TIME SERIES ANALYSIS WITH GAUSSIAN PROCESSES AND THE SEARCH FOR EARTH 2.0

Dan Foreman-Mackey
CCPP@NYU // github.com/dfm // @exoplaneteer // dfm.io

I write code for good & astrophysics.

Photo credit **NASA Ames/SETI Institute/JPL-Caltech**

this isn't what
my data look like

I study space.

Photo credit **NASA Ames/SETI Institute/JPL-Caltech**

I do data science.

not data science.

Photo credit **James Silvester**
silvesterphoto.tumblr.com

data science.

data science.

my

GOALS

for today's talk

convince you that

exoplanets are cool

GOALS

for today's talk

convince you that

exoplanets are cool

demonstrate some

sick Python code

for today's talk

ex·o·plan·et

ˈeksōˌplanət/

noun. *a planet that orbits a star outside the solar system.*

Credit **Google**

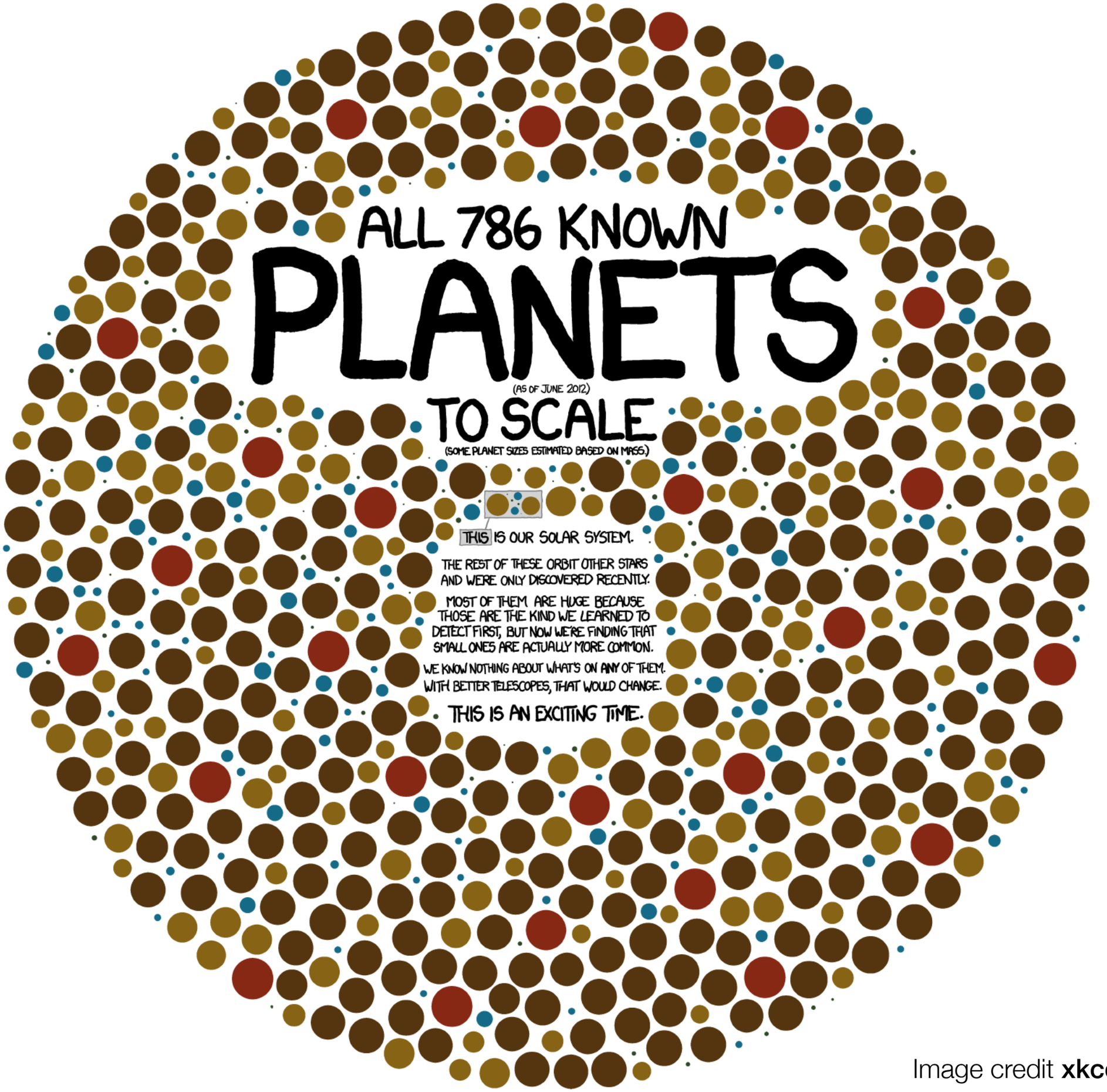


Image credit **xkcd.com/1071**

OUR NEIGHBORHOOD

A PORTRAIT OF ALL HABITABLE-ZONE PLANETS WITHIN 60 LIGHT-YEARS OF EARTH
(CONSTRUCTED FROM STATISTICAL DATA ON TYPICAL PLANET SIZES AND ORBITS)

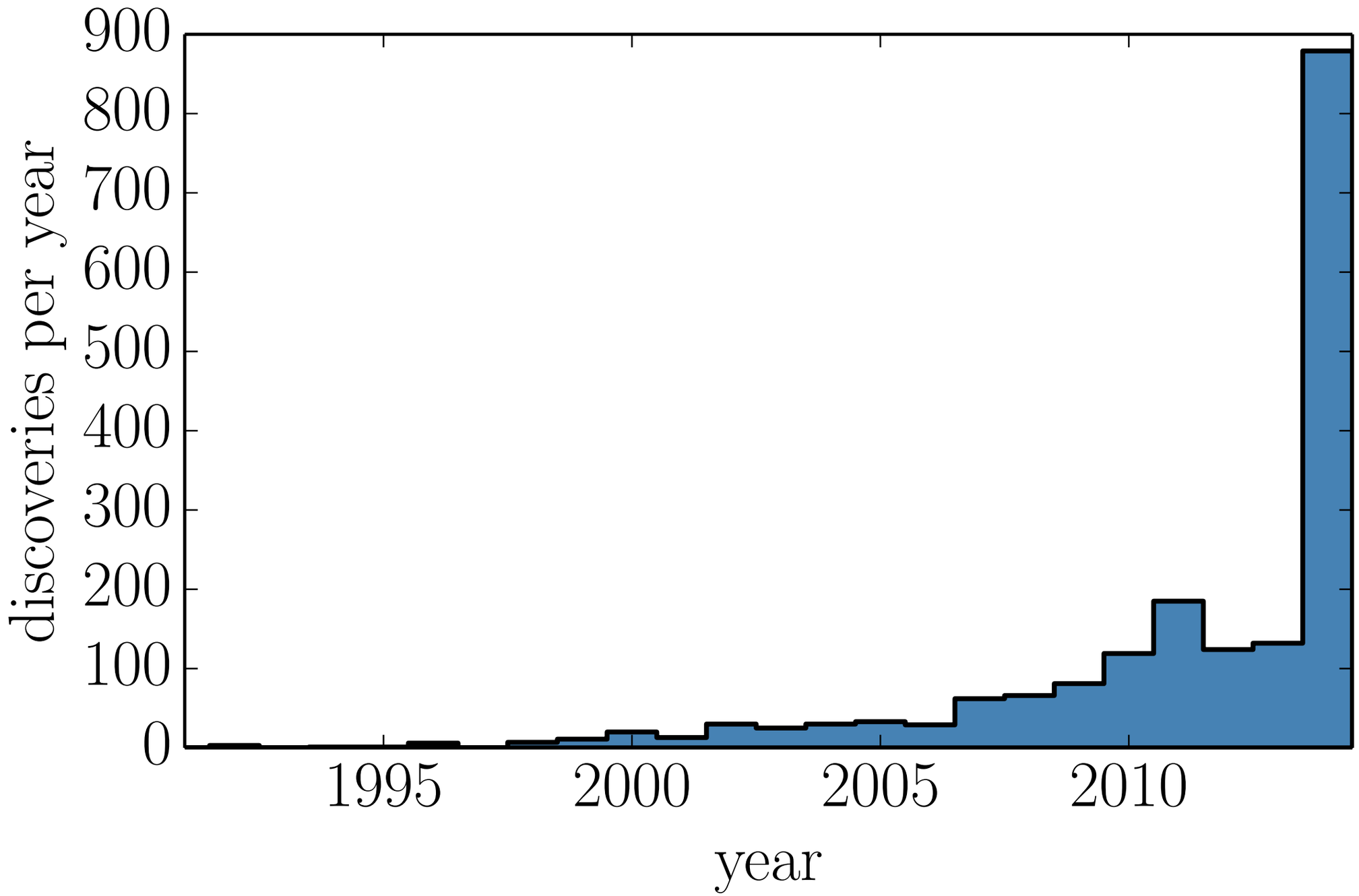
PLANETS AROUND SUN-LIKE STARS

PLANETS AROUND OTHER STARS

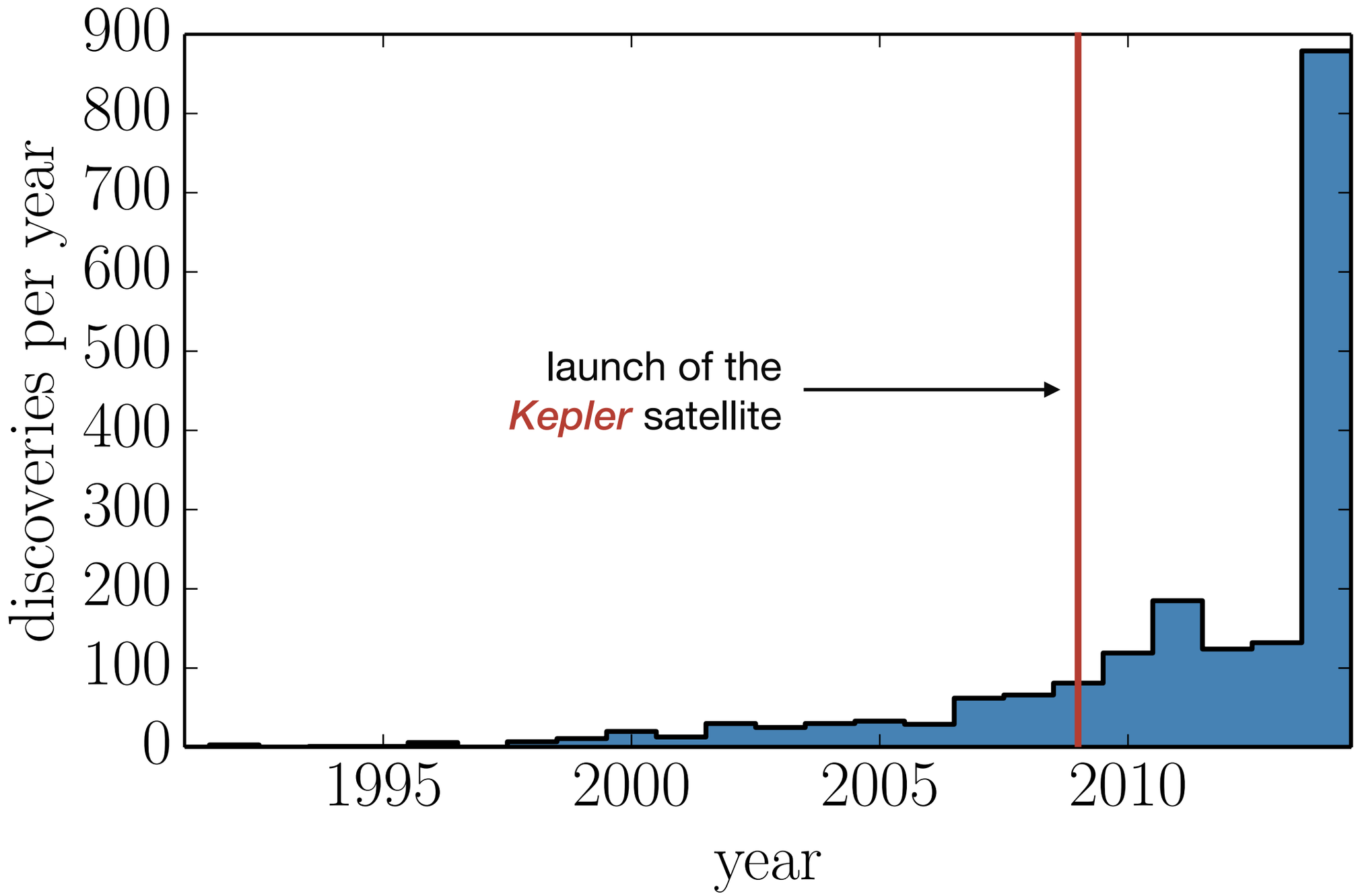
EARTH-SIZED PLANETS

EARTH

Image credit **xkcd.com/1298**



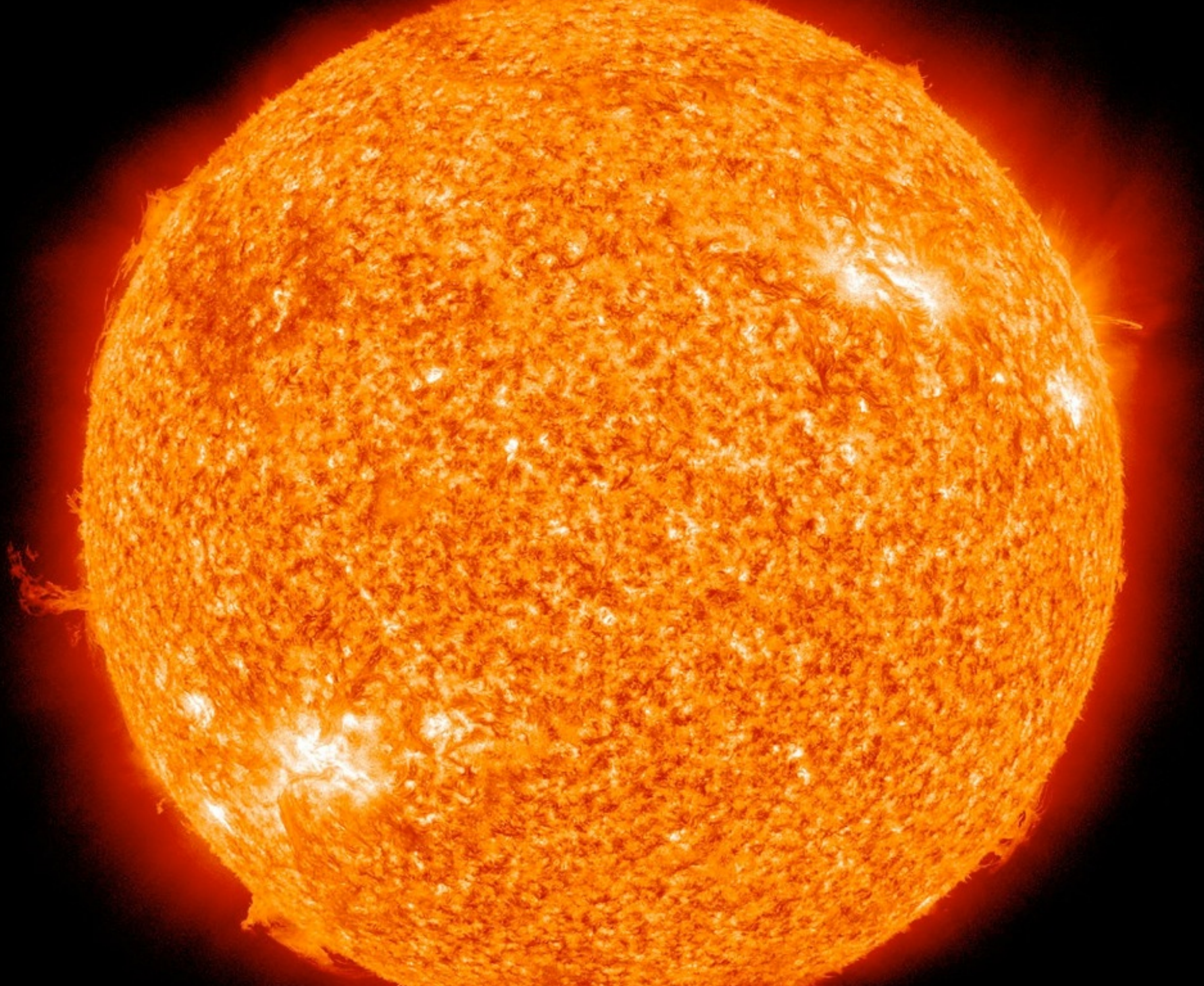
Data from **The Open Exoplanet Catalogue**

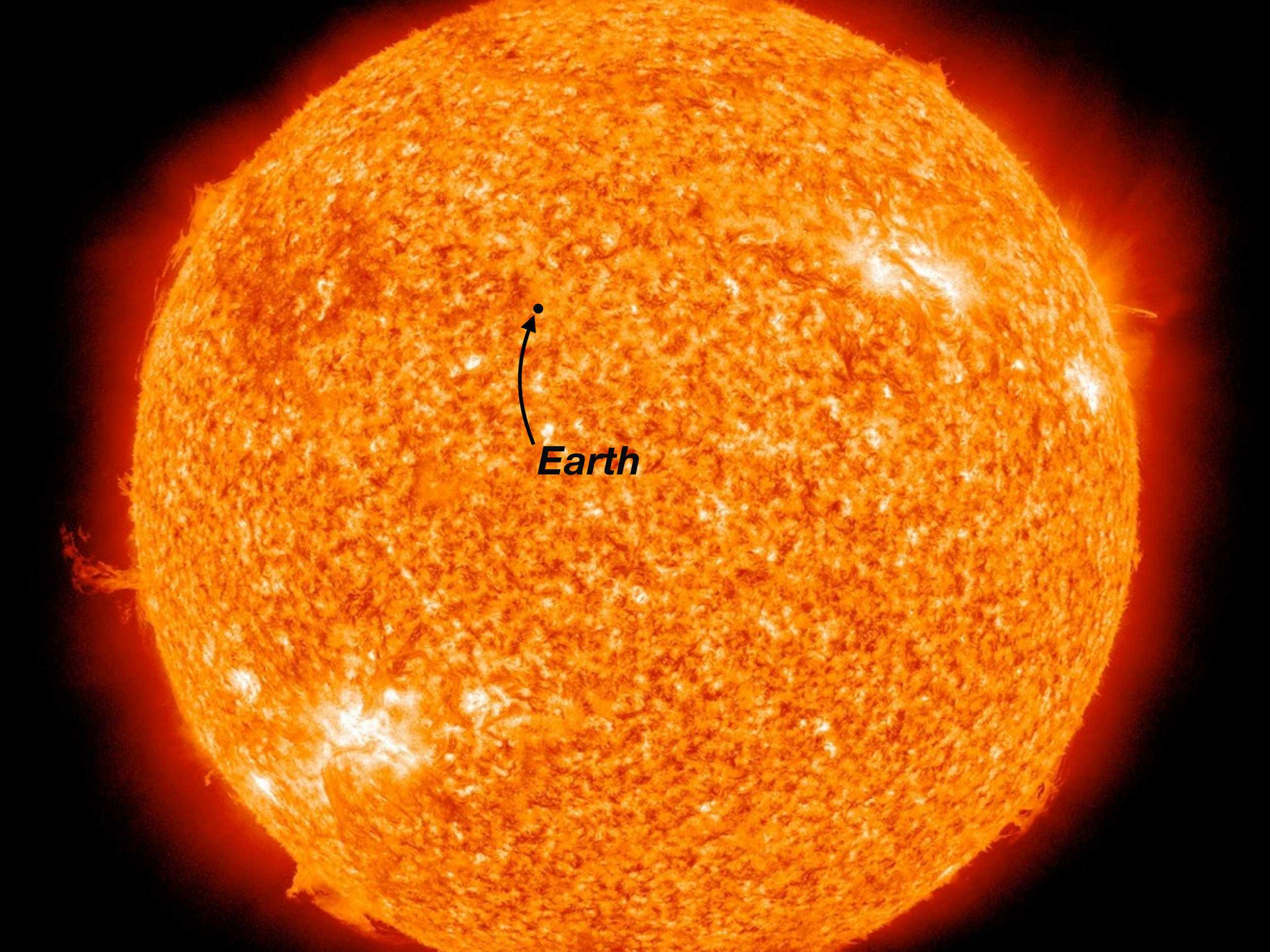


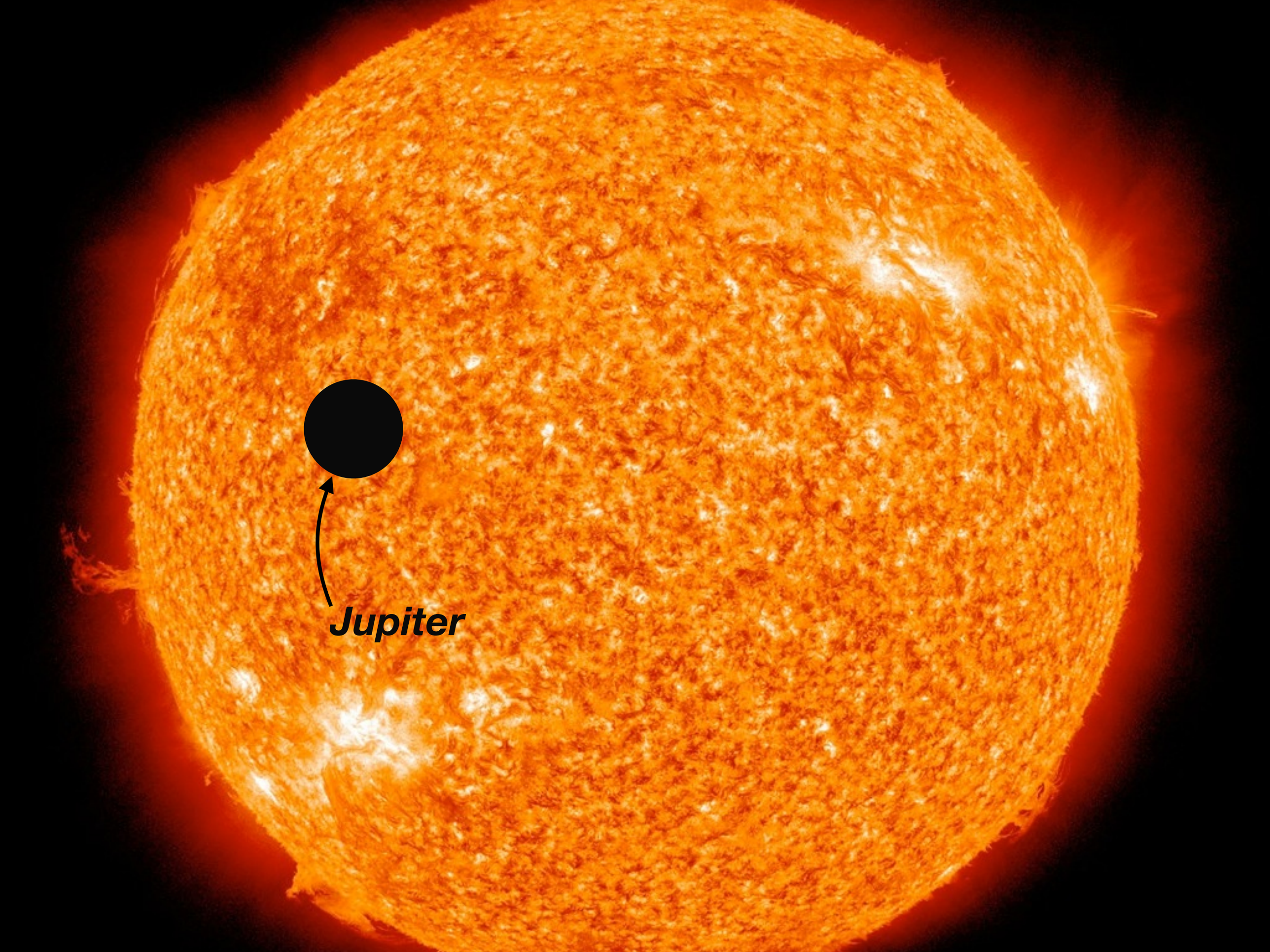
Data from **The Open Exoplanet Catalogue**

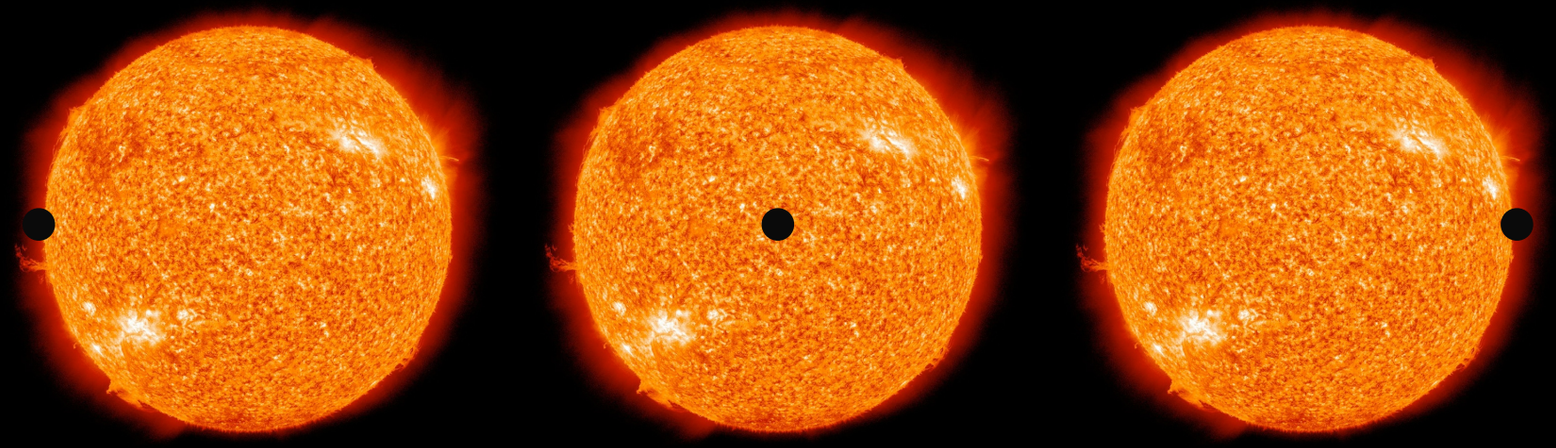
Exoplanets are **hard** to find

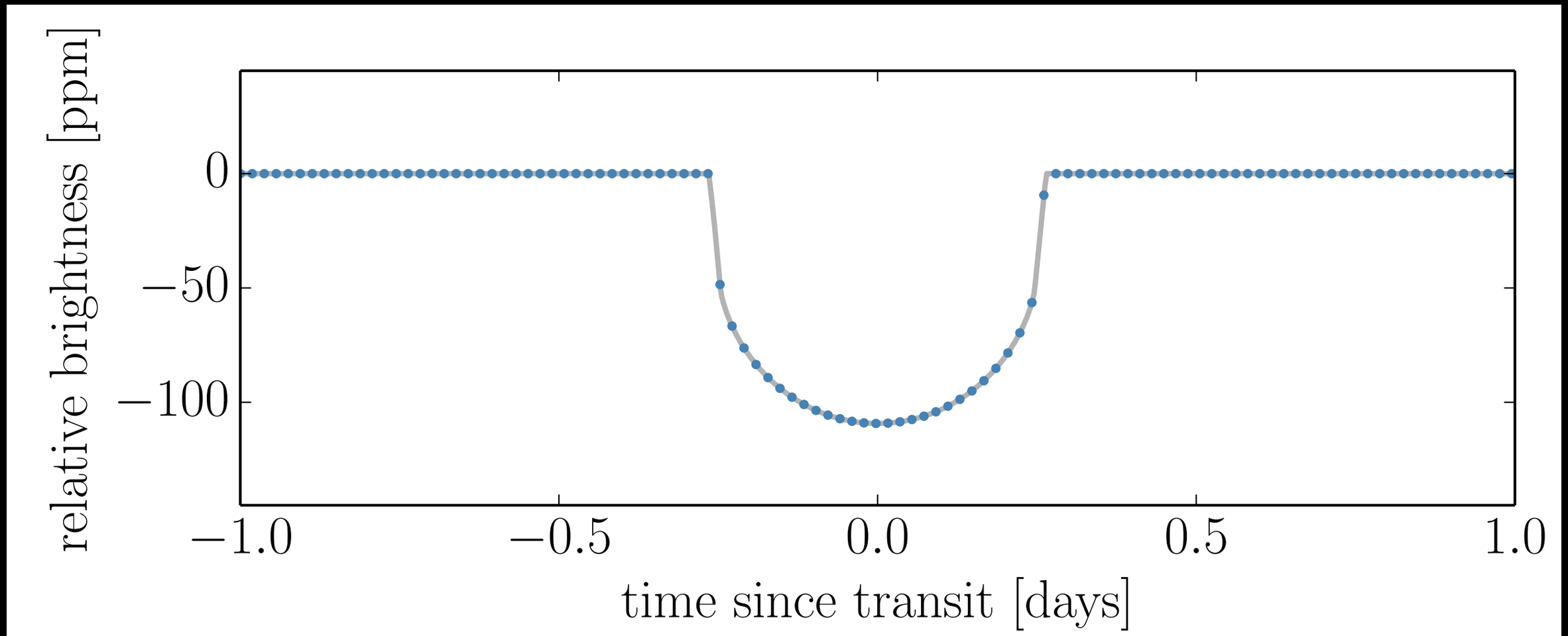
The **transit** method




Earth


Jupiter




relative brightness [ppm]
0
−50
−100
−1.0
−0.5
0.0
0.5
1.0
time since transit [days]

need to look at **the right place**
at **the right time**

and measure brightness
extremely precisely

Kepler

Image credit **NASA**

Photo credit: **NASA**

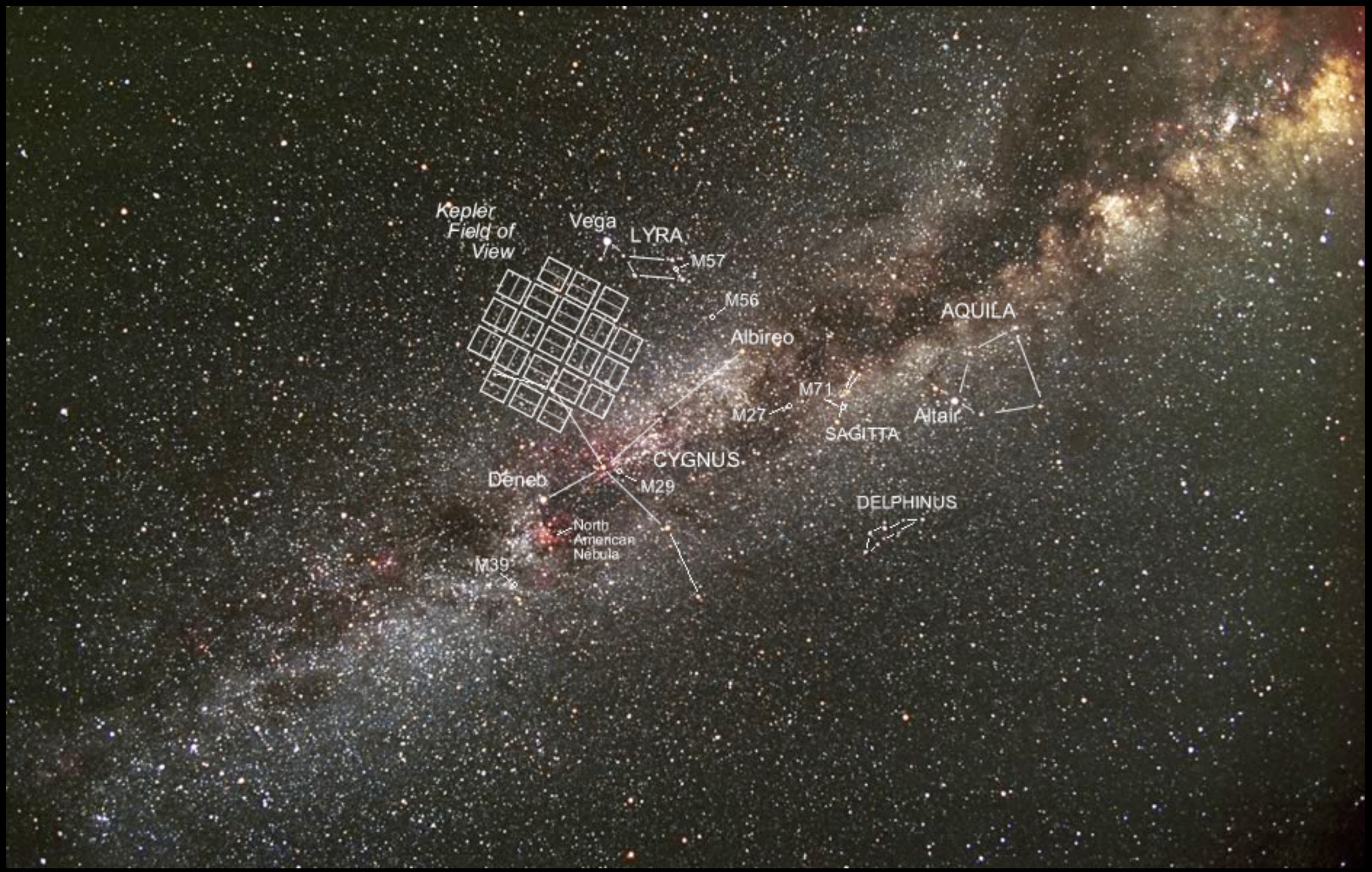


Image credit **Carter Roberts**

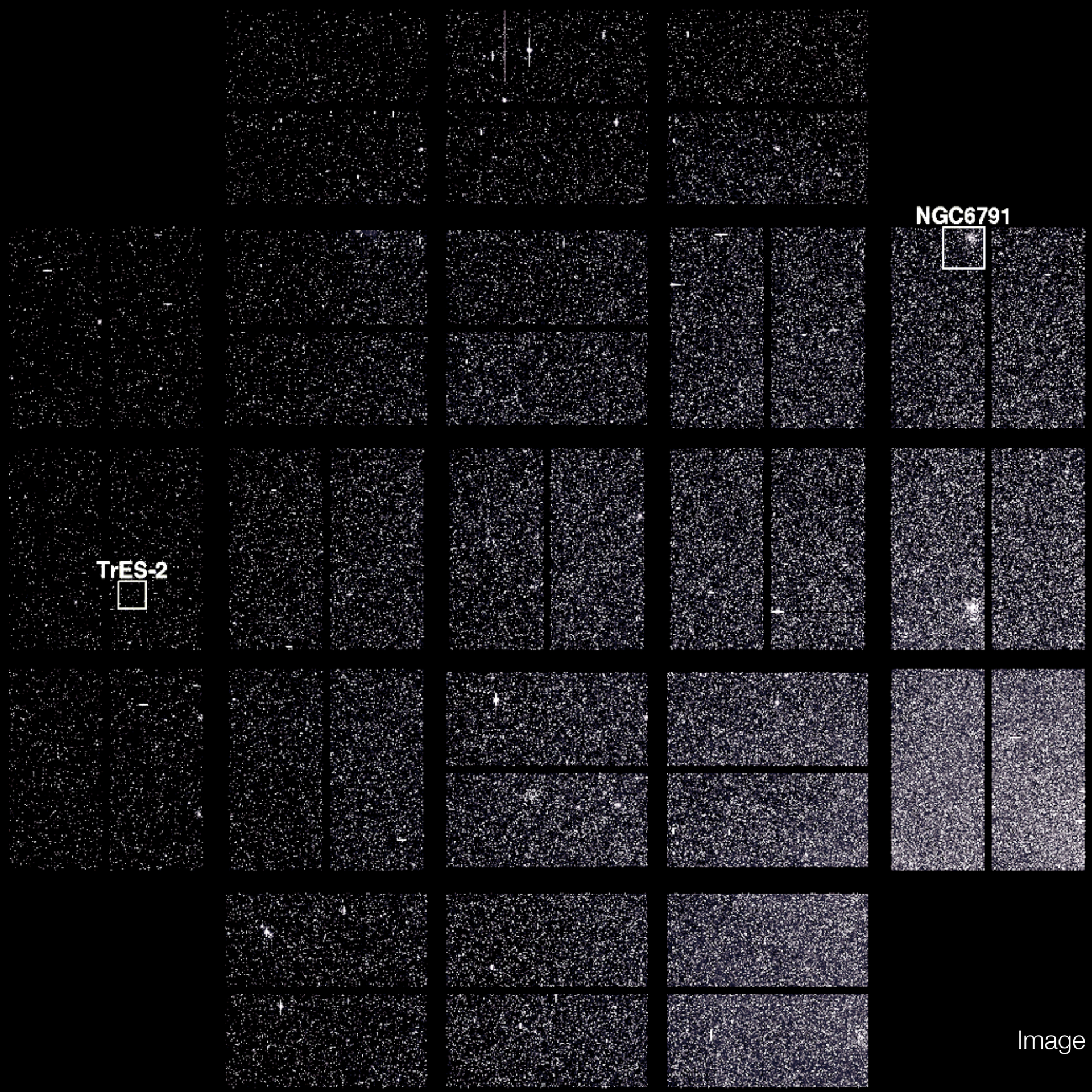
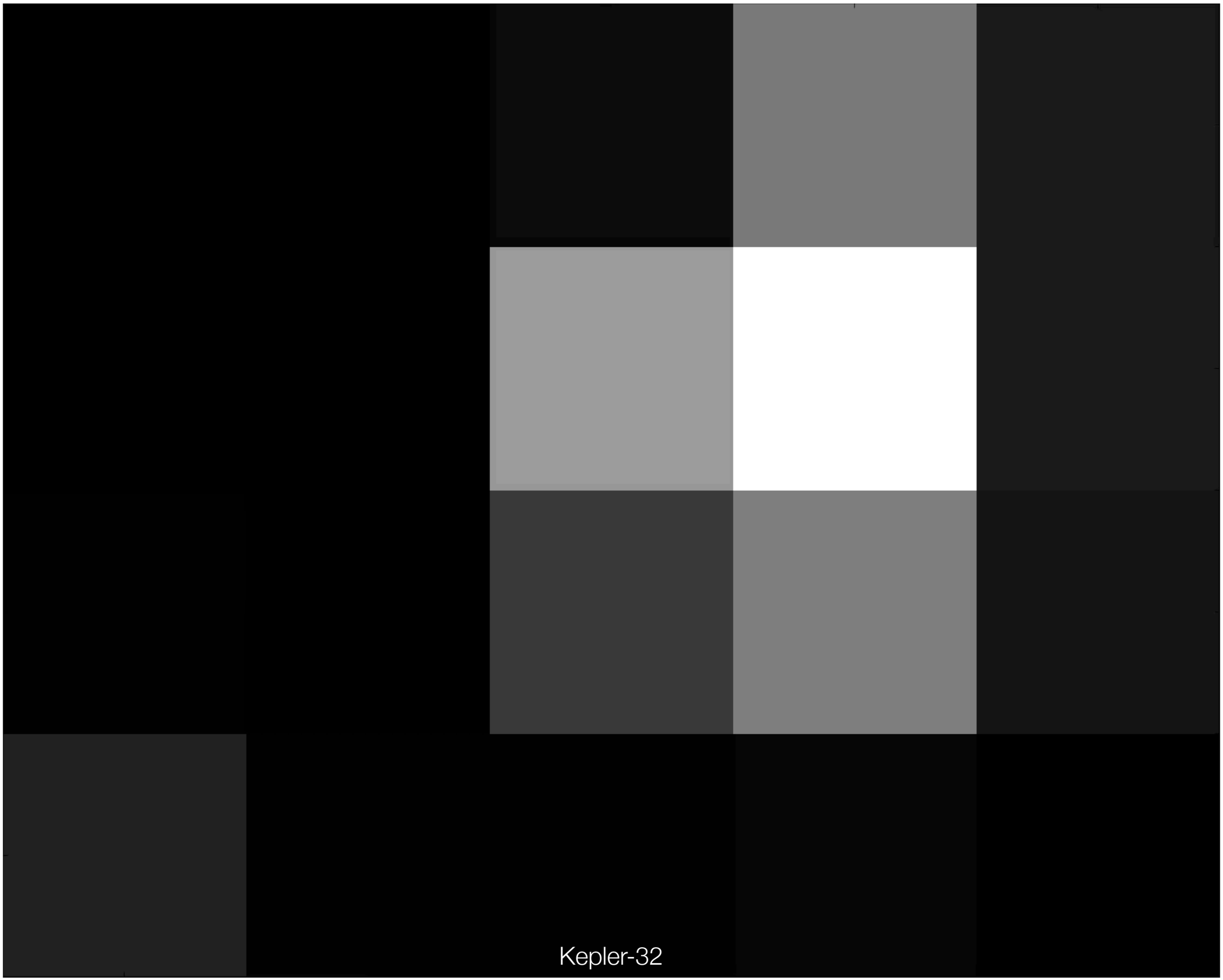
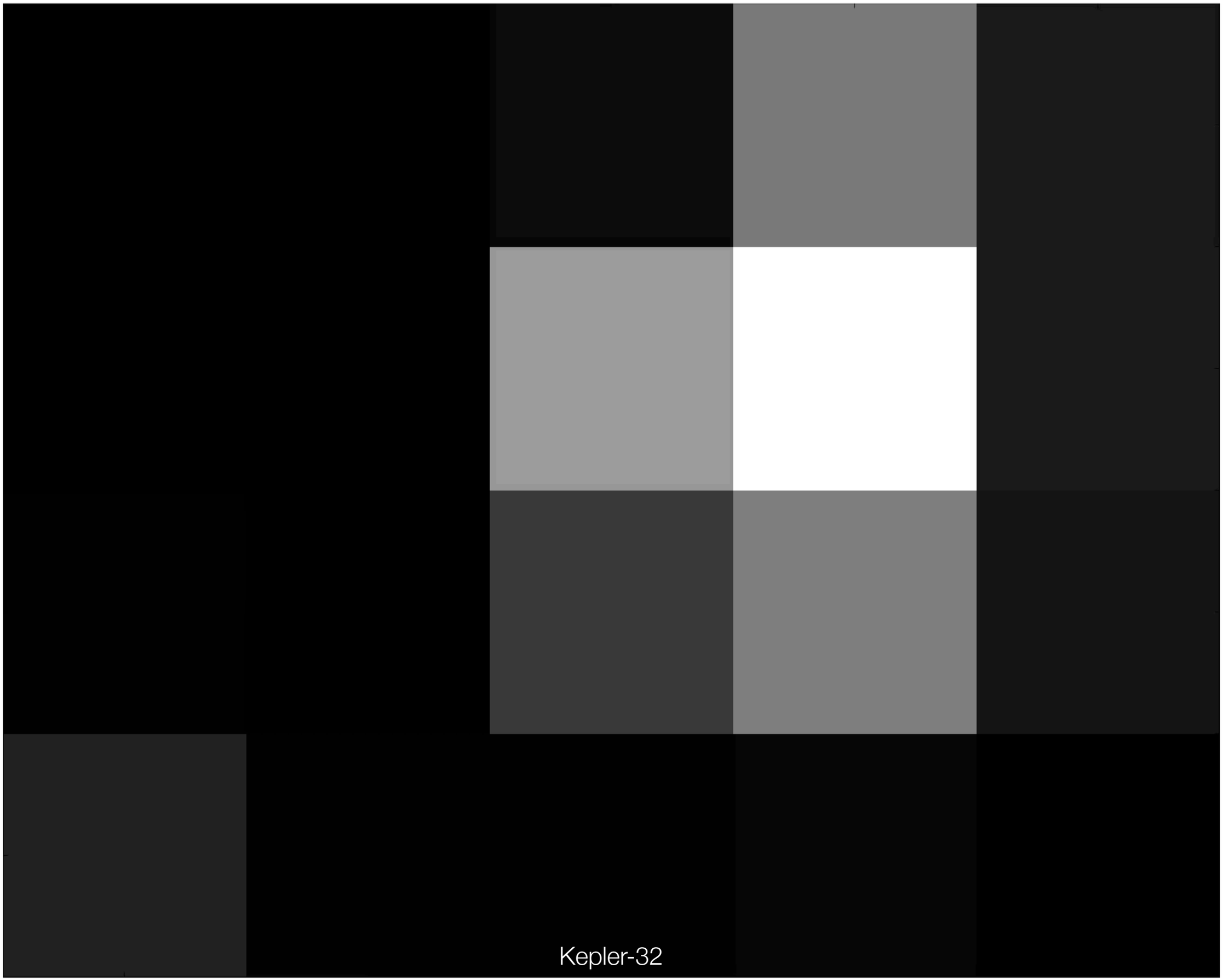
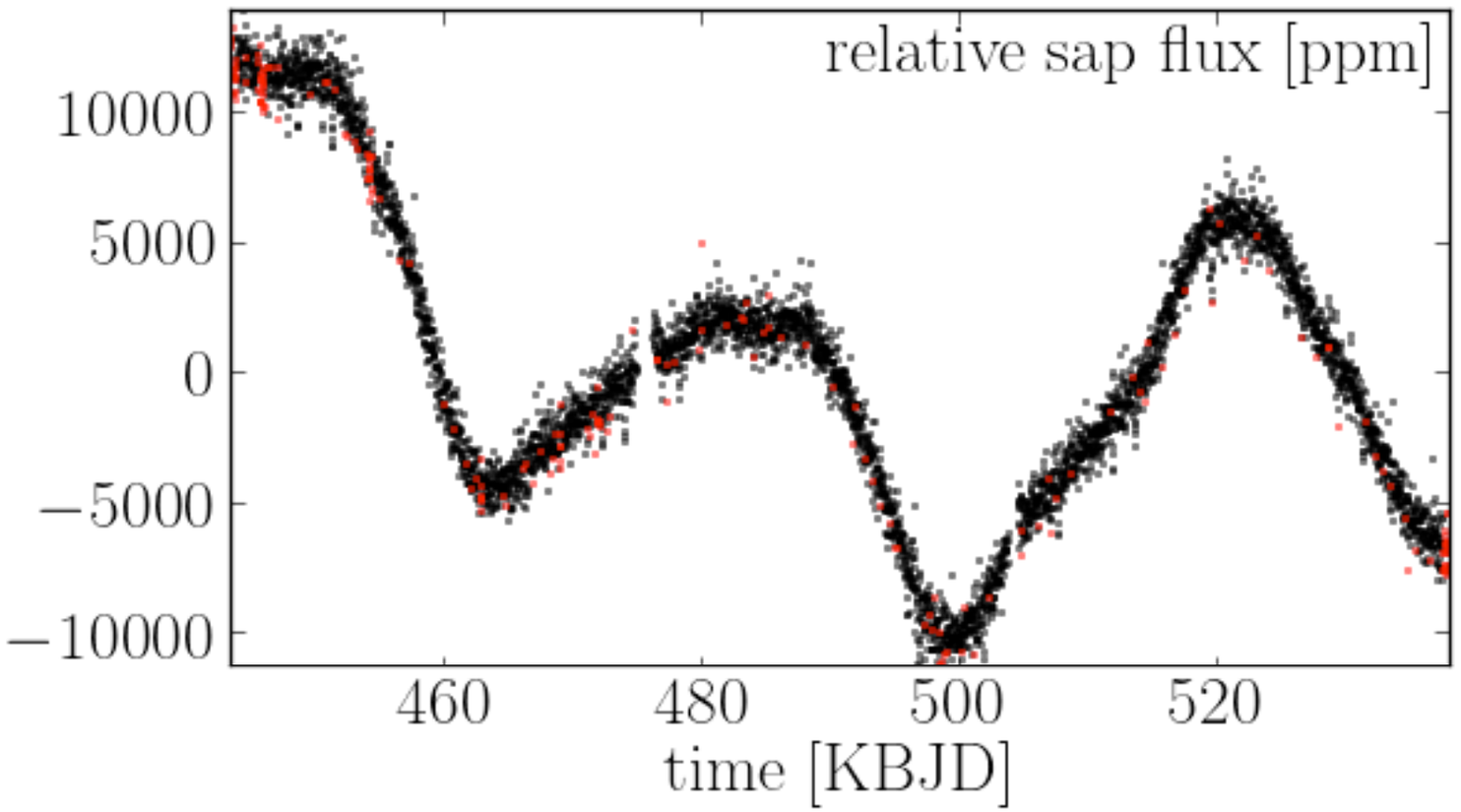


Image credit **NASA**

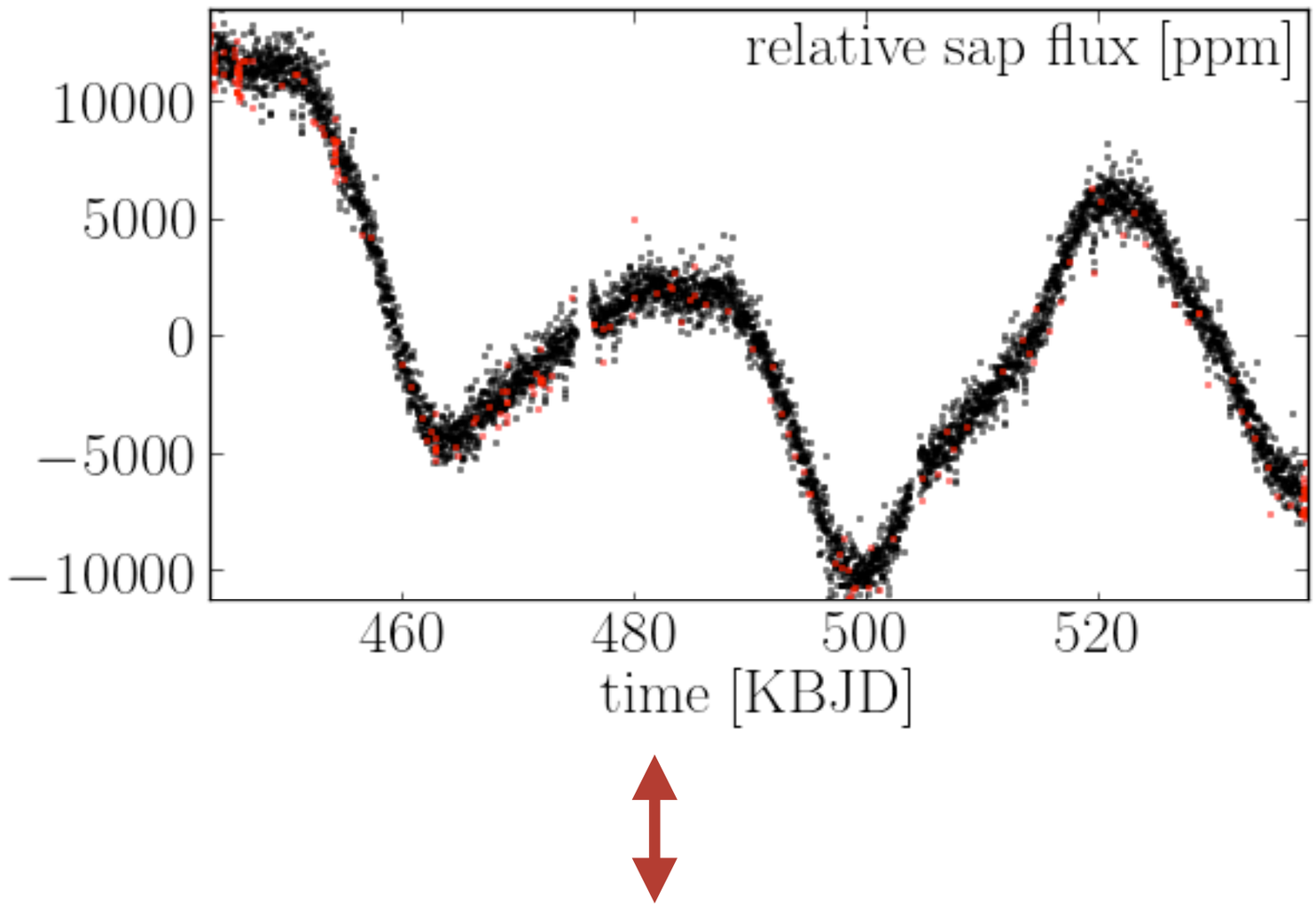

Kepler-32

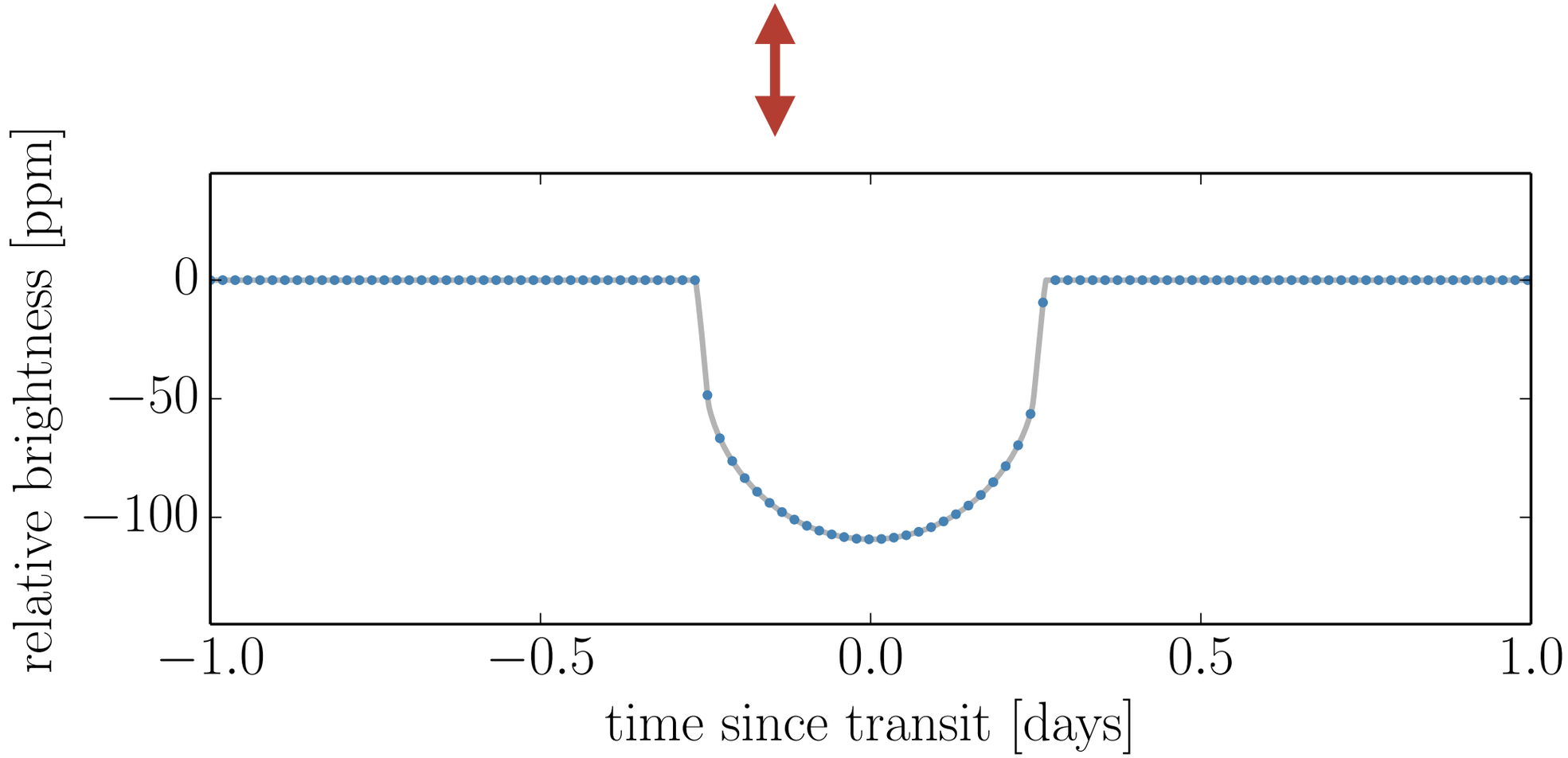

Kepler-32

Kepler-32

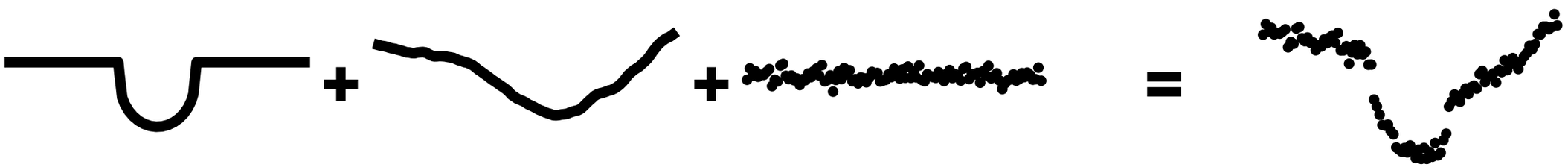

relative sap flux [ppm]
10000
5000
0
−5000
−10000
460
480
500
520
time [KBJD]

Kepler-32


relative sap flux [ppm]
10000
5000
0
−5000
−10000
460
480
500
520
time [KBJD]

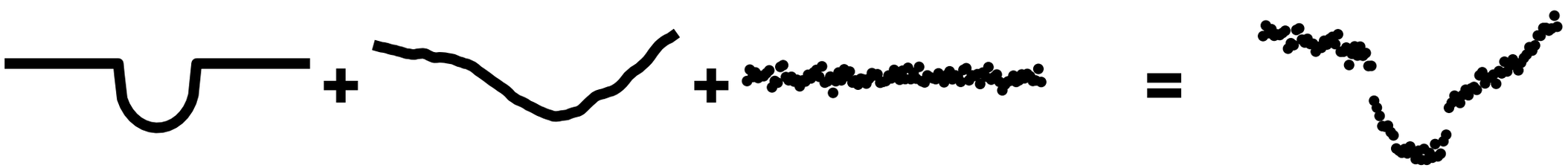

relative brightness [ppm]
0
−50
−100
−1.0
−0.5
0.0
0.5
1.0
time since transit [days]

The anatomy of a transit observation



signal variability noise data

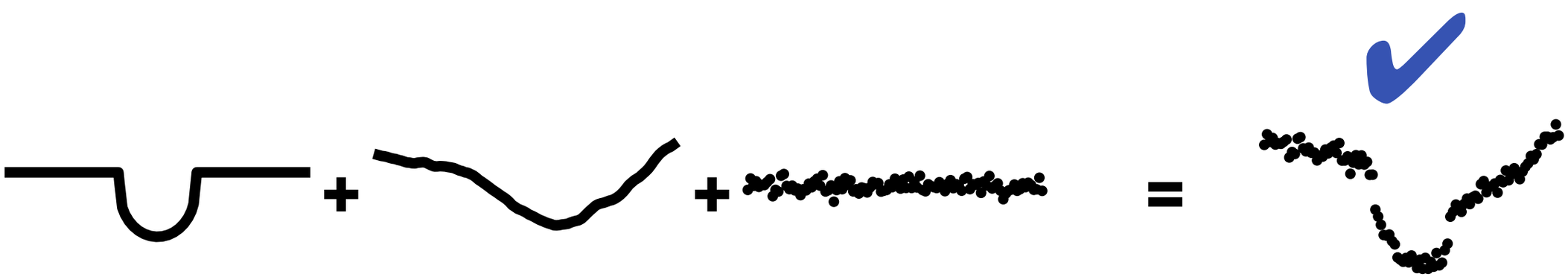
The anatomy of a transit observation



signal variability noise data

astrophysics and **spacecraft**

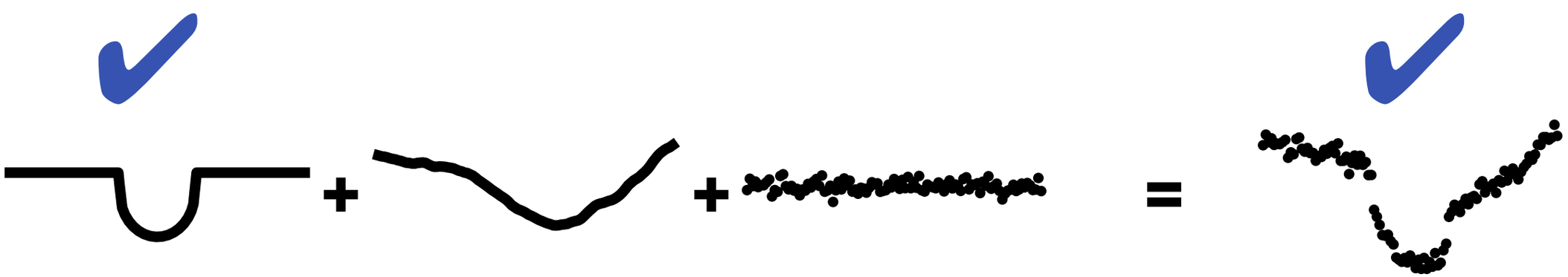
The anatomy of a transit observation



signal variability noise data

astrophysics and **spacecraft**

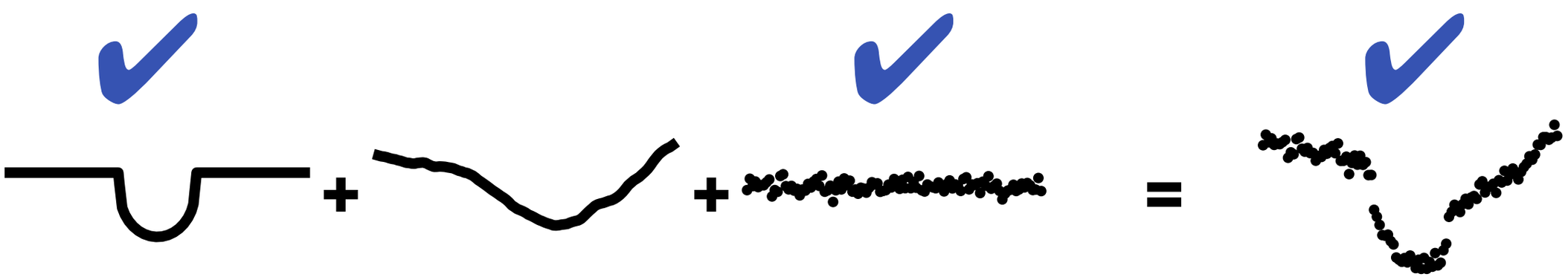
The anatomy of a transit observation



signal variability noise data

astrophysics and **spacecraft**

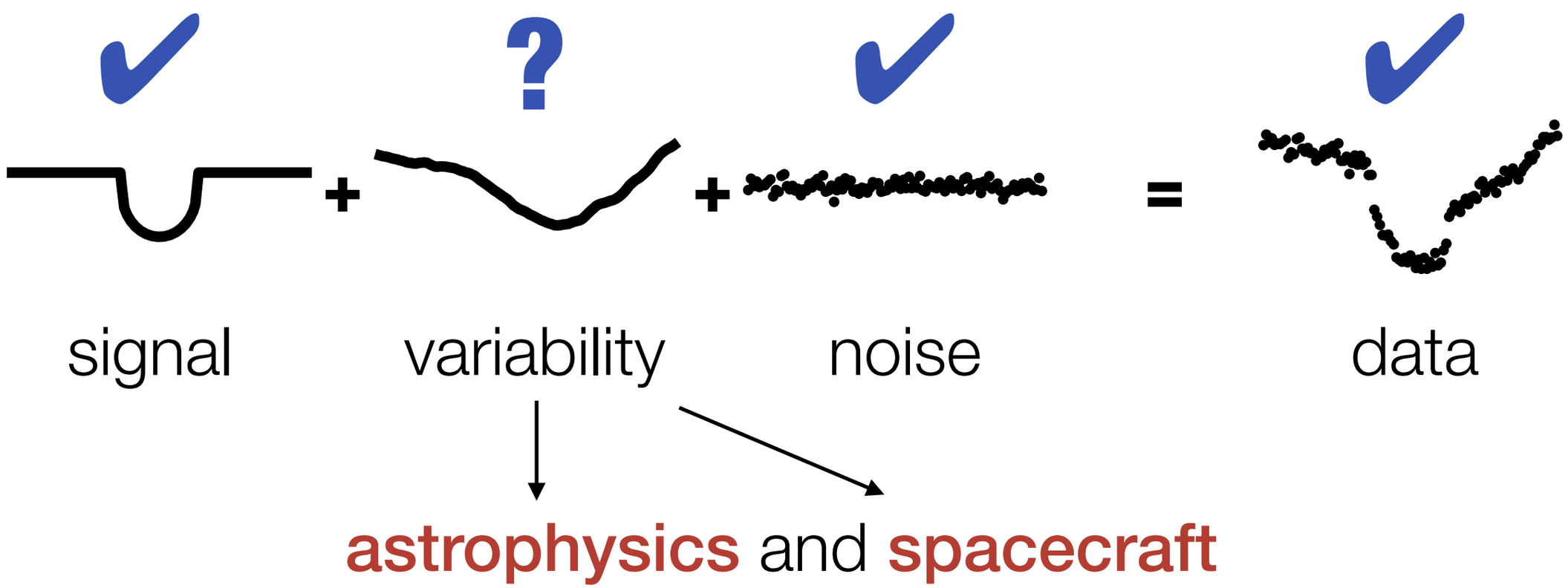
The anatomy of a transit observation



signal variability noise data

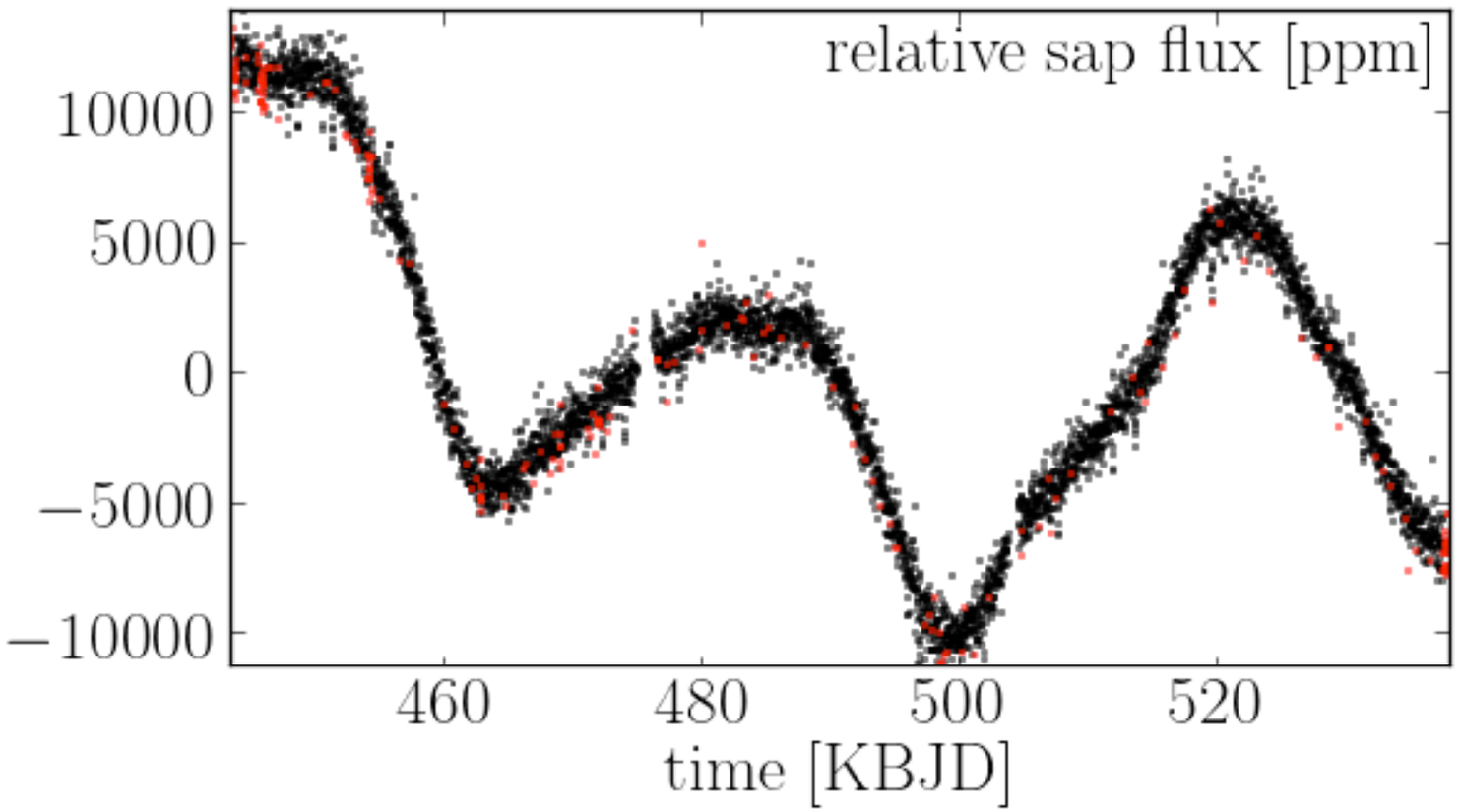
astrophysics and **spacecraft**

The anatomy of a transit observation

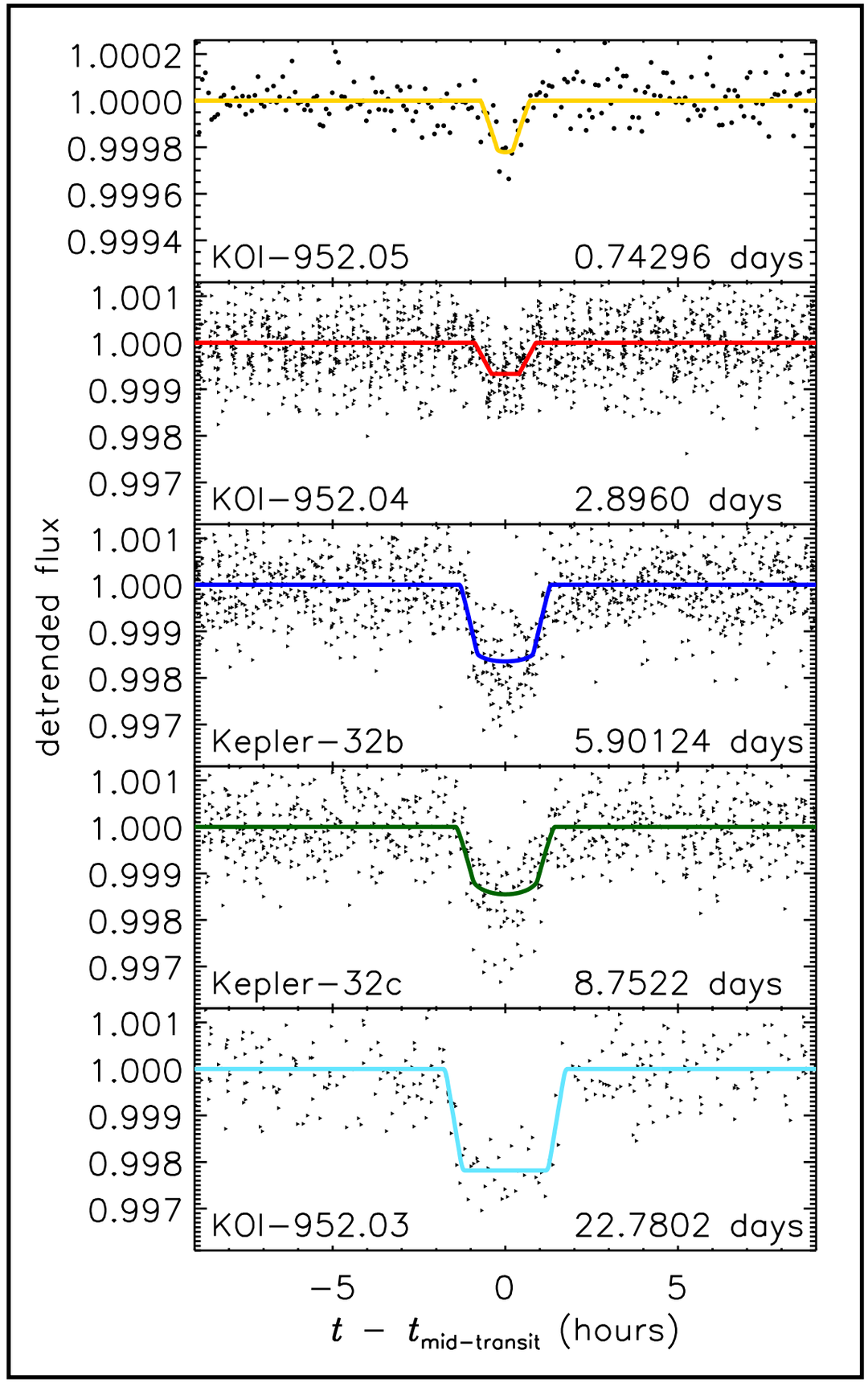


so.
how do we clean up the crap?

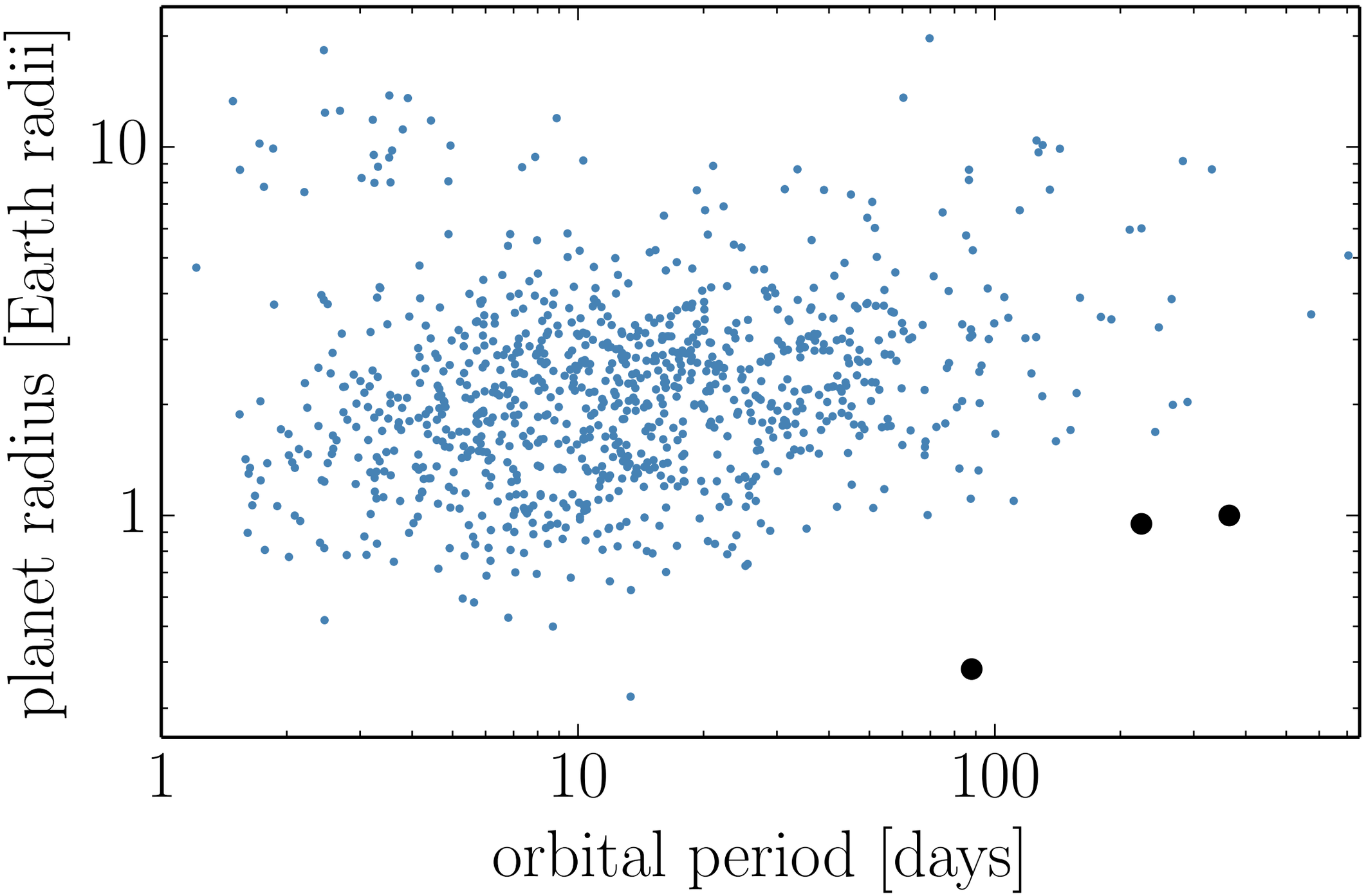
Standard practice: **Filtering**


relative sap flux [ppm]
10000
5000
0
−5000
−10000
460
480
500
520
time [KBJD]

Kepler-32



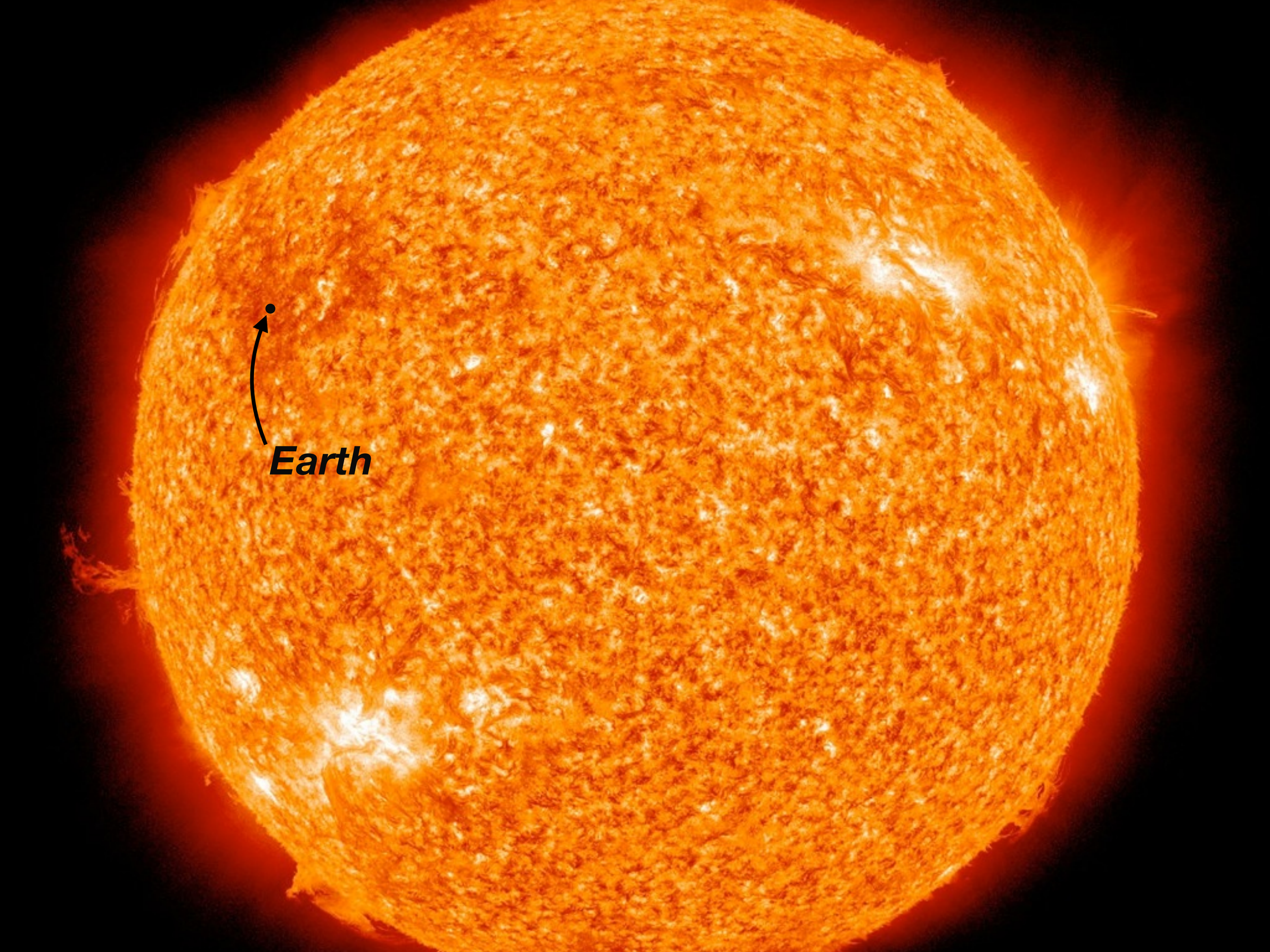
Credit **Fabrycky *et al.* (2012)**



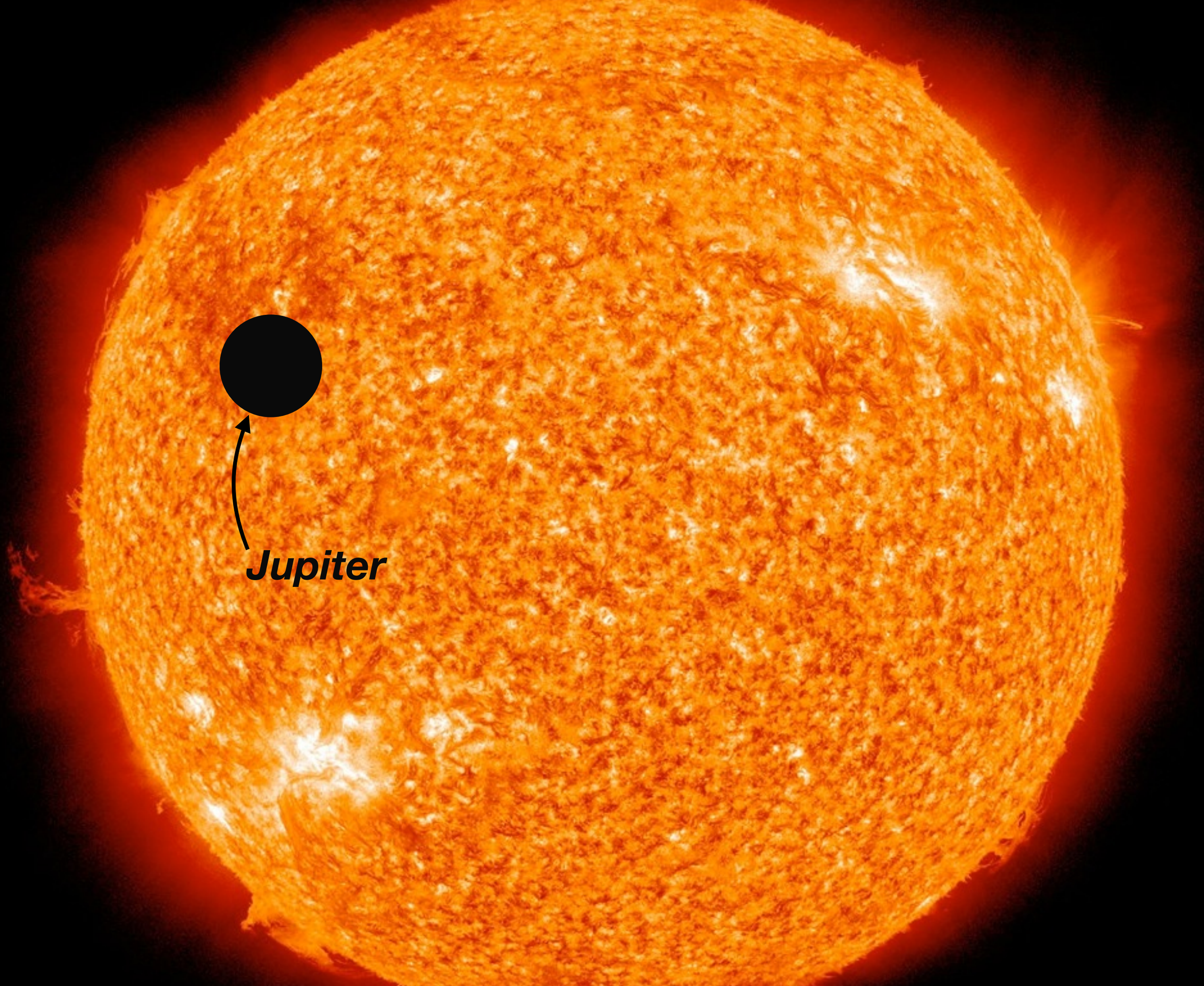
Data from **NASA Exoplanet Archive (11/22)**

Exoplanets are **hard** to find

Small exoplanets are **harder** to find


Earth

Exoplanets on long orbits are **even harder** to find


Jupiter

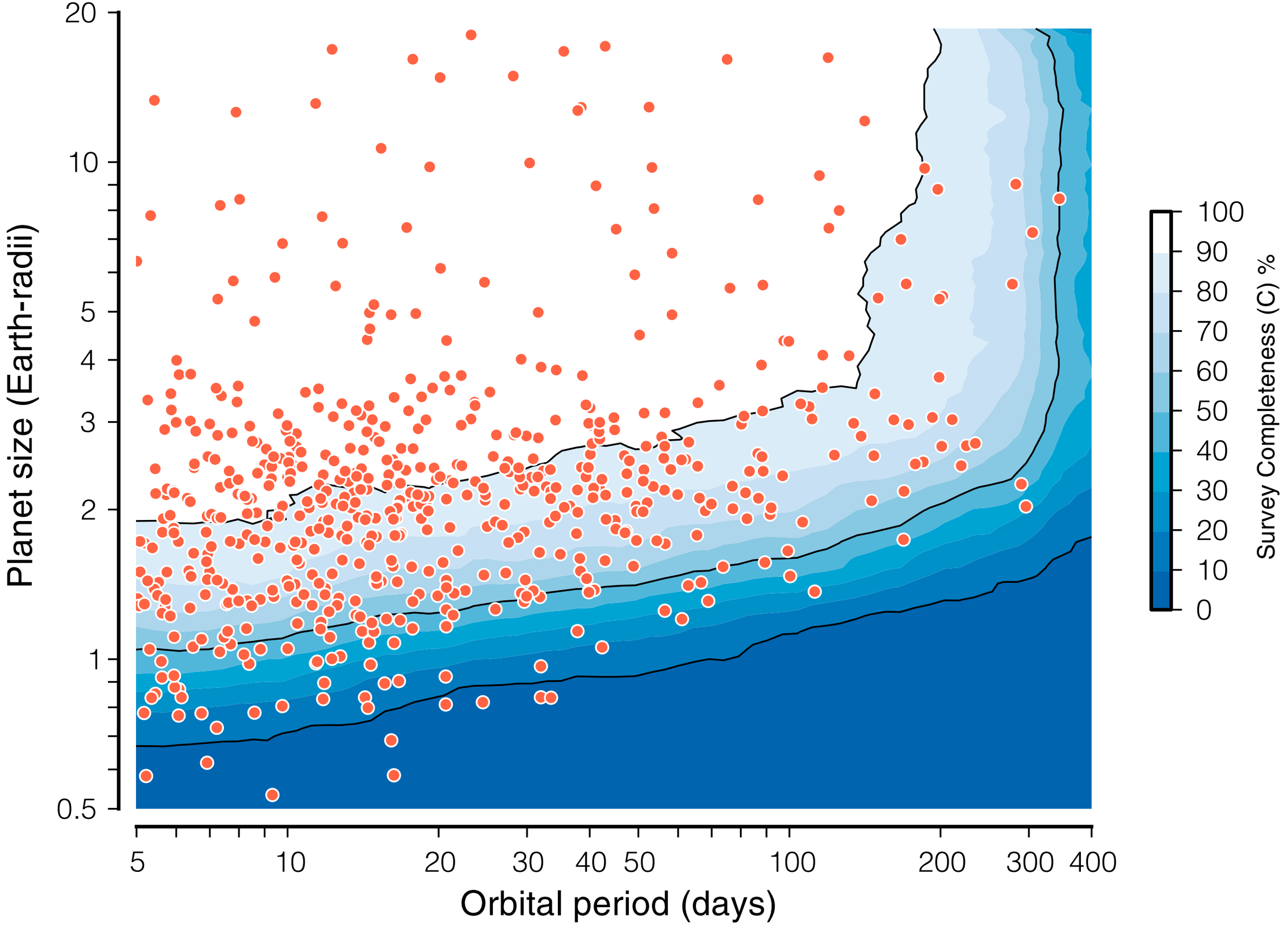


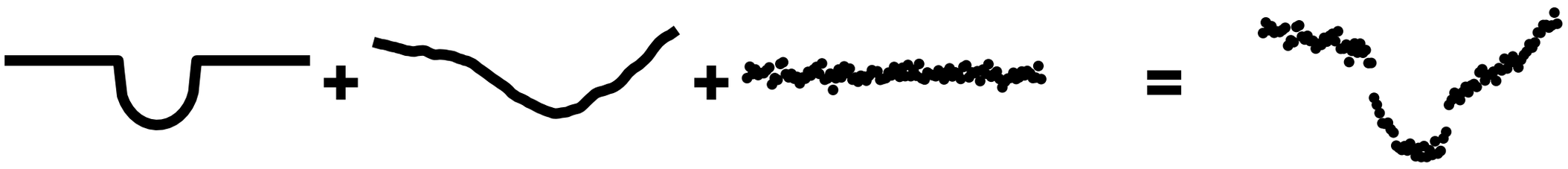
Figure credit: **Petigura, Howard & Marcy (2013)**

What about Gaussian Processes?

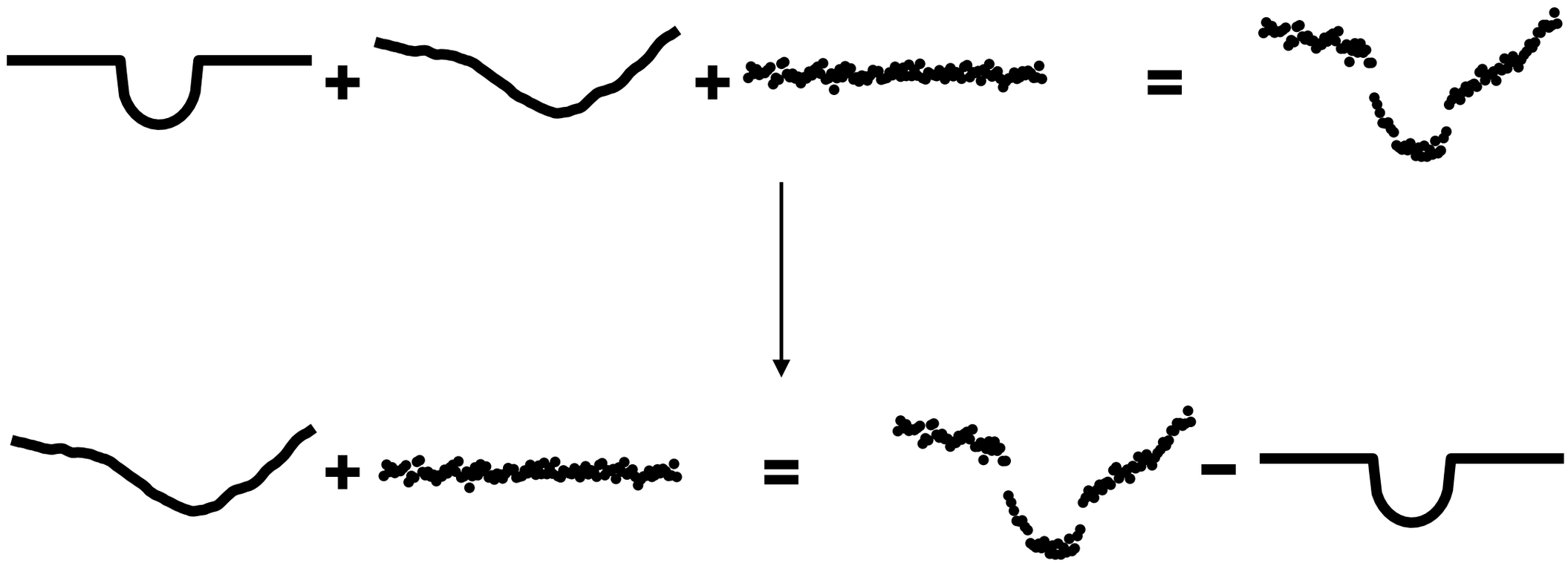
gaussianprocess.org/gpml

Rasmussen & Williams

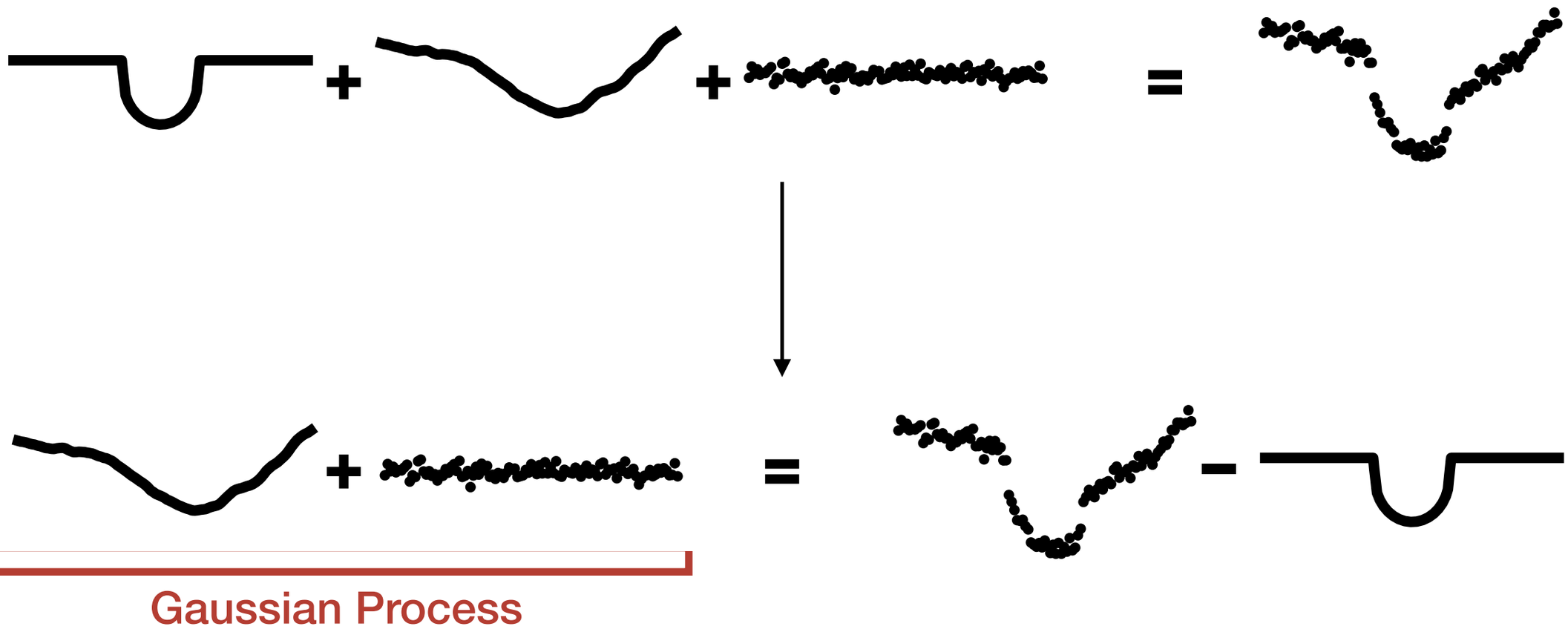
Modeling a light curve using a **Gaussian Processes**



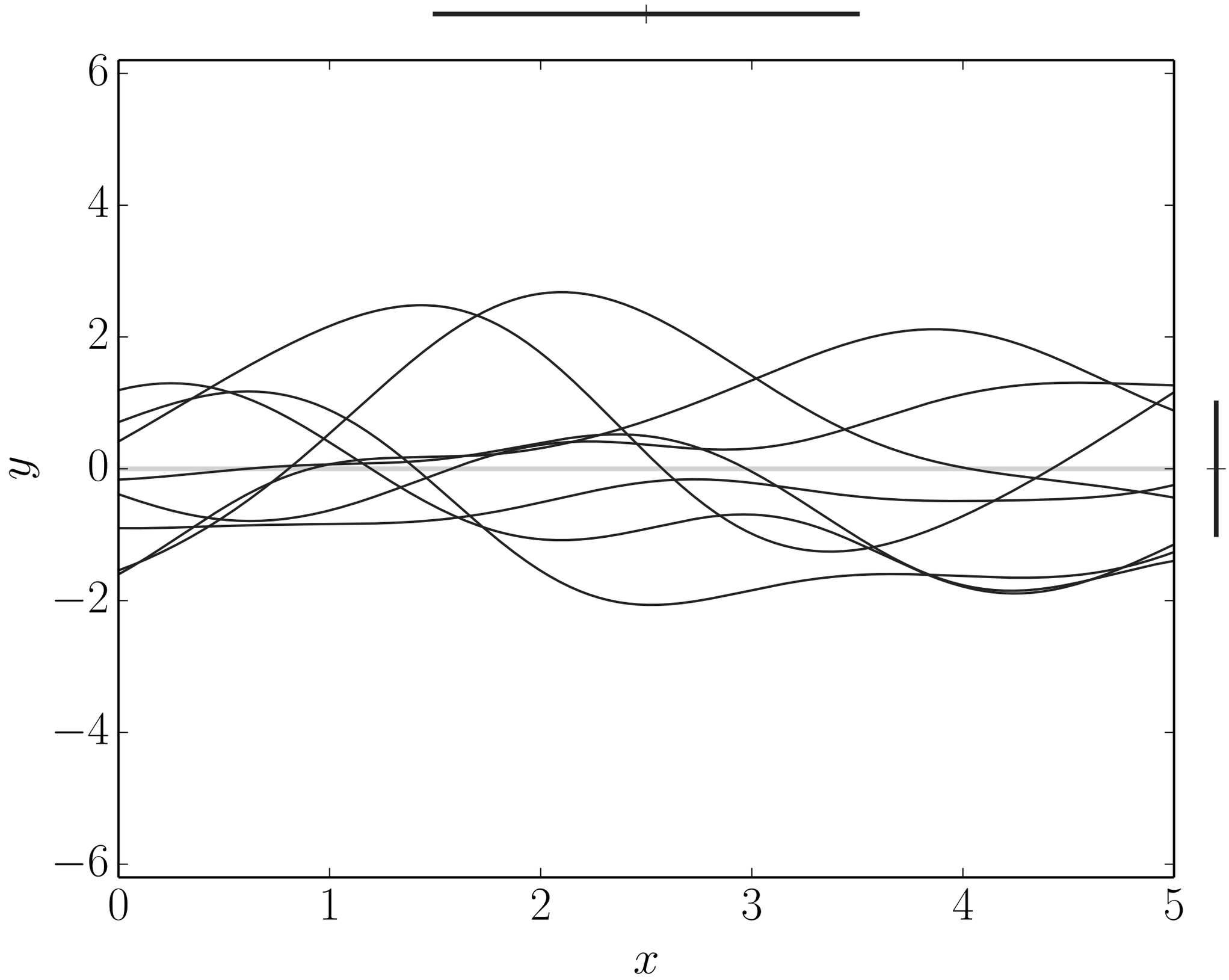
Modeling a light curve using a **Gaussian Processes**

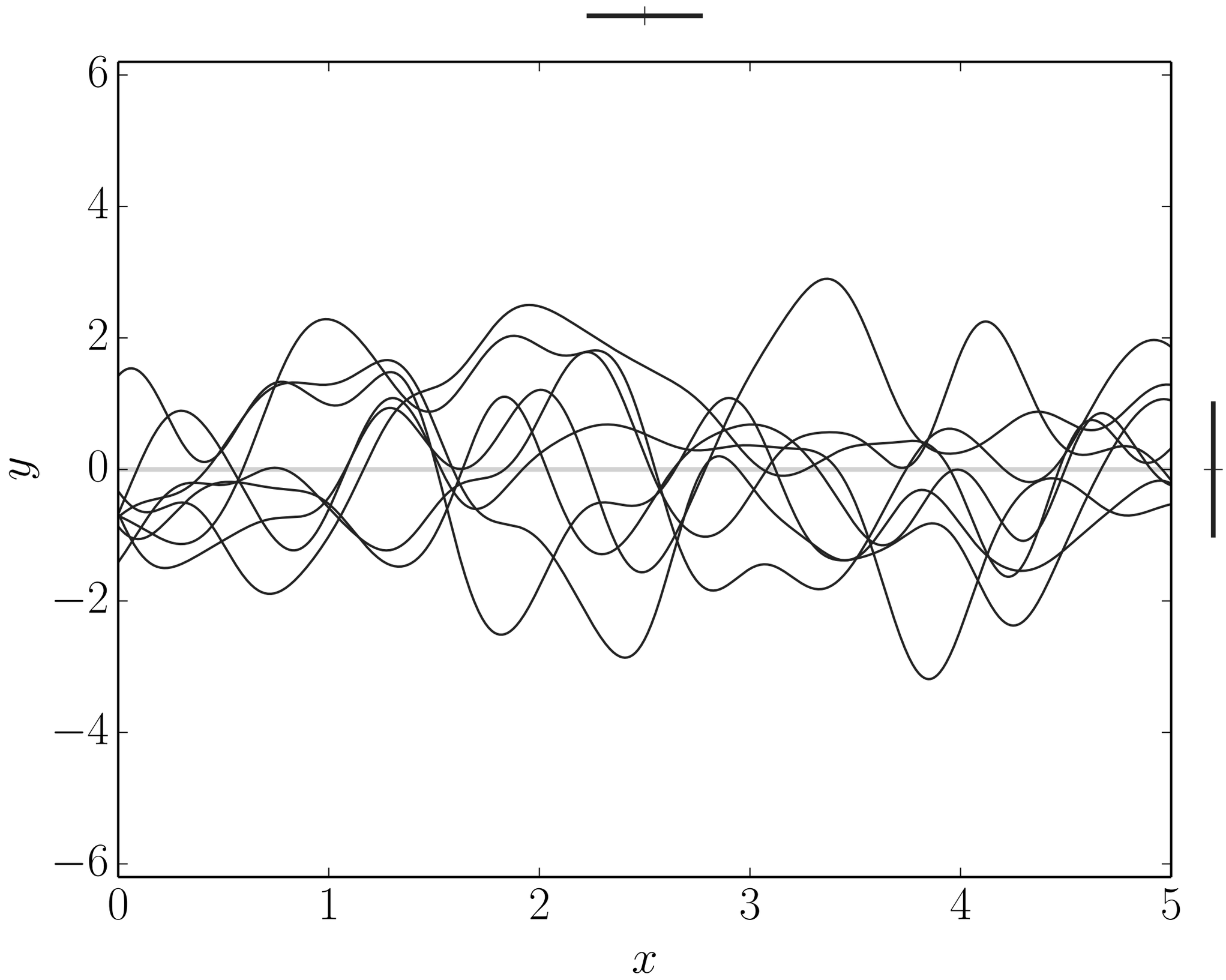


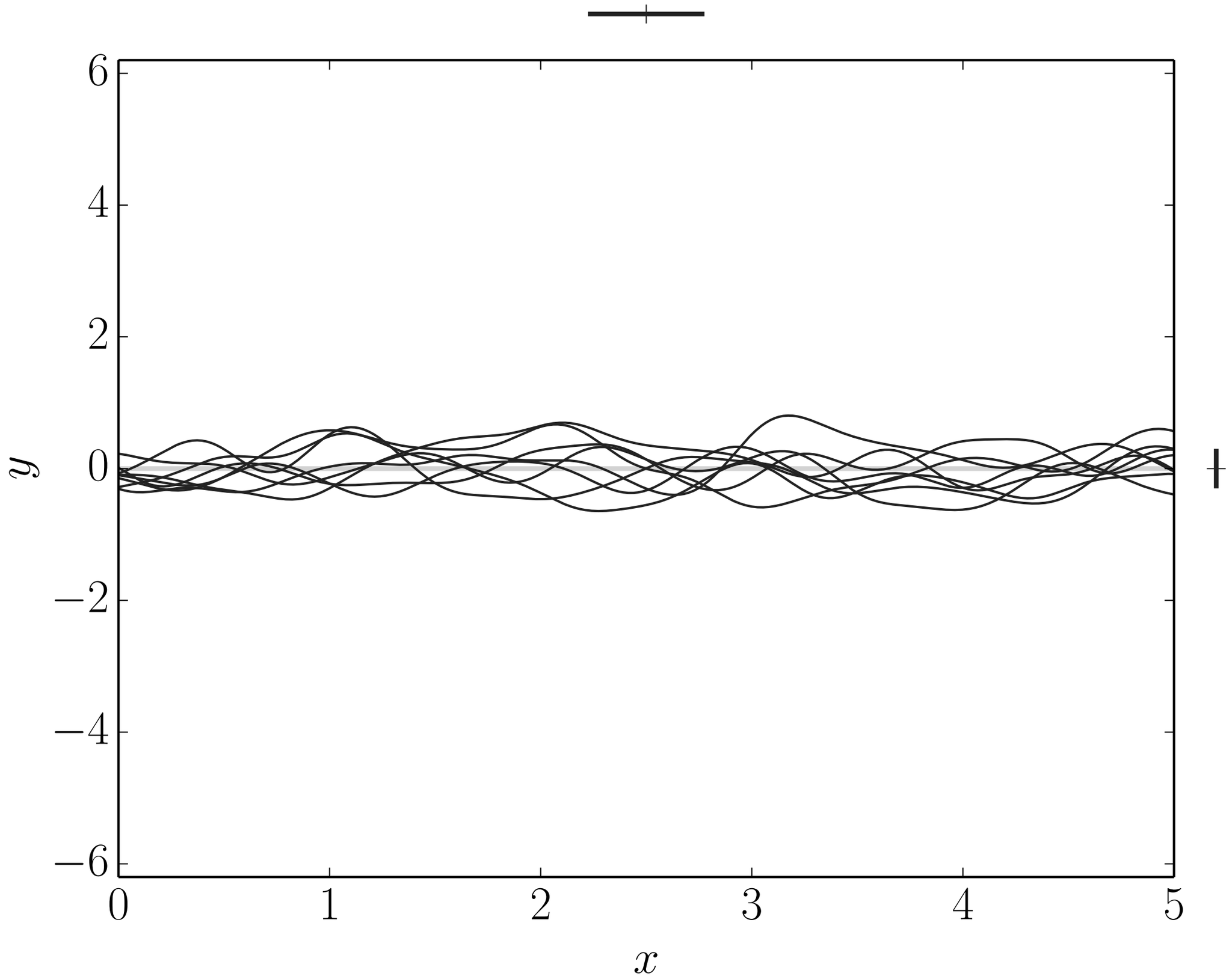
Modeling a light curve using a **Gaussian Processes**

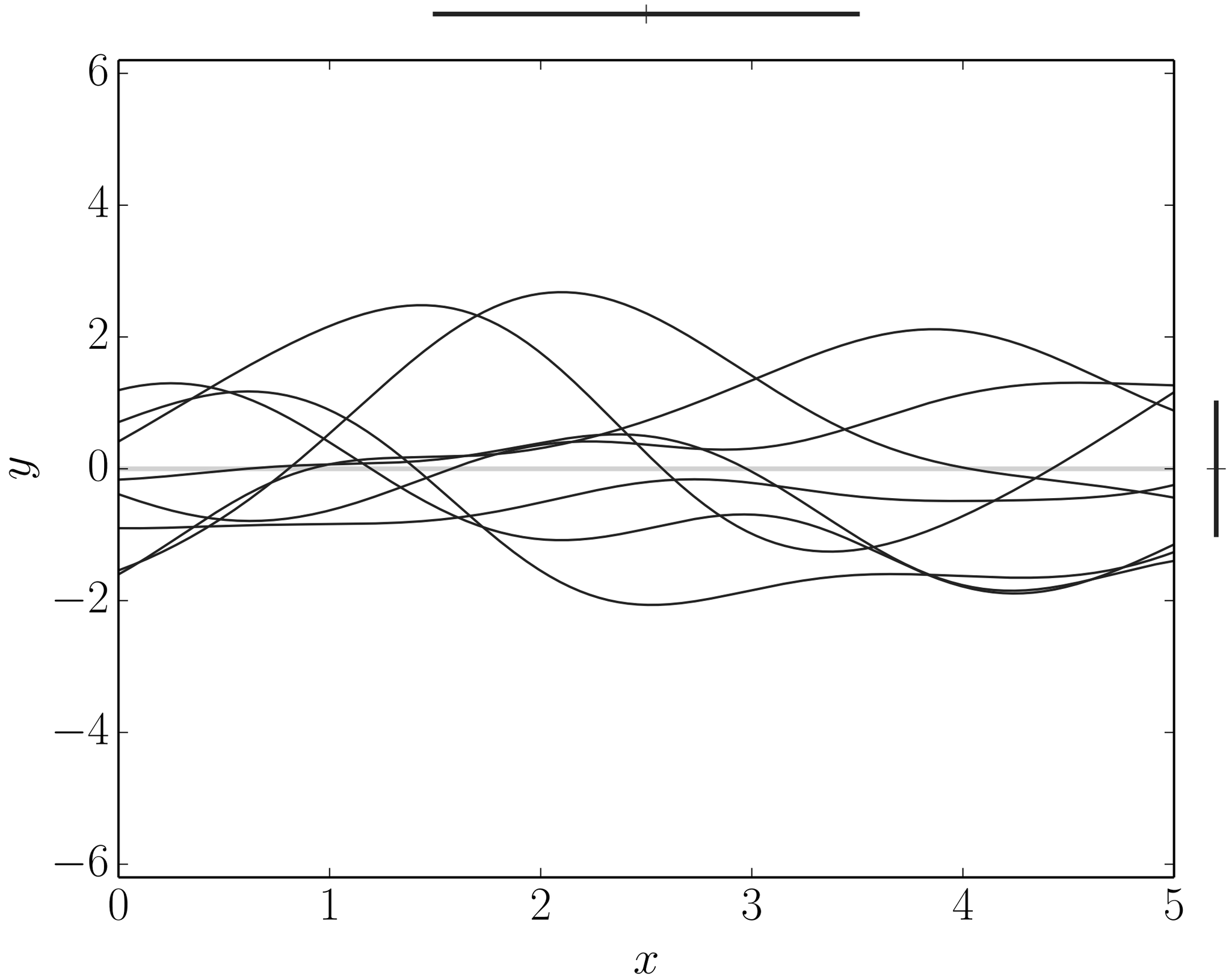


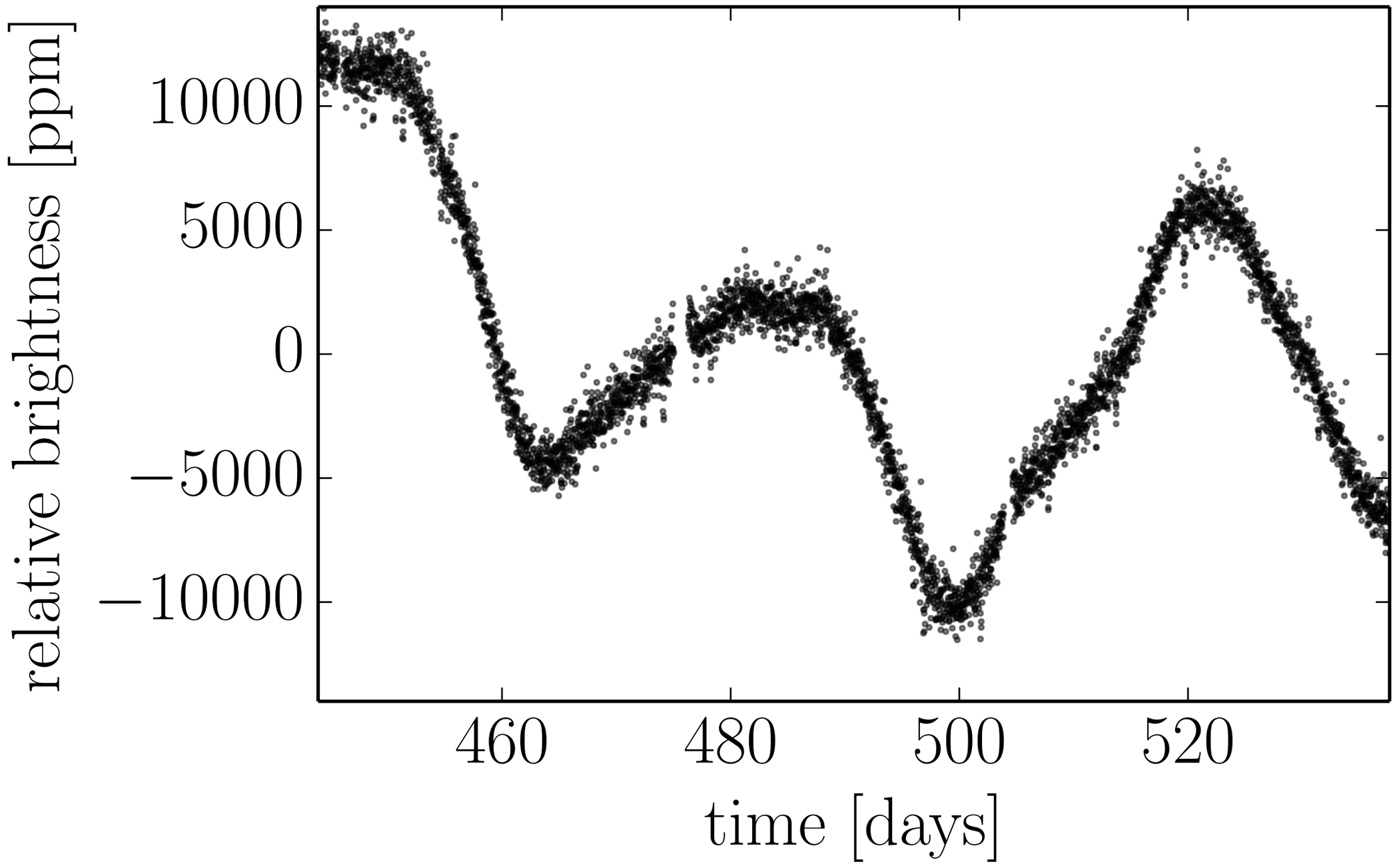
What is a Gaussian Process?

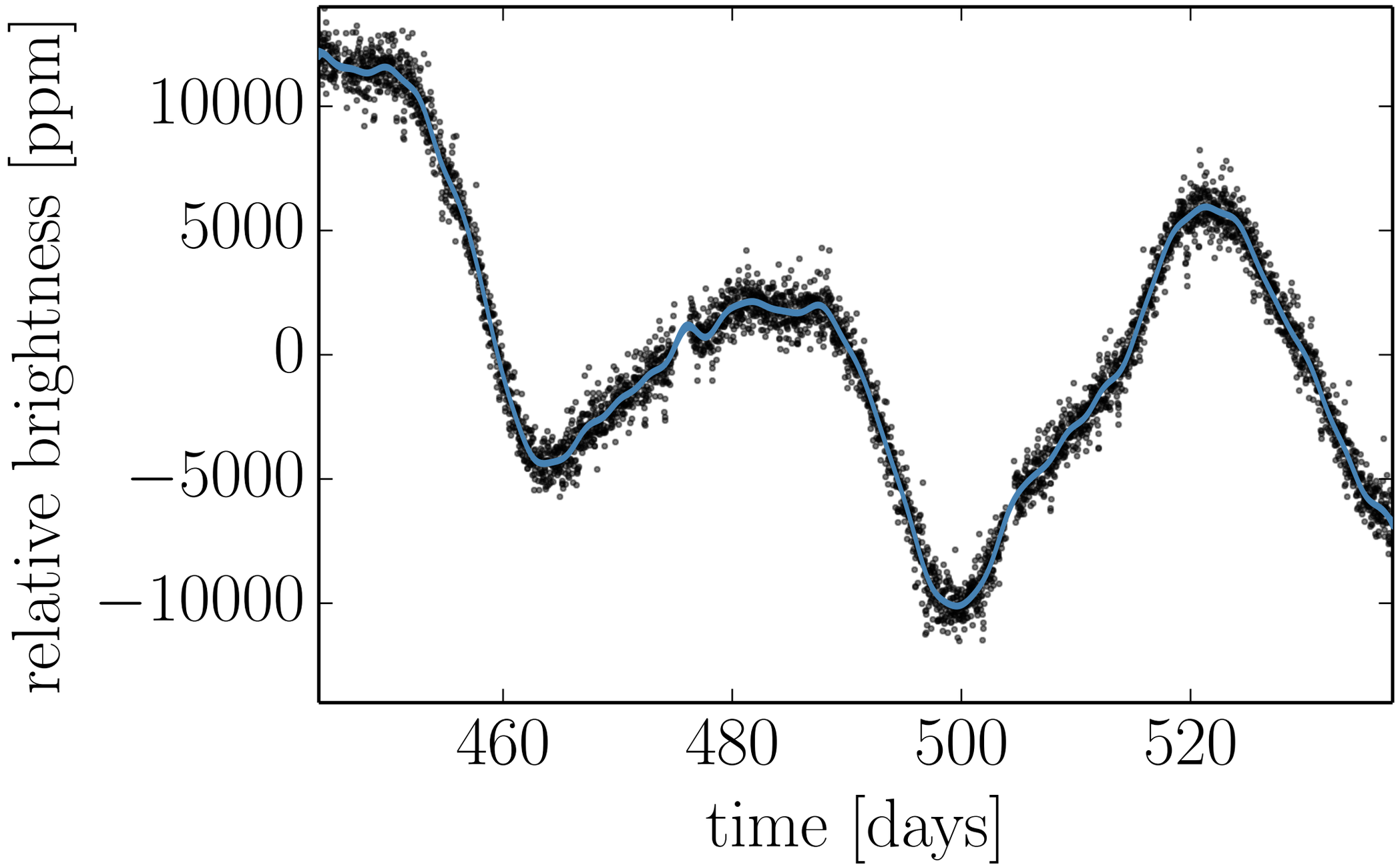

6
4
2
0
−2
−4
−6
y
0
1
2
3
4
5
x


6
4
2
0
−2
−4
−6
y
0
1
2
3
4
5
x


6
4
2
0
−2
−4
−6
y
0
1
2
3
4
5
x


6
4
2
0
−2
−4
−6
y
0
1
2
3
4
5
x


relative brightness [ppm]
10000
5000
0
−5000
−10000
460
480
500
520
time [days]


relative brightness [ppm]
10000
5000
0
−5000
−10000
460
480
500
520
time [days]

the data are drawn from one

HUGE*

Gaussian

* the dimension is the number of data points.

The mathematical model

$$\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{f_\theta}(\boldsymbol{x}),\, K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})\right)$$

where

$$\left[K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})\right]_{ij} = {\sigma_i}^2\, \delta_{ij} + k_{\boldsymbol{\alpha}}(x_i,\, x_j)$$

The mathematical model

$$\log p(\boldsymbol{y} \,|\, \boldsymbol{x},\, \boldsymbol{\sigma},\, \boldsymbol{\theta},\, \boldsymbol{\alpha}) = -\frac{1}{2}\,[\boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})]^{\mathrm{T}}\, K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})^{-1}\,[\boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})] - \frac{1}{2}\log\det K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma}) - \frac{N}{2}\log 2\,\pi$$

where

$$[K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})]_{ij} = {\sigma_i}^2\,\delta_{ij} + k_{\boldsymbol{\alpha}}(x_i,\, x_j)$$

The mathematical model

$$\log p(\boldsymbol{y} \,|\, \boldsymbol{x},\, \boldsymbol{\sigma},\, \boldsymbol{\theta},\, \boldsymbol{\alpha}) = -\frac{1}{2}\left[\boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})\right]^{\mathrm{T}} K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})^{-1}\left[\boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})\right] - \frac{1}{2}\log\det K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma}) - \frac{N}{2}\log 2\,\pi$$

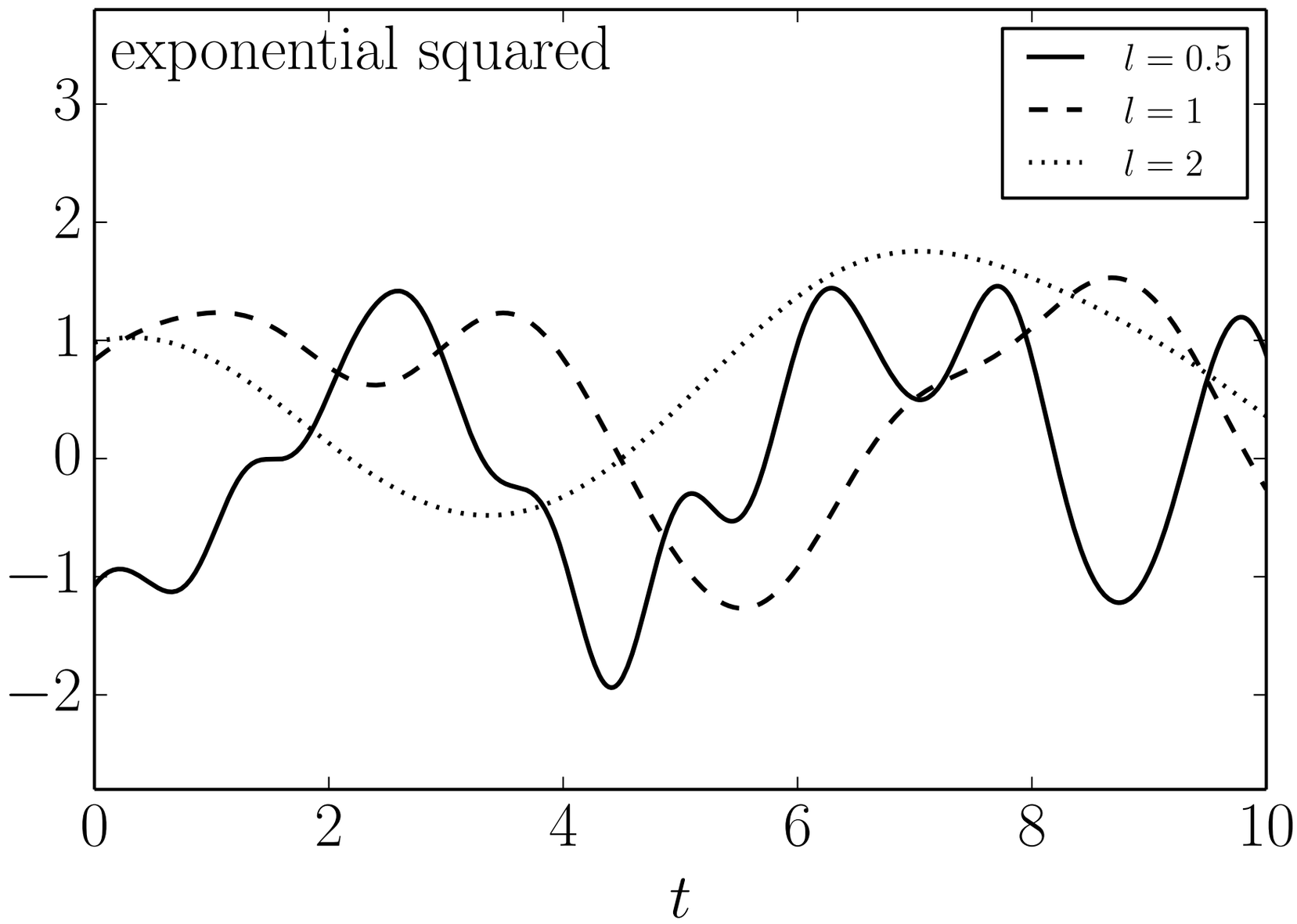
where

$$\left[K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})\right]_{ij} = {\sigma_i}^2\,\delta_{ij} + \underbrace{k_{\boldsymbol{\alpha}}(x_i,\, x_j)}_{\textbf{kernel function}}$$

(where the magic happens)

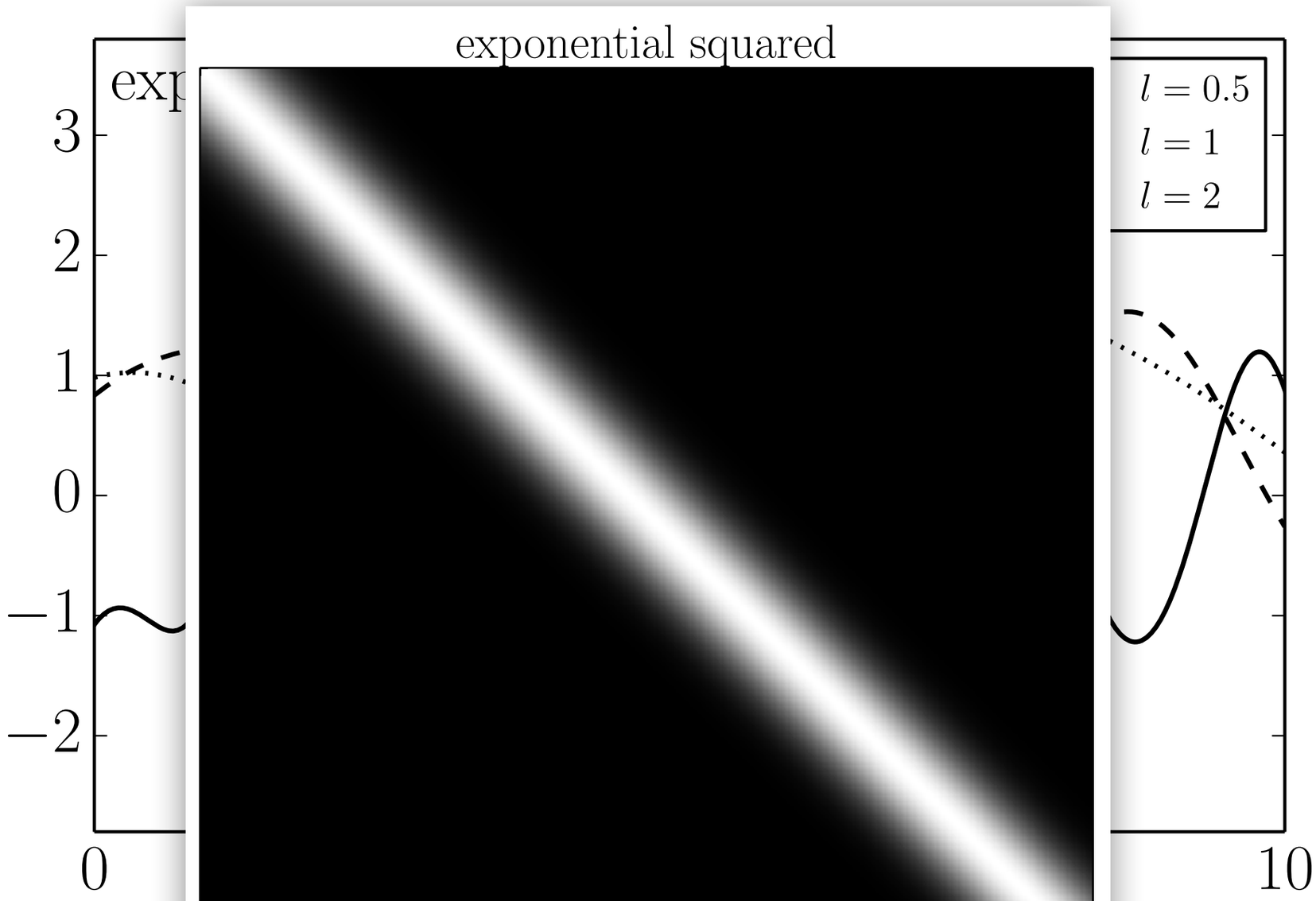
The choice of kernel

$$k_{\boldsymbol{\alpha}}(x_i,\, x_j) = \exp\left(-\frac{[x_i - x_j]^2}{2\,\ell^2}\right)$$



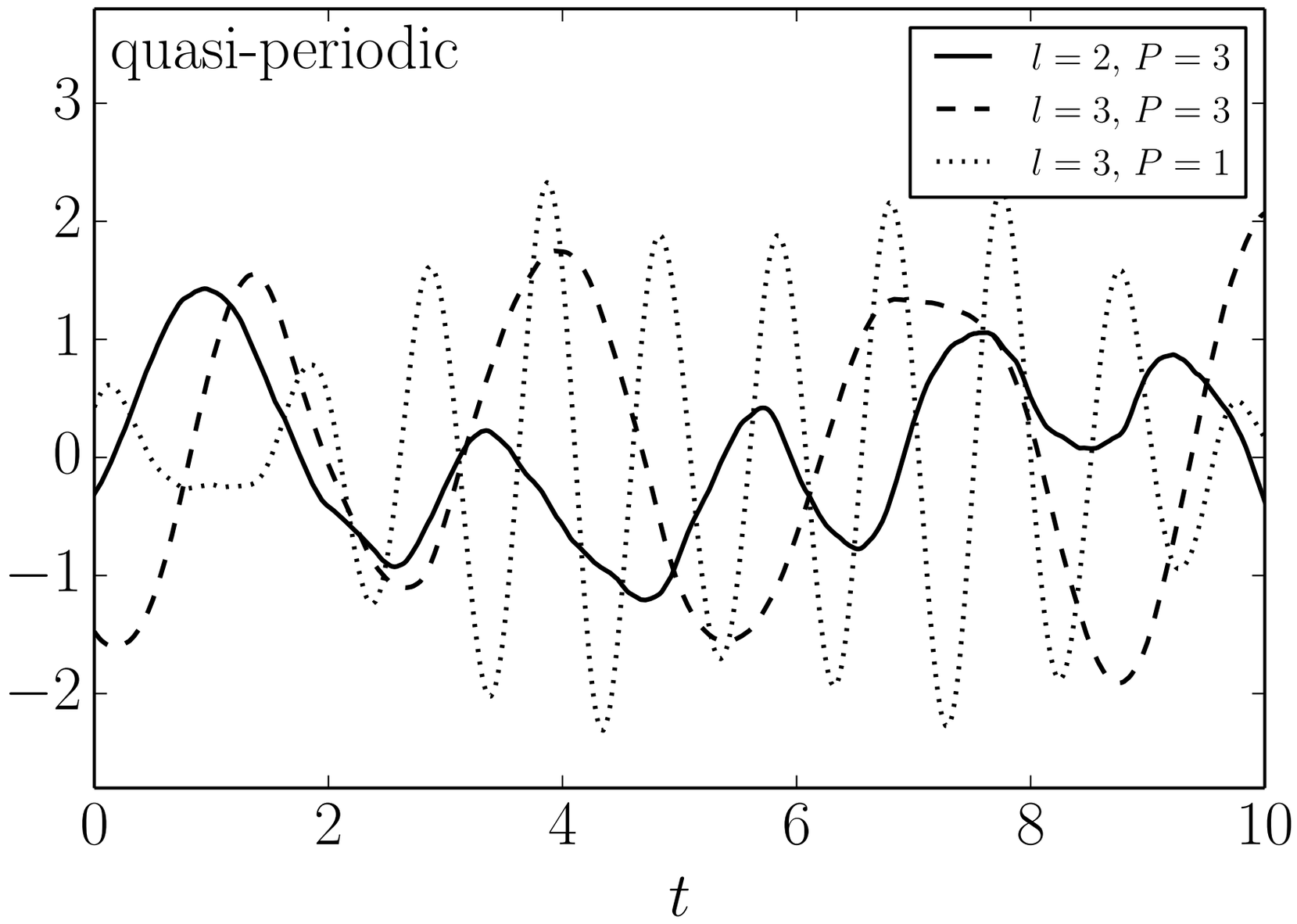
The choice of kernel

$$k_{\boldsymbol{\alpha}}(x_i,\, x_j) = \exp\left(-\frac{[x_i - x_j]^2}{2\,\ell^2}\right)$$



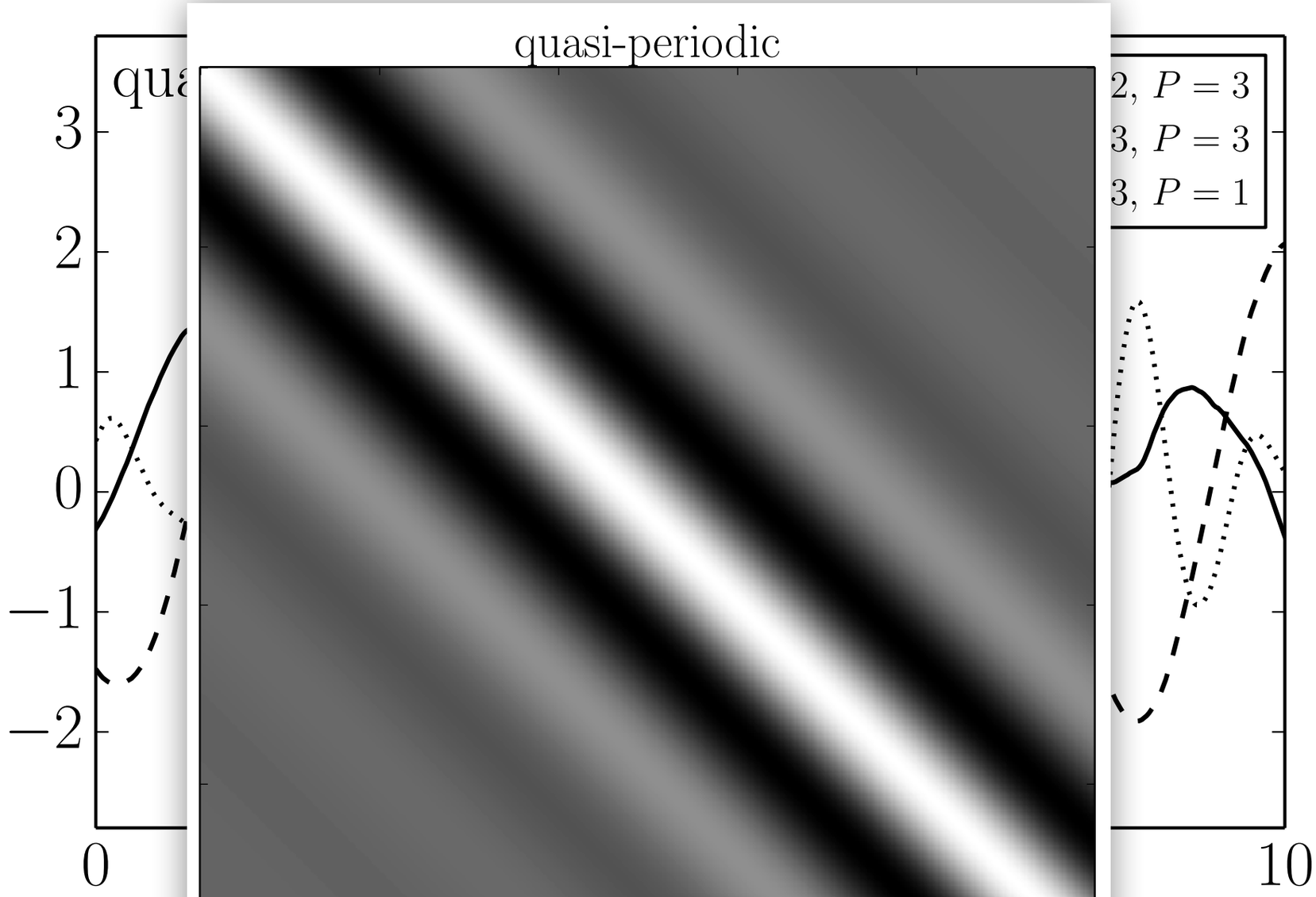
The choice of kernel

$$k_{\boldsymbol{\alpha}}(x_i,\, x_j) = \left[1 + \frac{\sqrt{3}\,|x_i - x_j|}{\ell}\right] \exp\left(-\frac{|x_i - x_j|}{\ell}\right) \cos\left(\frac{2\,\pi\,|x_i - x_j|}{P}\right)$$

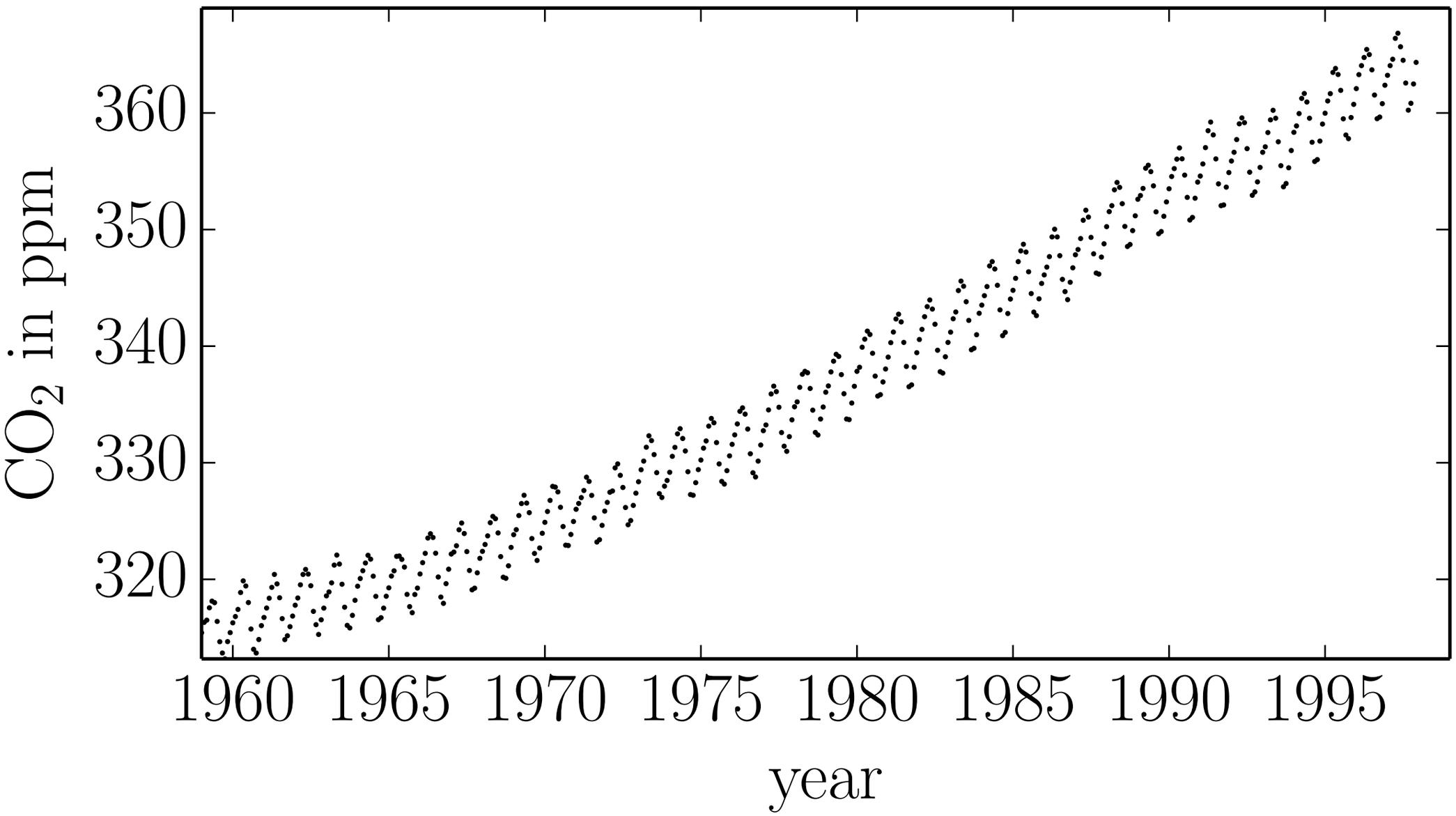


The choice of kernel

$$k_{\boldsymbol{\alpha}}(x_i,\, x_j) = \left[1 + \frac{\sqrt{3}\,|x_i - x_j|}{\ell}\right] \exp\left(-\frac{|x_i - x_j|}{\ell}\right) \cos\left(\frac{2\,\pi\,|x_i - x_j|}{P}\right)$$

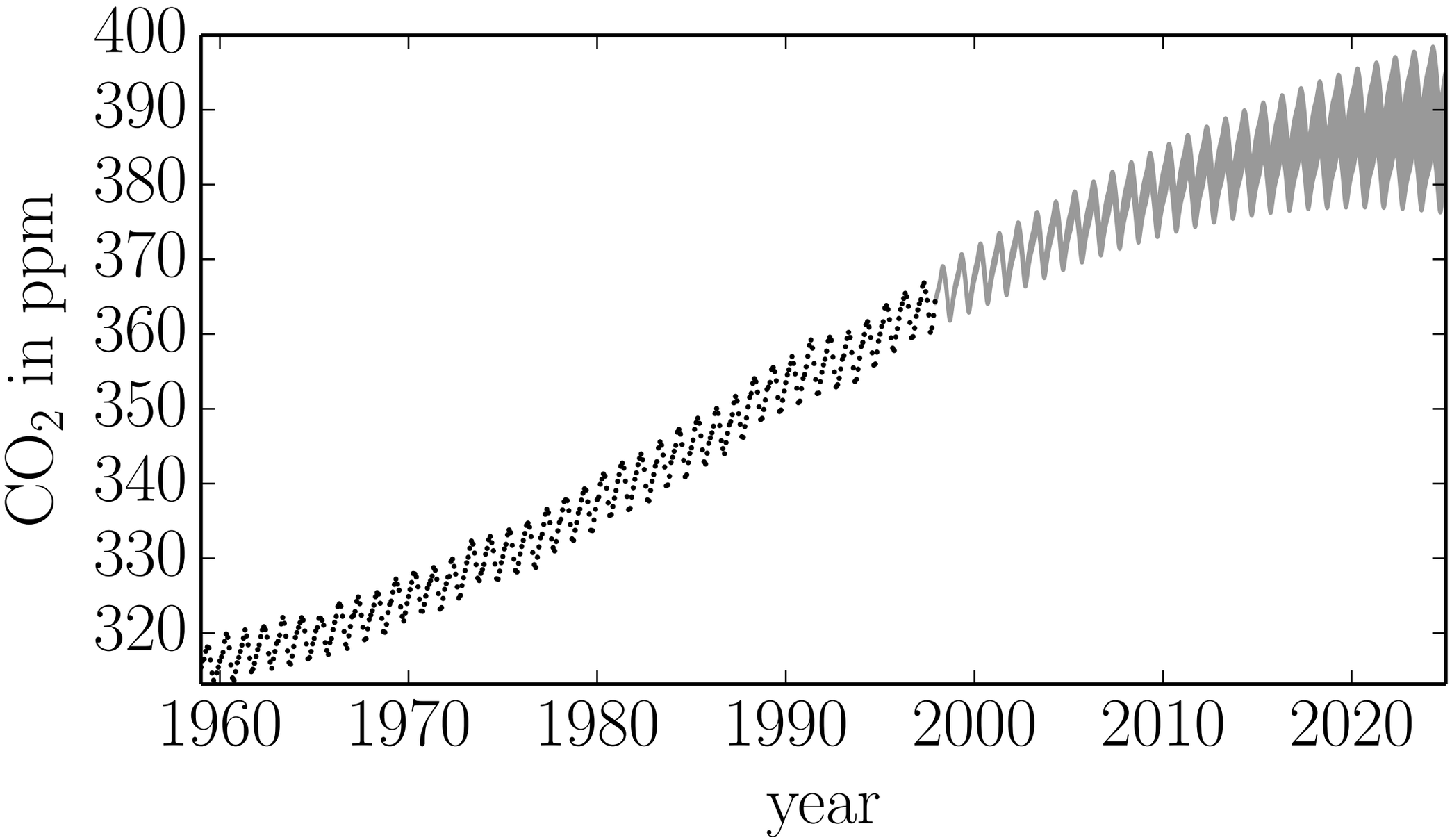


The choice of kernel



Full write-up at **dfm.io/george**

The choice of kernel



Full write-up at **dfm.io/george**

The mathematical model

$$\log p(\boldsymbol{y} \,|\, \boldsymbol{x},\, \boldsymbol{\sigma},\, \boldsymbol{\theta},\, \boldsymbol{\alpha}) = -\frac{1}{2}\left[\boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})\right]^{\mathrm{T}} K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})^{-1}\left[\boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})\right] - \frac{1}{2}\log\det K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma}) - \frac{N}{2}\log 2\,\pi$$

where

$$\left[K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})\right]_{ij} = {\sigma_i}^2\,\delta_{ij} + \underbrace{k_{\boldsymbol{\alpha}}(x_i,\, x_j)}_{\textbf{kernel function}}$$

(where the magic happens)

A simple & efficient Python implementation

```
import numpy as np
from scipy.linalg import cho_factor, cho_solve

def kernel(x1, x2):
    # ...

def gp_lnlike(x, y, yerr):
    C = kernel(x[:, None], x[None, :])
    C[np.diag_indicies_from(C)] += yerr ** 2
    factor, flag = cho_factor(C)
    logdet = 2*np.sum(np.log(np.diag(factor)))
    return -0.5 * (np.dot(y, cho_solve((factor, flag), y))
                   + logdet + len(x)*np.log(2*np.pi))
```

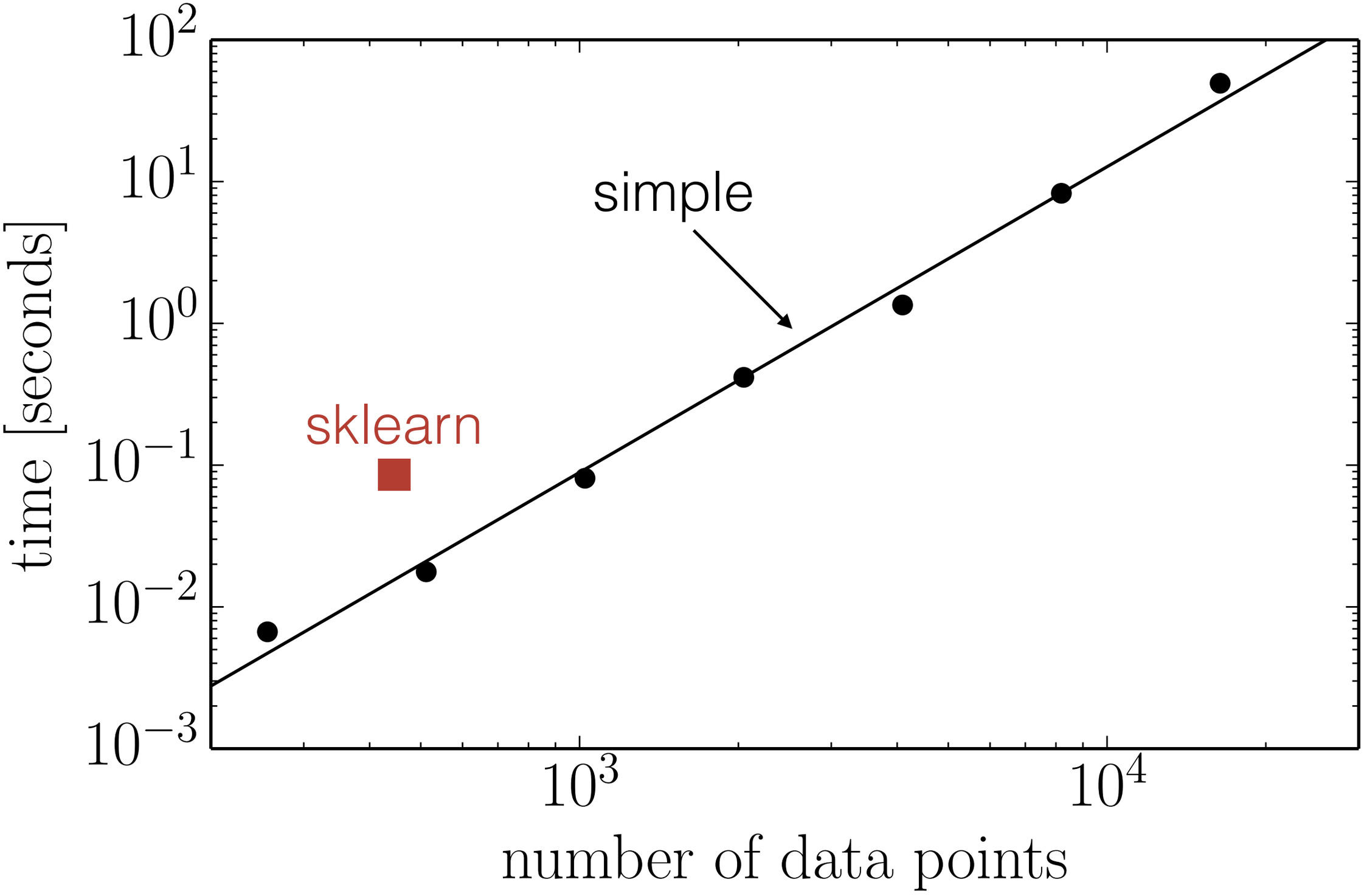
Using George

```
import george
import numpy as np

# kernel = george.kernels...

def george_lnlike(x, y, yerr):
    gp = george.GP(kernel)
    gp.compute(x, yerr)
    return gp.lnlikelihood(y)
```

github.com/dfm/george


simple
sklearn
time [seconds]
number of data points
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^3
10^4

What's the catch?

My Problem

=

Big Data

(by some definition)

Note: I hate myself for this slide too…

Computational complexity.

$$\log p(\boldsymbol{y} \,|\, \boldsymbol{x},\, \boldsymbol{\sigma},\, \boldsymbol{\theta},\, \boldsymbol{\alpha}) = -\frac{1}{2}\left[\boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})\right]^{\mathrm{T}} K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})^{-1}\left[\boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})\right] - \frac{1}{2}\log\det K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma}) - \frac{N}{2}\log 2\,\pi$$

compute factorization // evaluate log-det // apply inverse

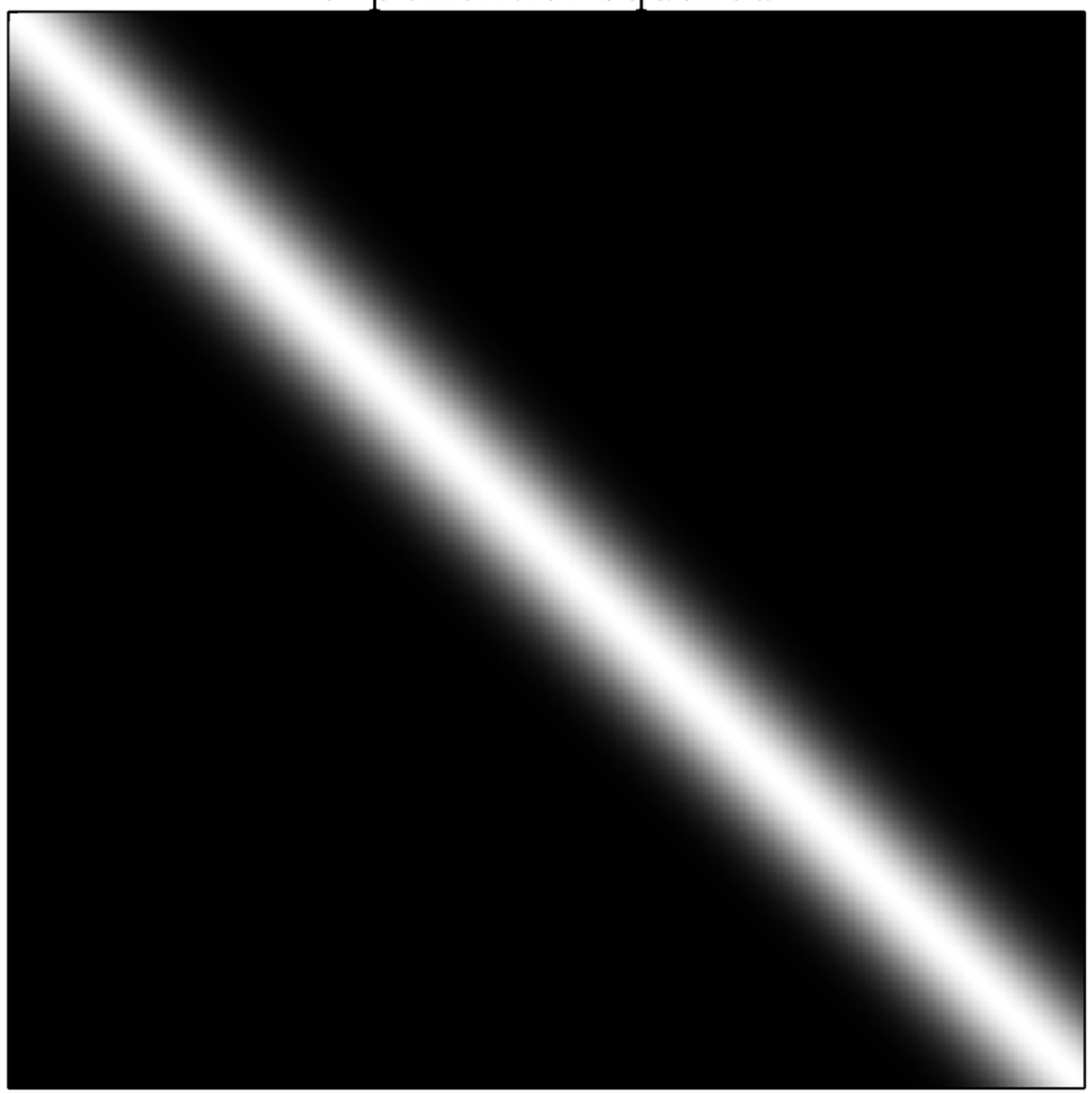
naïvely: $\mathcal{O}(N^3)$

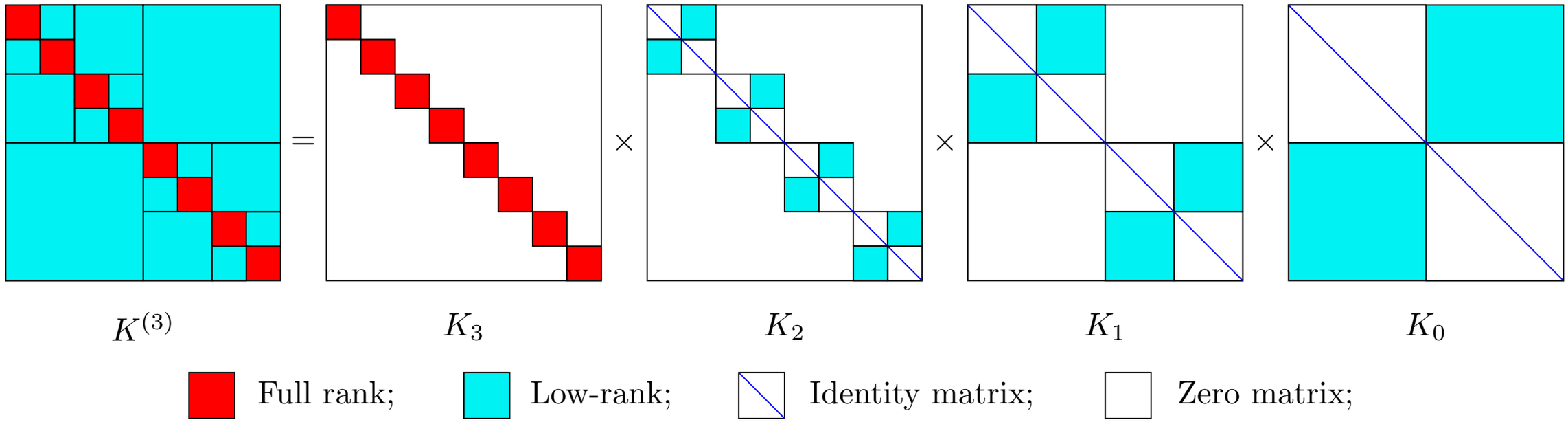
“

Aren’t kernel matrices Hierarchical Off-Diagonal Low-Rank?

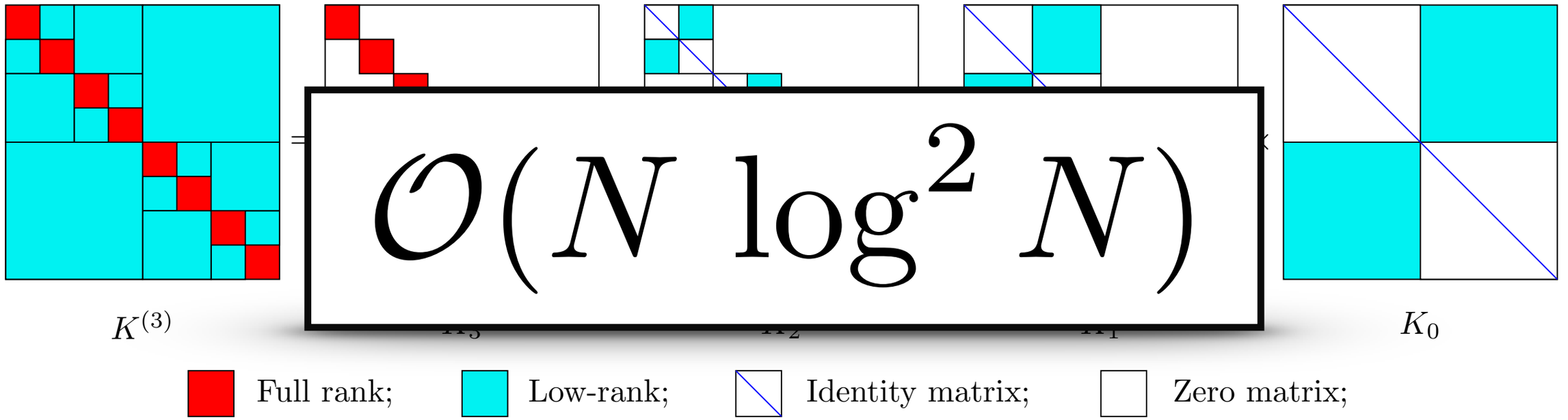
— not me

exponential squared



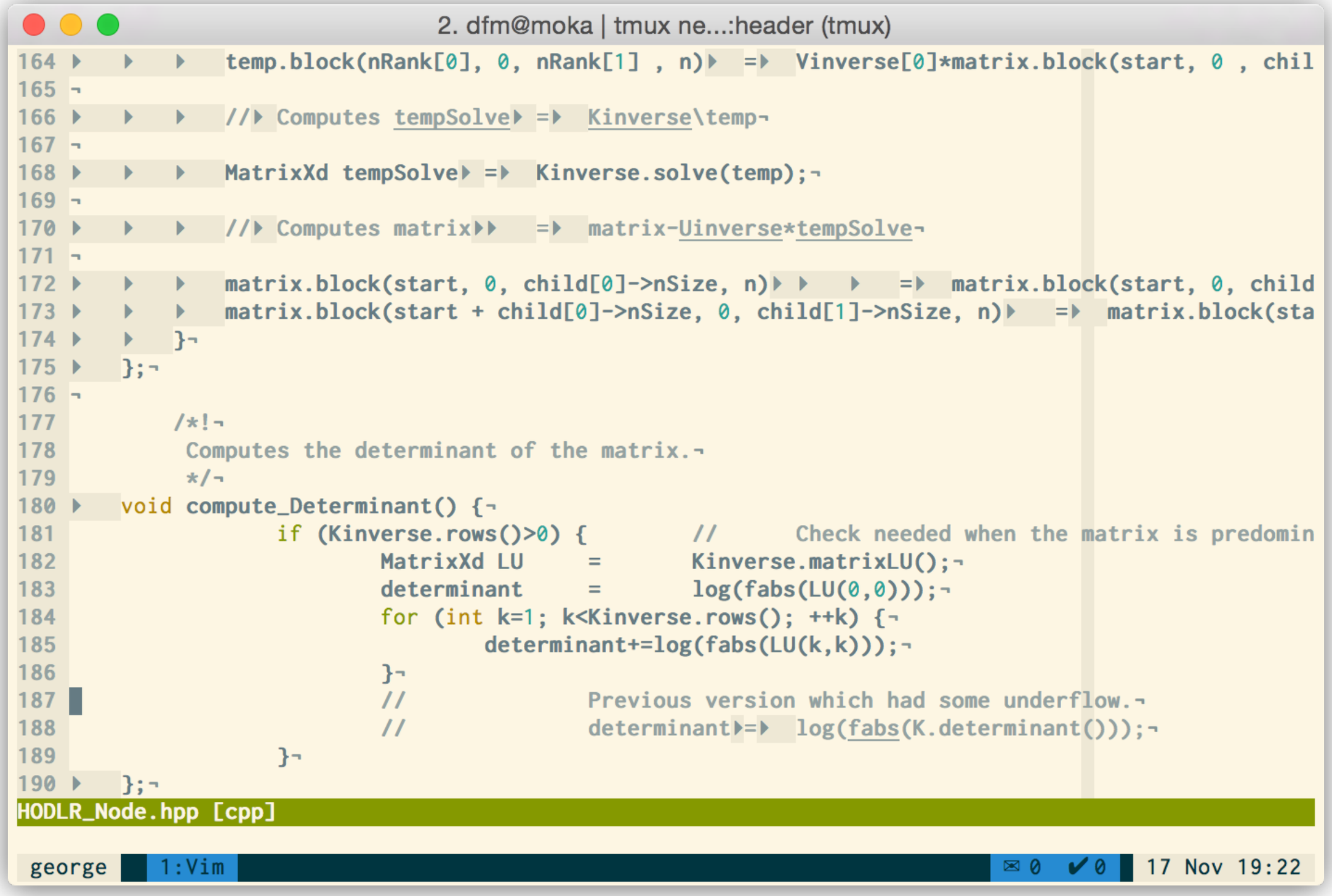


Ambikasaran, DFM, *et al.* (arXiv:1403.6015)



Ambikasaran, DFM, *et al.* (arXiv:1403.6015)

github.com/sivaramambikasaran/HODLR



github.com/sivaramambikasaran/HODLR

The HODLR solver from George

```python
import george
import numpy as np

# kernel = george.kernels...

def george_lnlike(x, y, yerr):
    gp = george.GP(kernel)
    gp.compute(x, yerr)
    return gp.lnlikelihood(y)
```

github.com/dfm/george

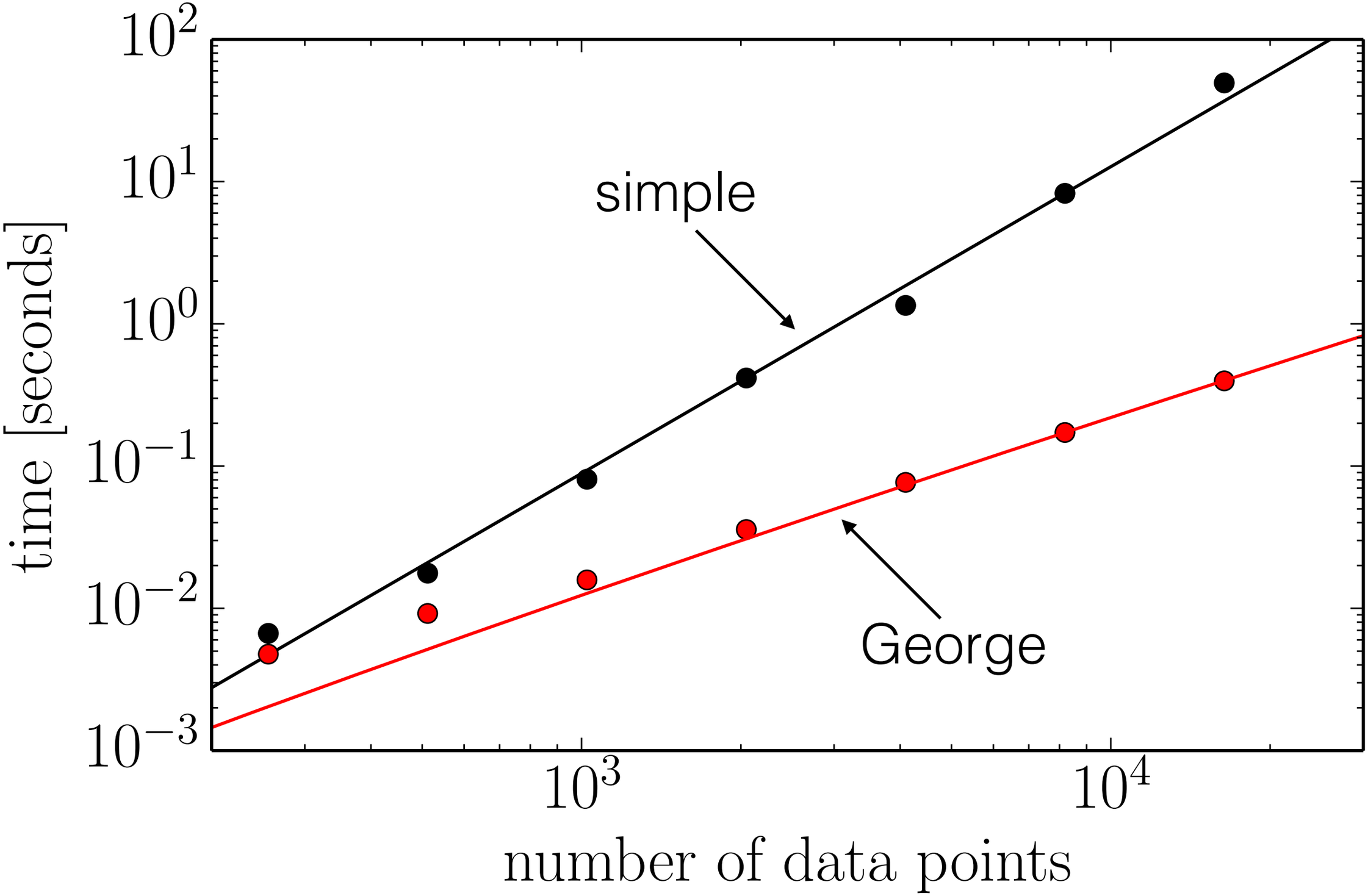
The HODLR solver from George

```python
import george
import numpy as np

# kernel = george.kernels...

def george_lnlike(x, y, yerr):
    gp = george.GP(kernel, solver=george.HODLRSolver)
    gp.compute(x, yerr)
    return gp.lnlikelihood(y)
```

github.com/dfm/george


10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
time [seconds]
simple
George
10^3
10^4
number of data points

References

gaussianprocess.org/gpml

github.com/dfm/[gp george

danfm@nyu.edu