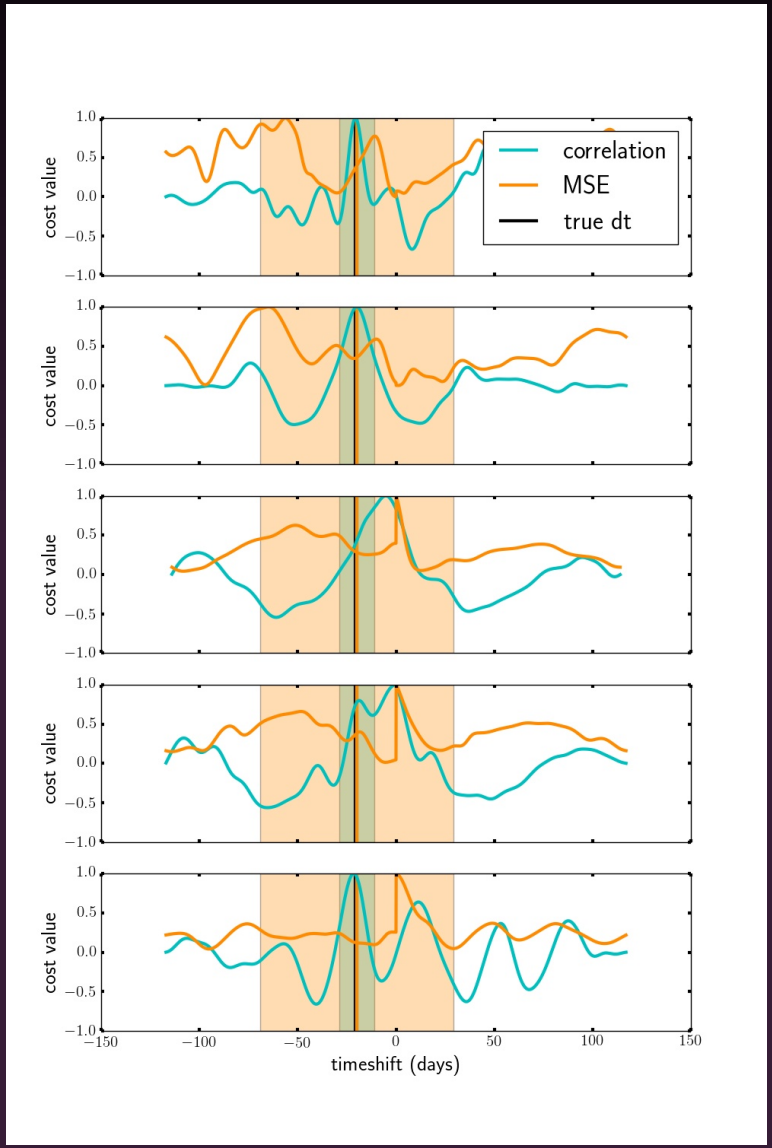
# Comparison - What we tried ... and failed

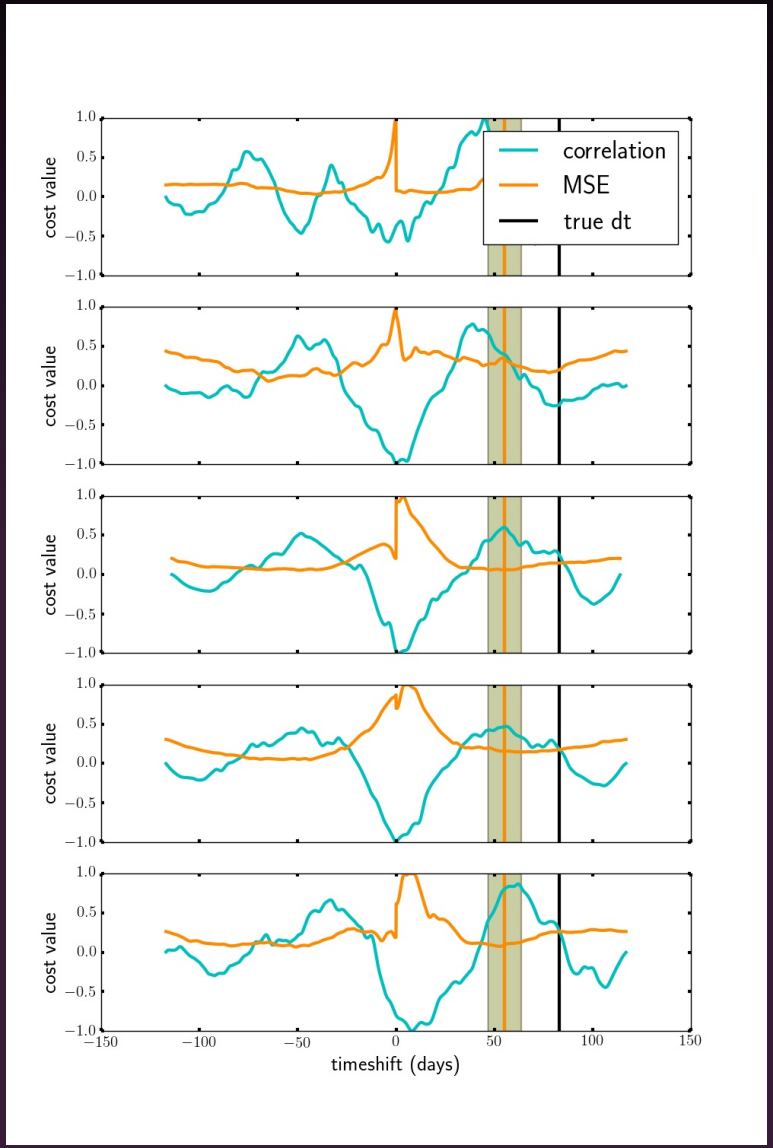
- Lomb-Scargle periodogram on raw, unevenly-sampled data
- FFT on even resampling of the smooth models

**Idea**: phase (angle of the complex FFT value) of the highest-amplitude frequency would correlate with the real dt.
**Problem**: Inside each window, the signal is highly a-periodic, which probably introduces a lot of noise.

- Linear regression (regularized and unregularized) with even resampling of smooth models + their uncertainties as features

# Comparison - What we tried ... and succeeded...kind of



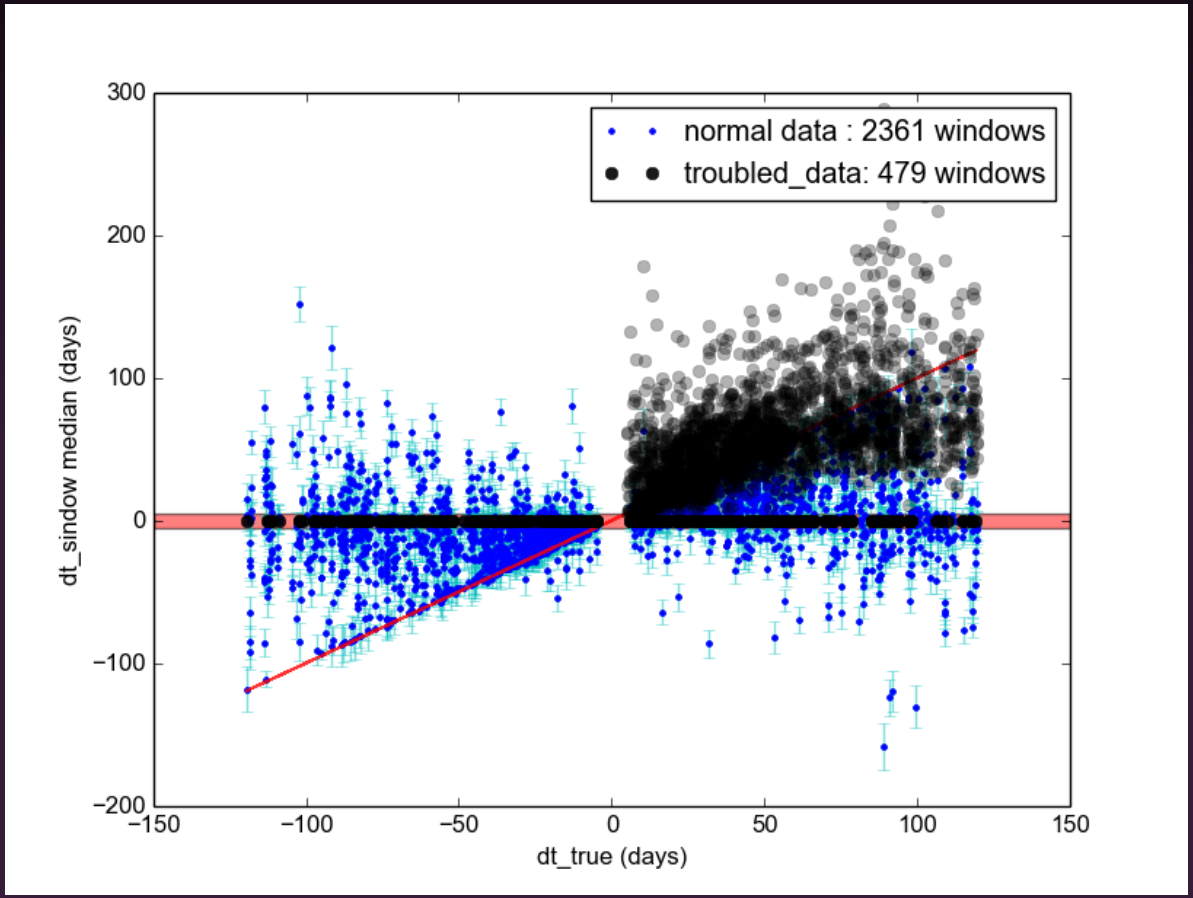



# Redice noise - clustering

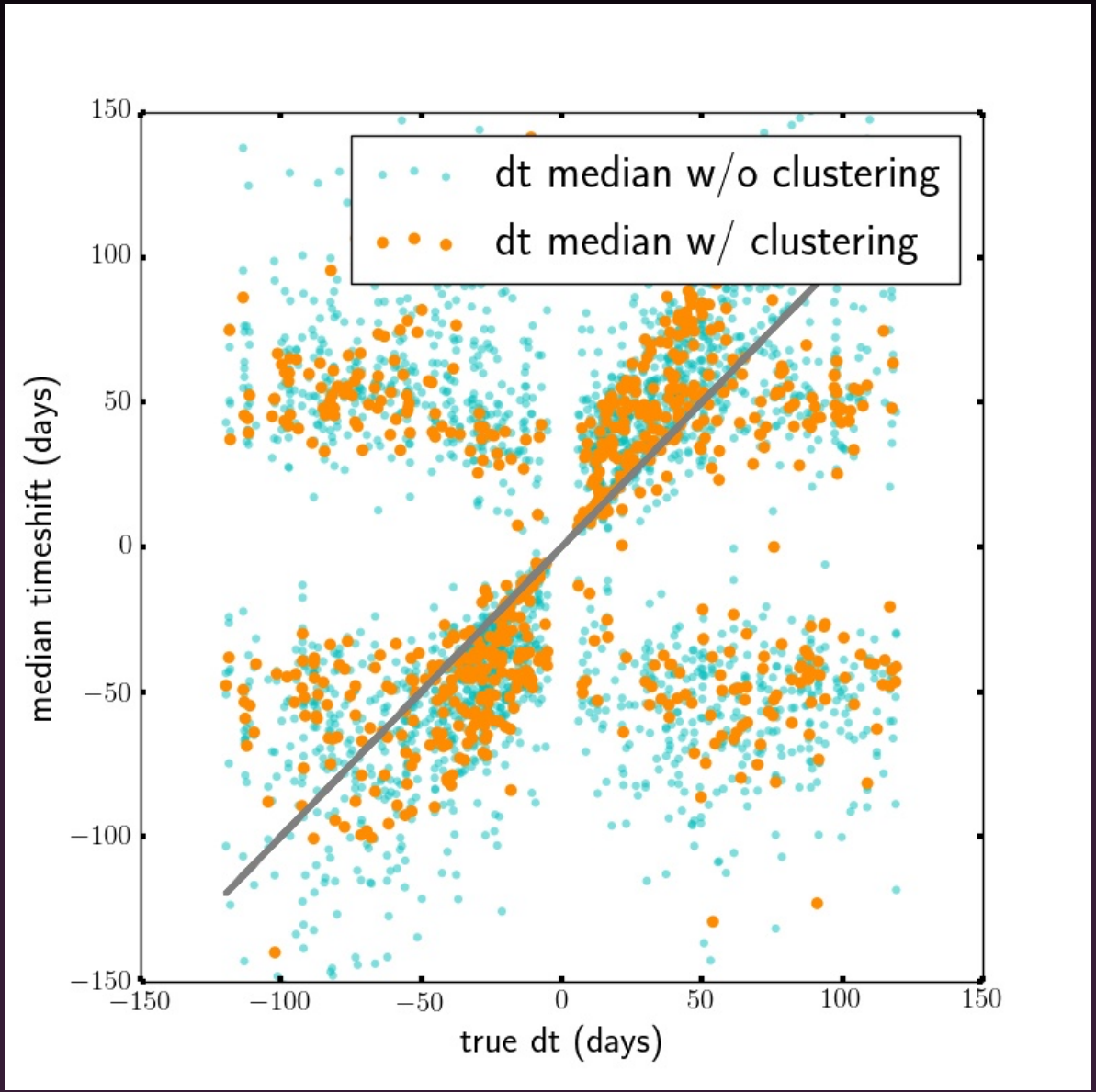
```
def findcluster_for_one_dt(x):
    res = np.zeros(len(x))
    for i in xrange(0, len(x)):
        res[i] = (1 / (1 + (x[i] - x)**2)).sum()
    return x[np.argmax(res)]

def cluster(res, dts, y):
    dt_values = np.unique(dts)
    res = zeros(len(dt_values))
    for i, dt in enumerate(dt_values):
        res[i] = findcluster_for_one_dt(y[dts == dt])
    return res
```

# Reduce noise - clustering



# Reduce noise - clustering



# Way to improvement

- Error analysis
- Regularized linear regression on correlation arrays
- SVM regression
- Unsupervised learning
- Neural network

# Future progress

Follow the project on GitHub:
https://github.com/asadisaghar/TimeDelay

# Metrics of success

- goodness-of-fit
- efficiency
- precision
- accuracy or bias

- $\chi^2$ < 1.5
- f > 0.5
- P < 0.03
- |A| < 0.03