

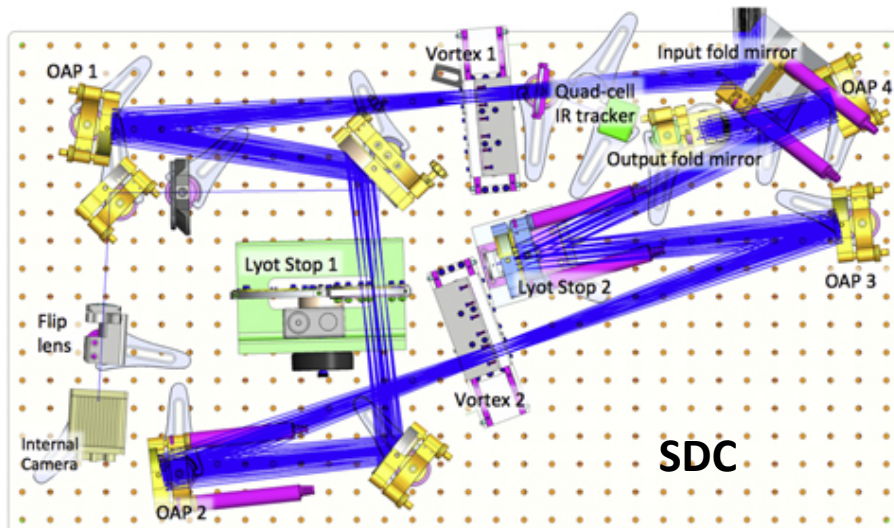
# Introduction to and state-of-the-art of post-processing for high-contrast imaging

Michael Bottom  
Optical Engineer  
High contrast imaging division (383A),  
Jet Propulsion Laboratory

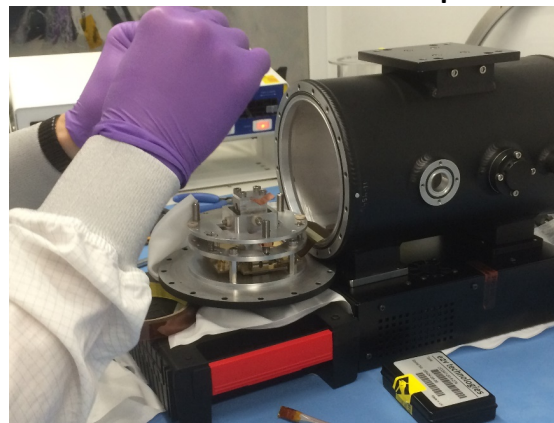
*with*  
Jeff Jewell  
Dimitri Mawet  
Garreth Ruane  
Graca Rocha

# (Things I work on)

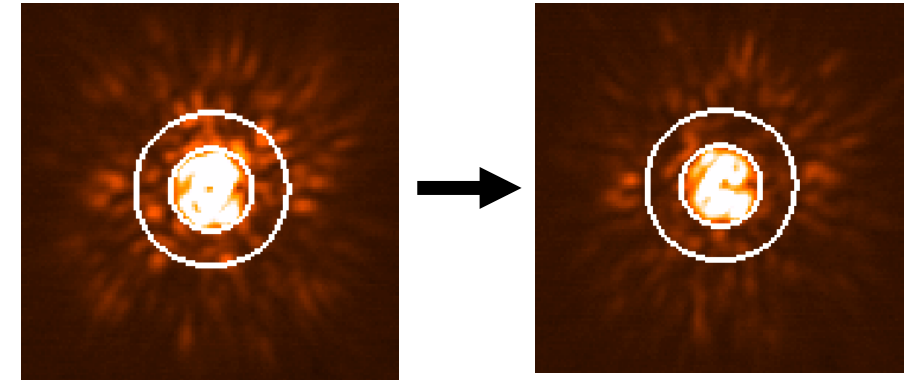
- WFIRST coronagraph EMCCD
- Starshade optical guidance systems
- On-sky, focal plane speckle suppression
- Stellar Double Coronagraph support
  - Binary star coronagraphy (Kuhn, Wang, Morales)
  - Microwave kinetic inductance detectors (Meeker, Mazin)
  - Self-coherent camera development (Delorme)
- Post-processing algorithms for high contrast imaging



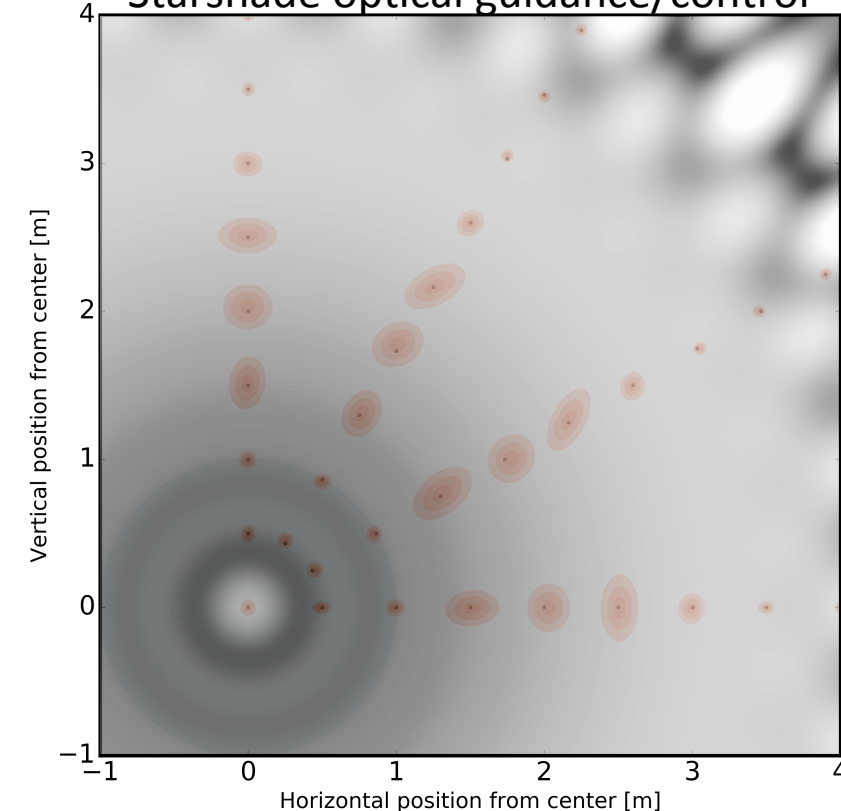
WFIRST detector development



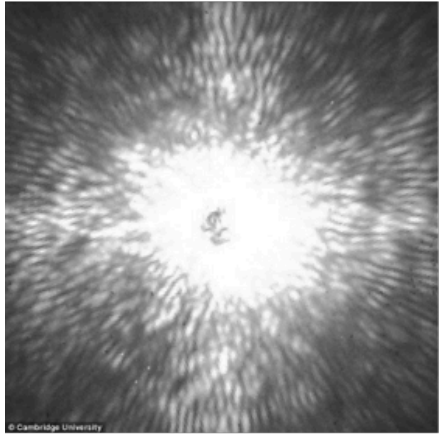
On-sky speckle nulling (Keck)



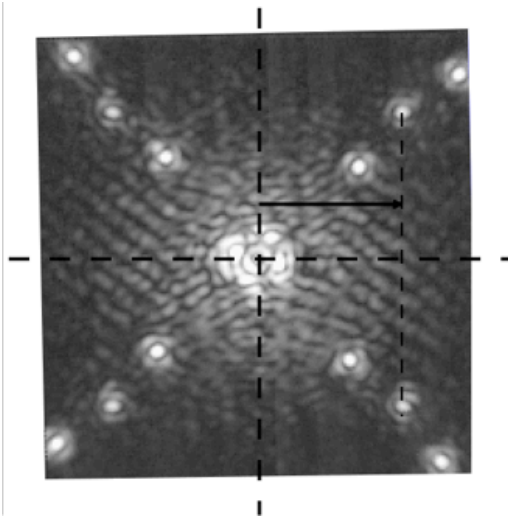
Starshade optical guidance/control



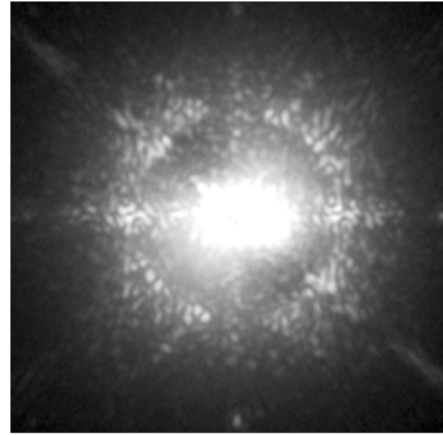
# High contrast imaging data is horrible



Project 1640



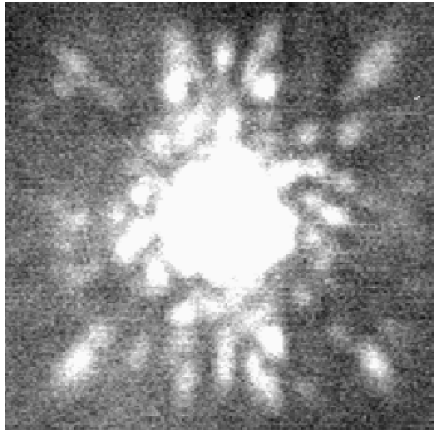
Gemini Planet Imager



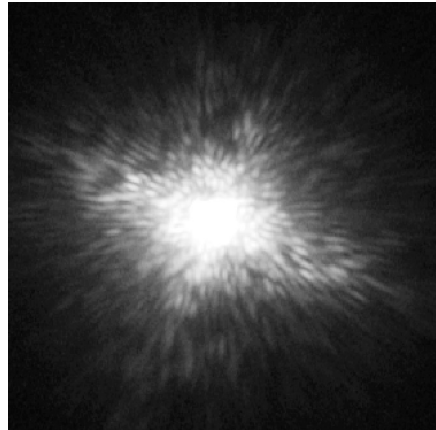
SPHERE

Planets, disks, etc are at or below the **systematic noise**

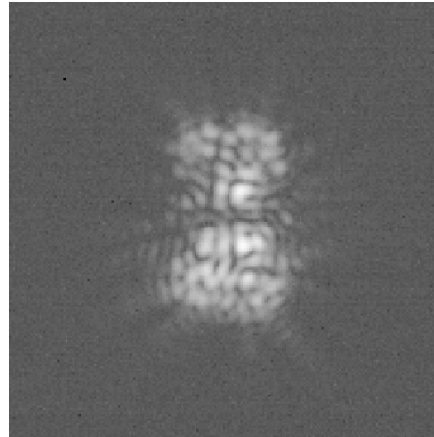
Post-processing is required



Keck



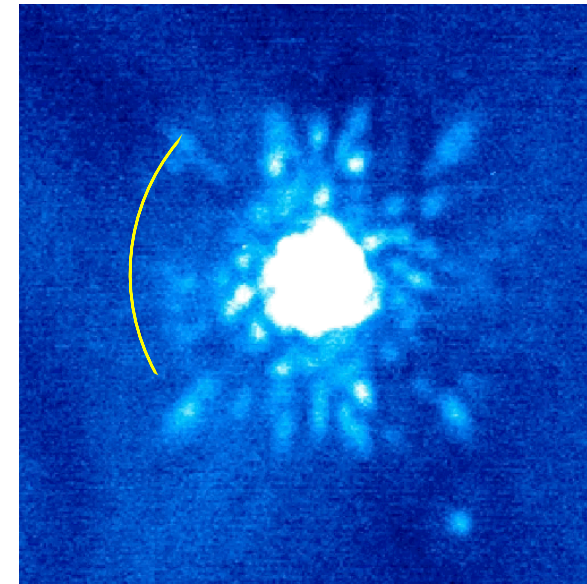
SDC



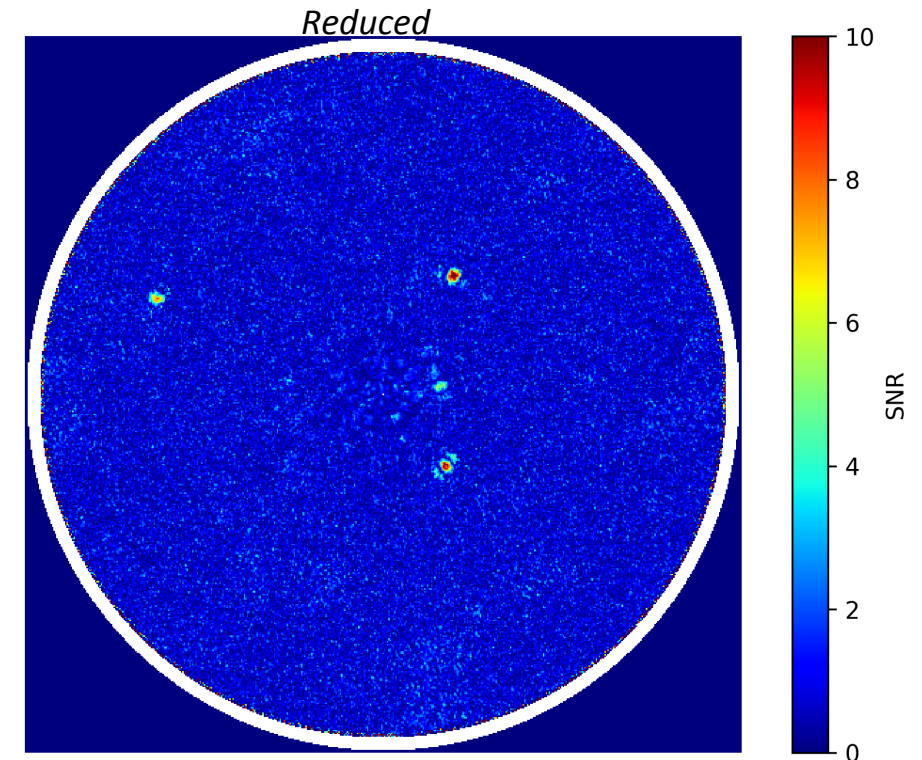
WFIRST-HCIT

# Typical data

- Many data frames of target star
  - Temporally separated
  - May be spectrally dispersed
- Coronagraph suppresses diffraction of central star
  - OK to saturate in areas you don't care about (eg central core)
- Speckles mostly fixed, but change through data
  - Atmosphere
  - Flexure
  - etc
  - *Minimizing changes (optical stability) is very important*
- Sky background fluctuations
- Detector noise
- Planet rotates through images



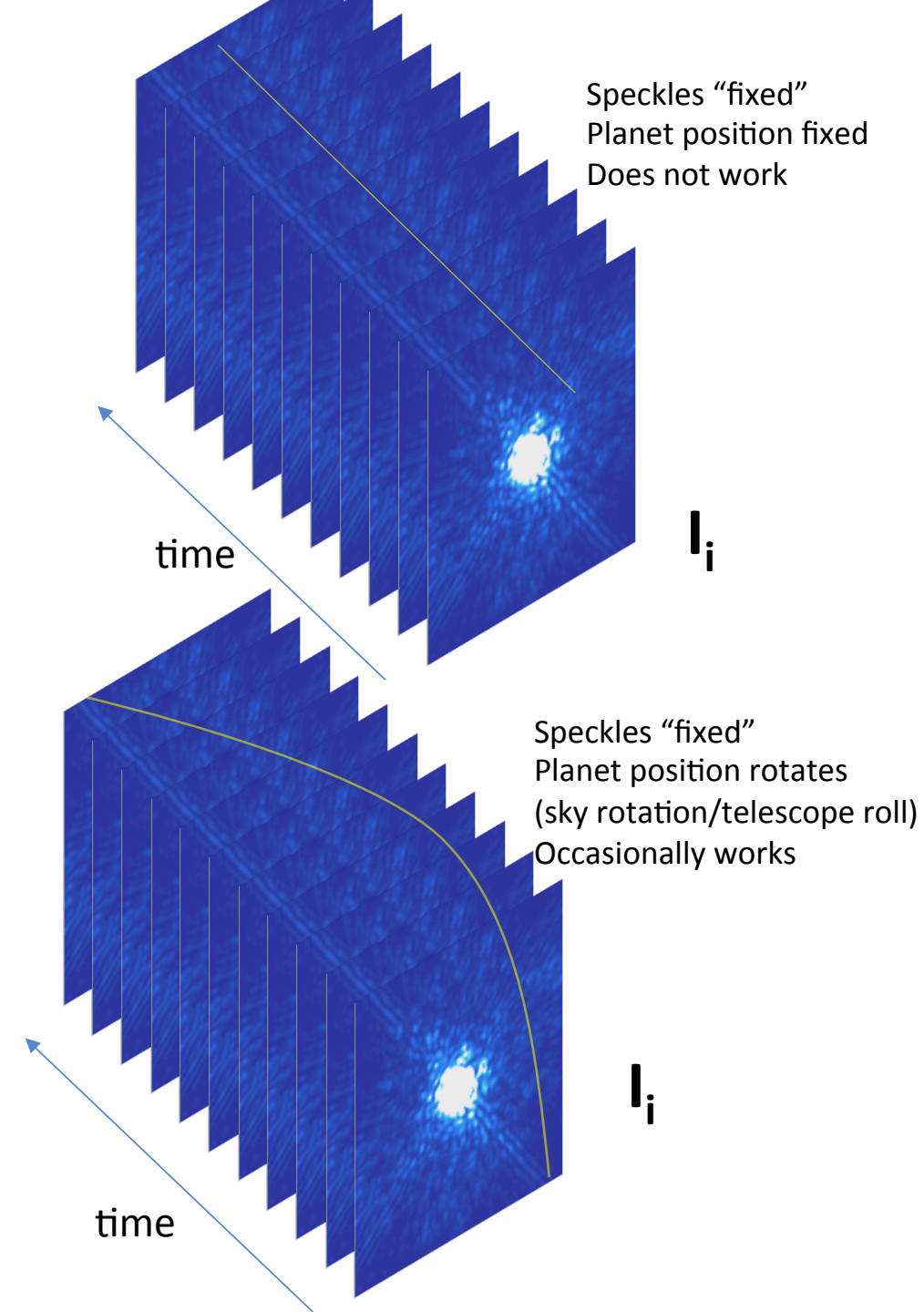
Raw data  
from Keck





# Typical data

- Many data frames of target star
  - Temporally separated
  - May be spectrally dispersed
- Coronagraph suppresses diffraction of central star
  - OK to saturate in areas you don't care about (eg central core)
- Speckles mostly fixed, but change through data
  - Atmosphere
  - Flexure
  - etc
  - *Minimizing changes (optical stability) is very important*
- Sky background fluctuations
- Detector noise
- Planet rotates through images



# Perfect post-processing and analysis

- For each frame of the datacube, let the intensity be

*Intensity measured*   *Speckle signal*   *Speckle noise*   *Planet signal*   *Planet noise*   *Detector noise*

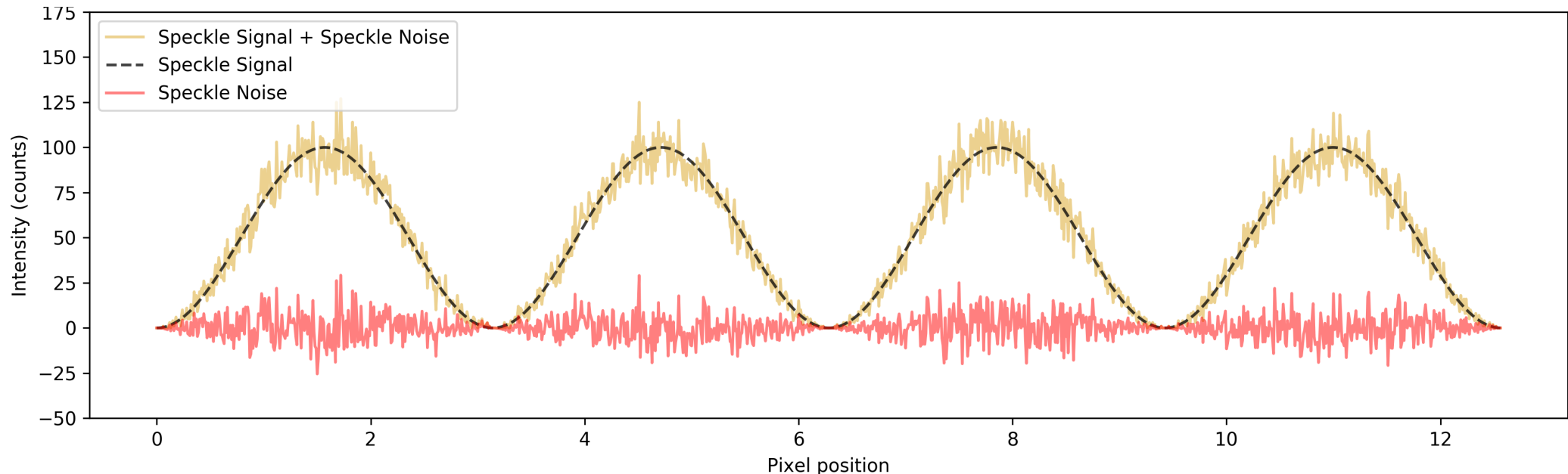
$$\text{Frame \#} \rightarrow I_i(x, y) = S_i(x, y) + \mathcal{N}_{s,i}(x, y) + P_i(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y)$$

( $\mathcal{N}$  will be a spatially-dependent Poisson random variable, but may be more complicated for the detector)

- (optional) Calculate the “reduced datacube”

$$R_i(x, y) = I_i(x, y) - S_i(x, y)$$

$$= P_i(x, y) + \mathcal{N}_{s,i}(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y)$$



# Perfect post-processing and analysis

- For each frame of the datacube, let the intensity be

$$\begin{array}{ccccccc} \text{Intensity measured} & \text{Speckle signal} & \text{Speckle noise} & \text{Planet signal} & \text{Planet noise} & \text{Detector noise} \\ \text{Frame \#} \rightarrow & I_i(x, y) = S_i(x, y) + \mathcal{N}_{s,i}(x, y) + P_i(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y) \end{array}$$

(N will be a spatially-dependent Poisson random variable, but may be more complicated for the detector)

- (optional) Calculate the “reduced datacube”

$$\begin{aligned} R_i(x, y) &= I_i(x, y) - S_i(x, y) \\ &= P_i(x, y) + \mathcal{N}_{s,i}(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y) \end{aligned}$$

- Create a model of your planet/disk signal P, for example

$$P_i(r, \theta) = a \cdot PSF(r_0, \theta_0 + \theta_i)$$

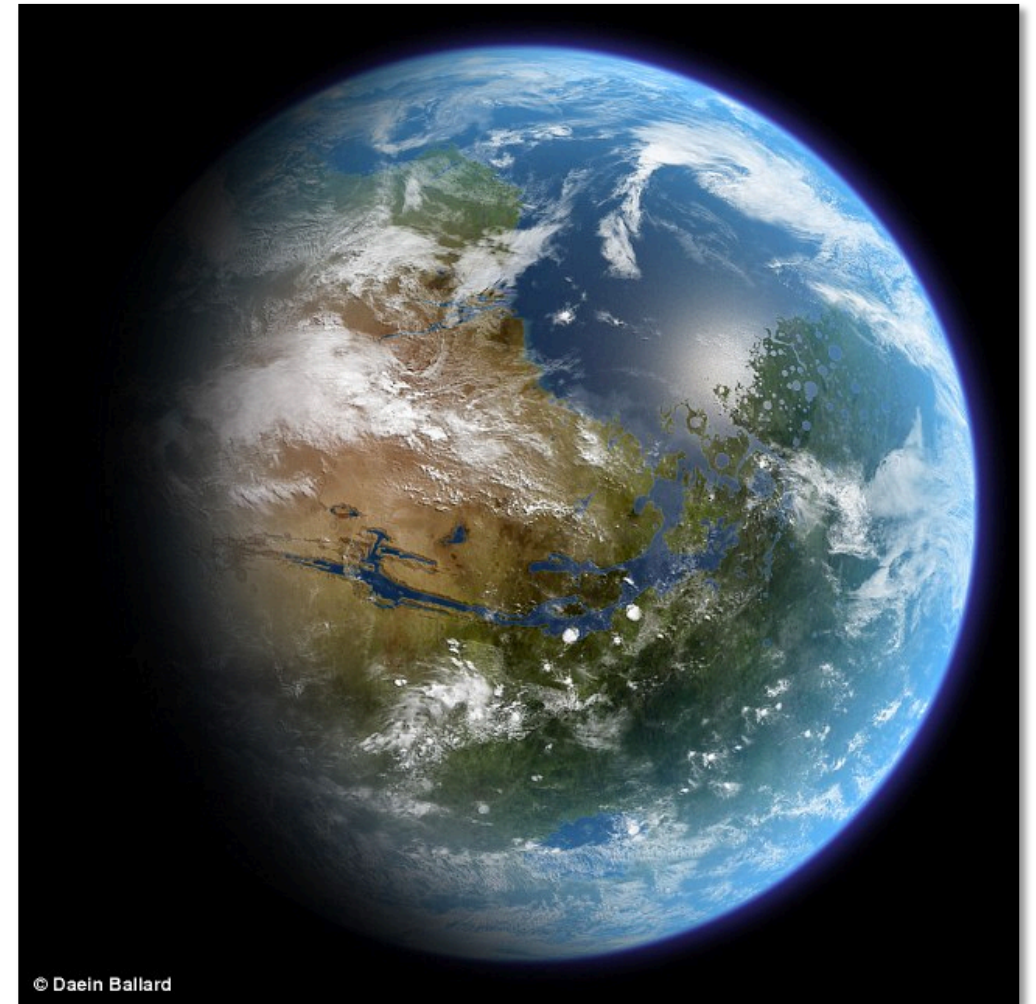
eg, Airy function, or your known instrument PSF  
sky rotation angle

- Create a likelihood function (knowing the noise) of the model and run an MCMC over the parameters you care about ( $a, r_0, \theta_0$ ) for **characterization**
  - What are the position, amplitude, etc of my planet or disk?
- Calculate the **Bayesian information criterion** against a model with no planet for **detection**
  - Is there really a planet or disk there?

# Thanks for your attention! Questions?

- Acknowledgements

- Jeff Jewell
- Dimitri Mawet
- Garreth Ruane
- Graça Rocha



Reduced image of one of the planets around Alpha Cen (in prep)



# ...not that simple

*Intensity measured*   *Speckle signal*   *Speckle noise*   *Planet signal*   *Planet noise*   *Detector noise*  
Frame #  $\rightarrow I_i(x, y) = S_i(x, y) + \mathcal{N}_{s,i}(x, y) + P_i(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y)$

- **You don't know  $S_i$ , the speckle signal**
  - You (usually) don't know  $\mathcal{N}_{s,i}$  either!
- You must estimate it from the science data  $I_i$
- You can also estimate it from other PSFs that behave similarly (a “reference” set)
- *The goals of post-processing are to **best estimate** (or equivalently, remove) **the speckle signal  $S_i$**  for each frame, **without including  $P_i$**  in the estimate of  $S_i$*

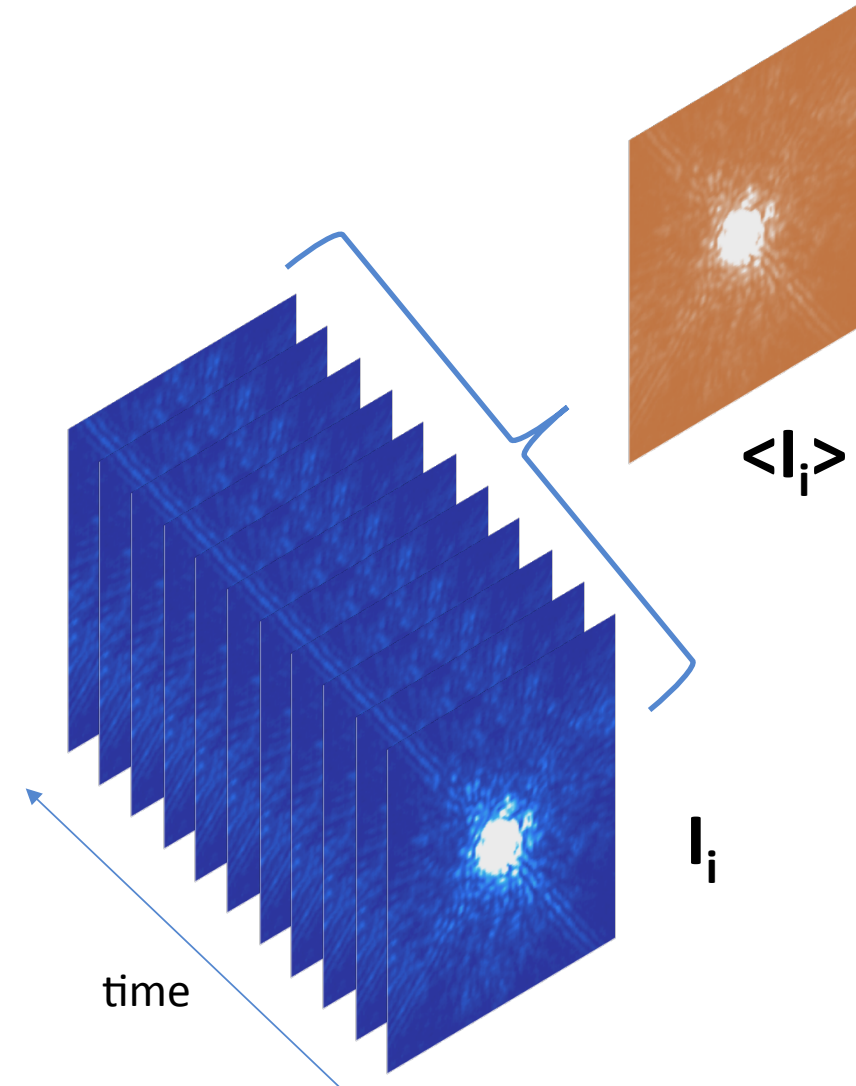
# Classic PSF subtraction

- For each (sub)image  $I_i$  in your datacube
- Minimize

$$R_i = I_i - \alpha_i \langle I_i \rangle$$

using least-squares

- The signal is rotating through the cube, so should survive
- Derotate the residuals  $R_i$  and median along the time axis



# Analysis: classic PSF subtraction

$$\begin{array}{cccccc} \text{Intensity measured} & \text{Speckle signal} & \text{Speckle noise} & \text{Planet signal} & \text{Planet noise} & \text{Detector noise} \\ I_i(x, y) = & S_i(x, y) + & \mathcal{N}_{s,i}(x, y) + & P_i(x, y) + & \mathcal{N}_{p,i}(x, y) + & \mathcal{N}_{d,i}(x, y) \end{array}$$

Perfect reduction  $\longrightarrow R_i(x, y) = I_i(x, y) - S_i(x, y)$

$$= P_i(x, y) + \mathcal{N}_{s,i}(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y)$$

- $S_i$  is estimated as a scaled version of the median of all the data frames

$$R_i = I_i - \alpha_i \langle I_i \rangle$$

that is,  $S_i \sim \alpha_i \langle I_i \rangle$

- As the number of frames grows,  $N \rightarrow 0$  by the CLT
  - $P \rightarrow 0$  as well due to rotation, or is 0 if a reference star is used
- Since you are minimizing residuals, you run the risk of fitting away the planet when the planet brightness gets really large...
- Since this is essentially a scalar, it works *only when  $S_i$  is not changing*.
  - **Never works well** in ground-based planet data
  - But it works **just as well as modern methods** when the speckles are static, such as in space coronagraph testbeds!!

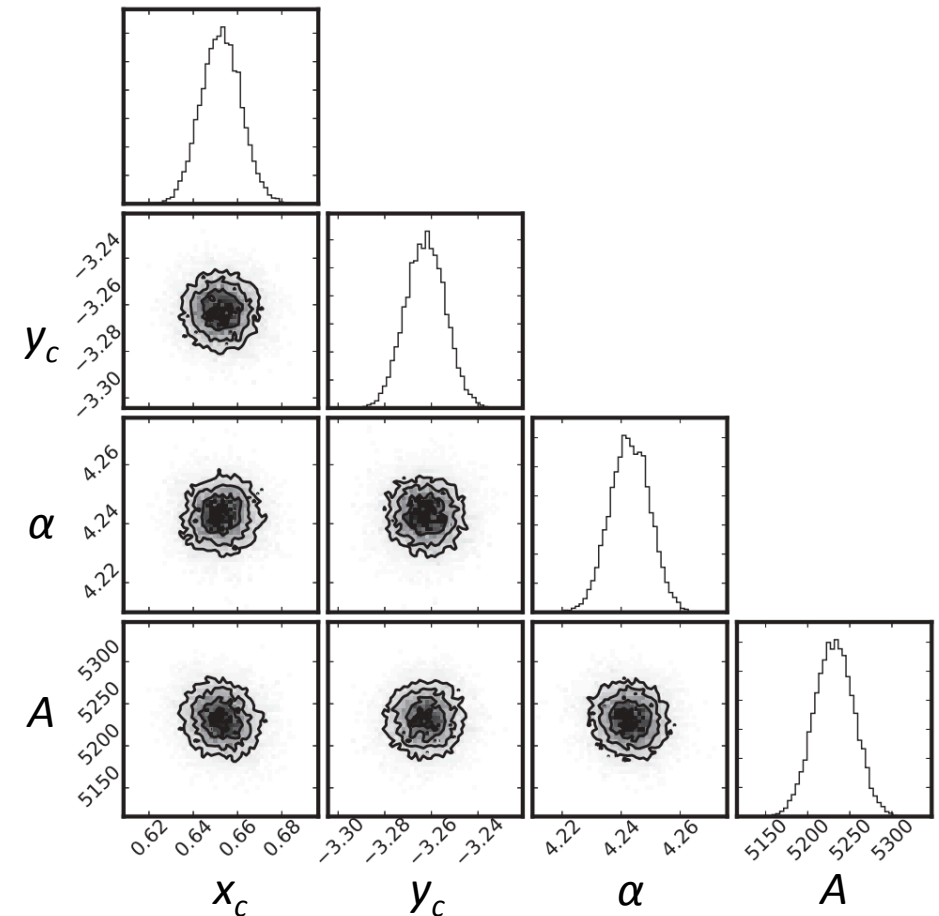
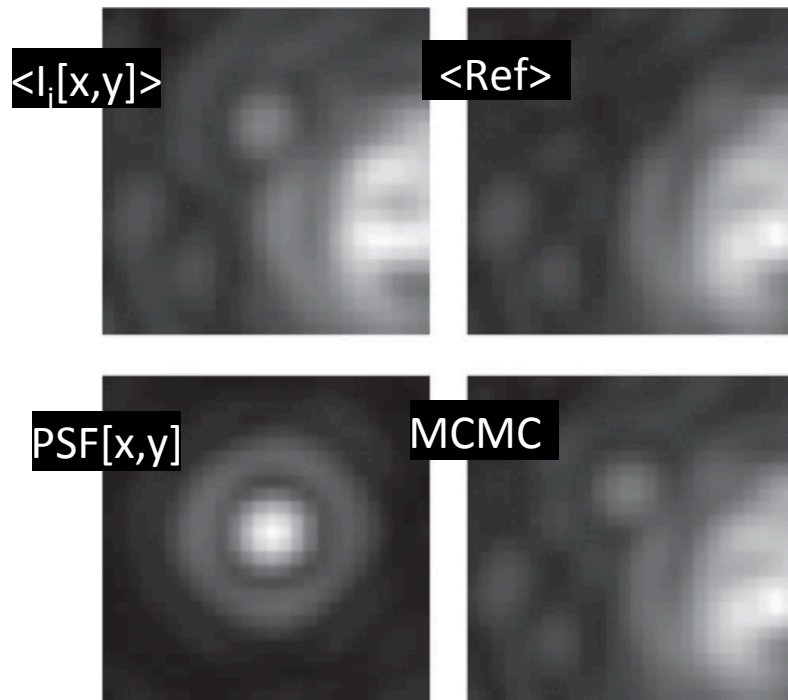
# Latest: classic PSF subtraction

Intensity measured    Speckle signal    Speckle noise    Planet signal    Planet noise    Detector noise

$$I_i(x, y) = S_i(x, y) + \mathcal{N}_{s,i}(x, y) + P_i(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y)$$

- Combined into an MCMC model:

$$\langle I_i[x, y] \rangle = \alpha \langle \text{Ref}[x, y] \rangle + A \cdot \text{PSF}[x - x_c, y - y_c]$$





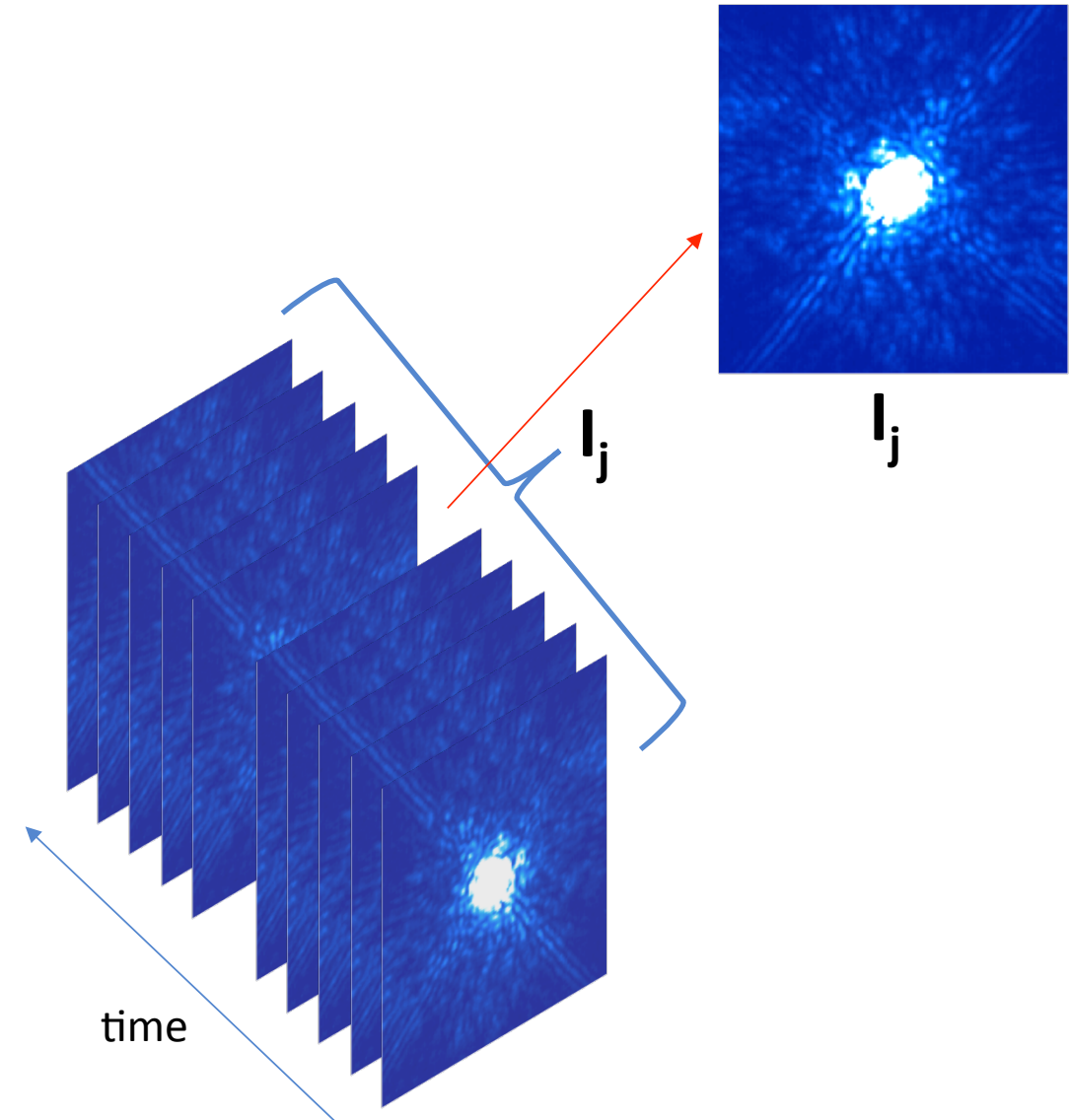
# Locally optimized combinations of images (LOCI)

- For each (sub)image  $I_i$  in your datacube
- Try to reproduce it by minimizing

$$R_i = I_i - \sum_j \alpha_{ij} I_j$$

solving for  $\alpha_{ij}$  using least-squares

- Replace  $I_j$  with a reference  $I'_j$  if using RDI
- The signal should be rotating through the cube, so should survive
- Derotate the residuals  $R_i$  and median along the time axis



# Analysis: LOCI

$$\begin{array}{cccccc} \text{Intensity measured} & \text{Speckle signal} & \text{Speckle noise} & \text{Planet signal} & \text{Planet noise} & \text{Detector noise} \\ I_i(x, y) = & S_i(x, y) + & \mathcal{N}_{s,i}(x, y) + & P_i(x, y) + & \mathcal{N}_{p,i}(x, y) + & \mathcal{N}_{d,i}(x, y) \end{array}$$

Perfect reduction  $\longrightarrow R_i(x, y) = I_i(x, y) - S_i(x, y)$

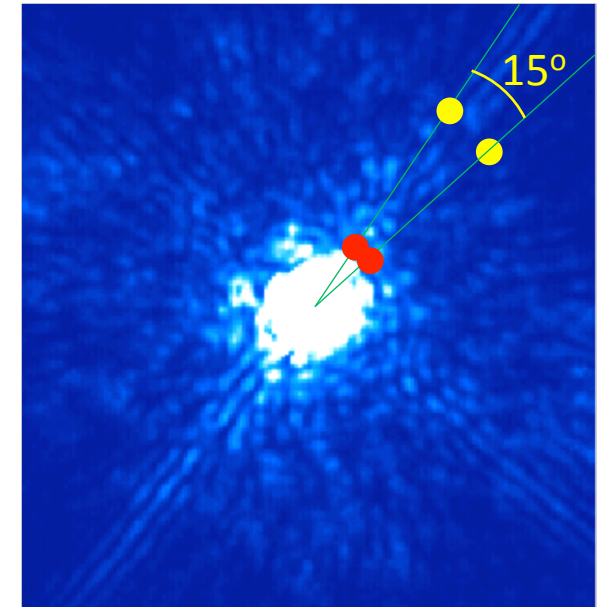
$$= P_i(x, y) + \mathcal{N}_{s,i}(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y)$$

- $S_i$  is estimated as a linear combination of other similar dataframes

$$R_i = I_i - \sum_j \alpha_{ij} I_j$$

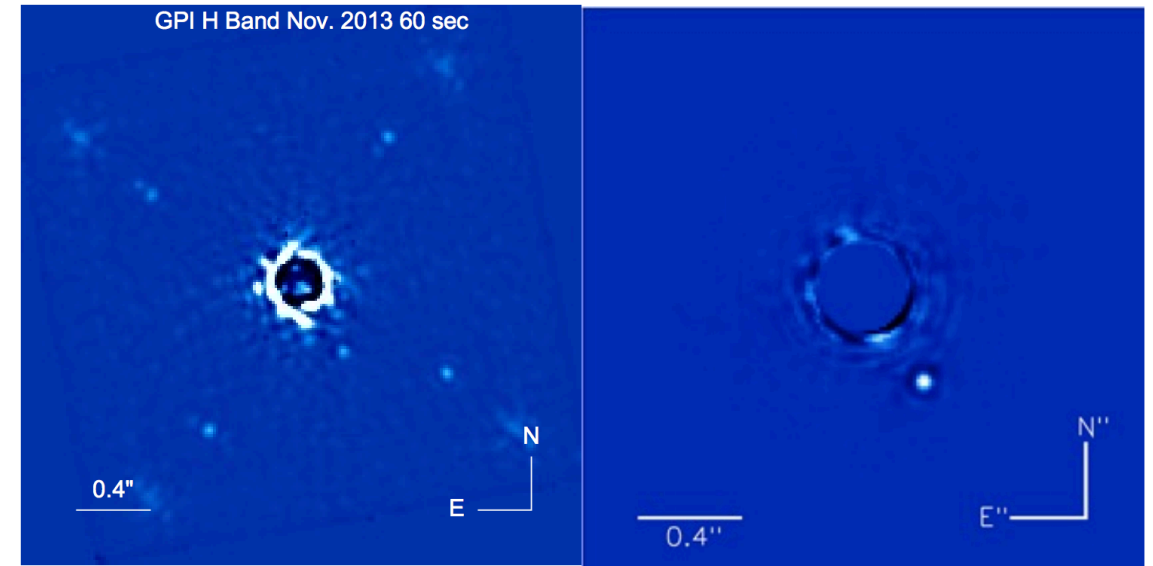
that is,  $S_i \sim \sum_j \alpha_{ij} I_j$

- The “best” dataframes to use will be the closest in time, but these will contain the planet signal in the estimate of  $S_i$ !
  - Can exclude nearby frames, making estimate of  $S_i$  worse
- By **minimizing** using least-squares residuals, you can fit away the planet!
- How do you get uncertainties and errors? MCMC?



# Latest: Template LOCI (TLOCI)

- **Incorporates PSF model** to prevent reduction from overfitting
  - Spatial PSF template
  - Spectral PSF template
- Selects images based on minimizing PSF contamination
- Forces coefficients to be positive
- Attempts to correct for planet subtraction
- Still challenging to get uncertainties
- No MCMC

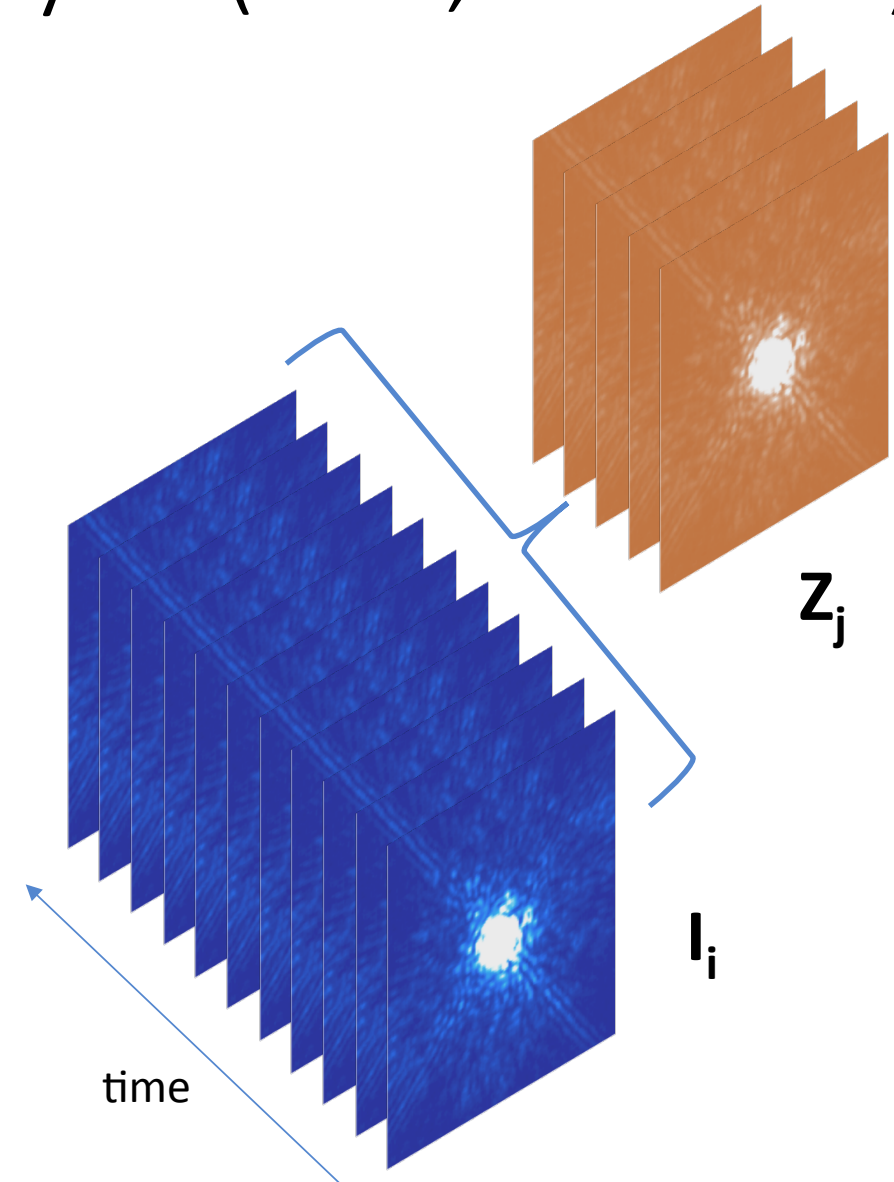


Beta Pic, single frame (GPI)

TLOCI reduction

# Principal Components Analysis (PCA, aka KLIP)

- Unlike LOCI, create “optimal” basis images  $\mathbf{Z}_j$  from your datacube  $\mathbf{I}_i$
- These basis vectors capture the *maximum variance in the shortest number of elements*
- Also known as the Karhunen-Loeve transform





# Principal Components Analysis (PCA, aka KLIP)

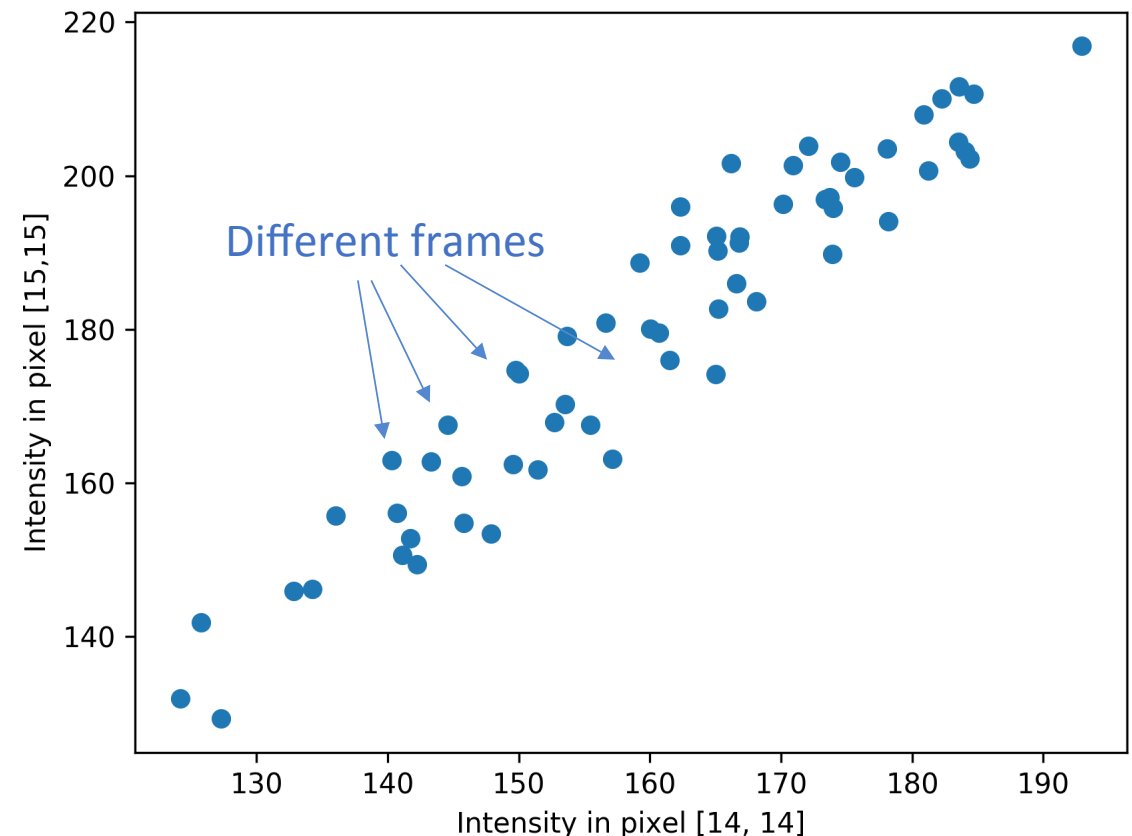
- Pixel intensities are correlated between images

- Procedure\*:

- Center data (mean subtract each pixel)
- Build *covariance matrix* of data cube

$$C = \frac{1}{N-1} X^T X$$

- Diagonal elements are variance of each pixel
- Off-diags are co-variances between pixels
- Calculate the eigenvectors, ordered by eigenvalue size-->principal components



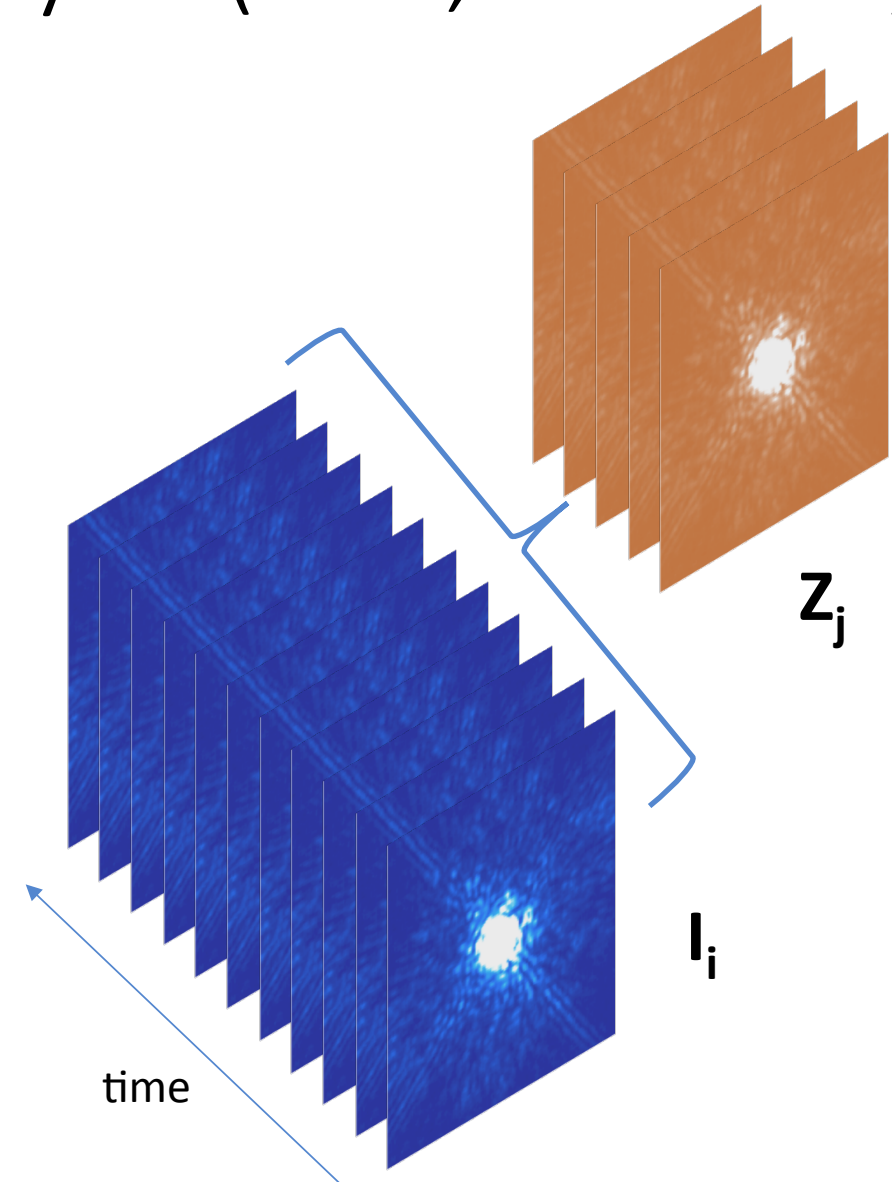
*\*NOTE: Don't do it this way. Modern methods to calculate the principal components use more stable and efficient algorithms to get same result*

# Principal Components Analysis (PCA, aka KLIP)

- For each (sub)image  $I_i$  in your datacube
- Calculate

$$R_i = I_i - \sum_j (Z_j \cdot I_i) Z_j$$

- Replace  $Z_j$  with a reference  $Z'_j$  if using RDI
- The signal should be rotating through the cube, so should survive
- Derotate the residuals  $R_i$  and median along the time axis



# Analysis: PCA

$$\begin{array}{cccccc} \text{Intensity measured} & \text{Speckle signal} & \text{Speckle noise} & \text{Planet signal} & \text{Planet noise} & \text{Detector noise} \\ I_i(x, y) = & S_i(x, y) + & \mathcal{N}_{s,i}(x, y) + & P_i(x, y) + & \mathcal{N}_{p,i}(x, y) + & \mathcal{N}_{d,i}(x, y) \end{array}$$

Perfect reduction  $\longrightarrow R_i(x, y) = I_i(x, y) - S_i(x, y)$

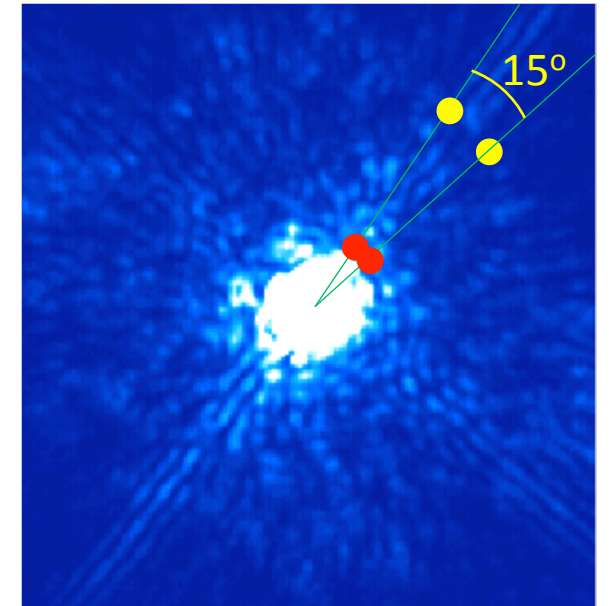
$$= P_i(x, y) + \mathcal{N}_{s,i}(x, y) + \mathcal{N}_{p,i}(x, y) + \mathcal{N}_{d,i}(x, y)$$

- $S_i$  is estimated as a linear combination of basis vectors

$$R_i = I_i - \sum_j (Z_j \cdot I_i) Z_j$$

that is,  $S_i \sim \sum_j (Z_j \cdot I_i) Z_j$

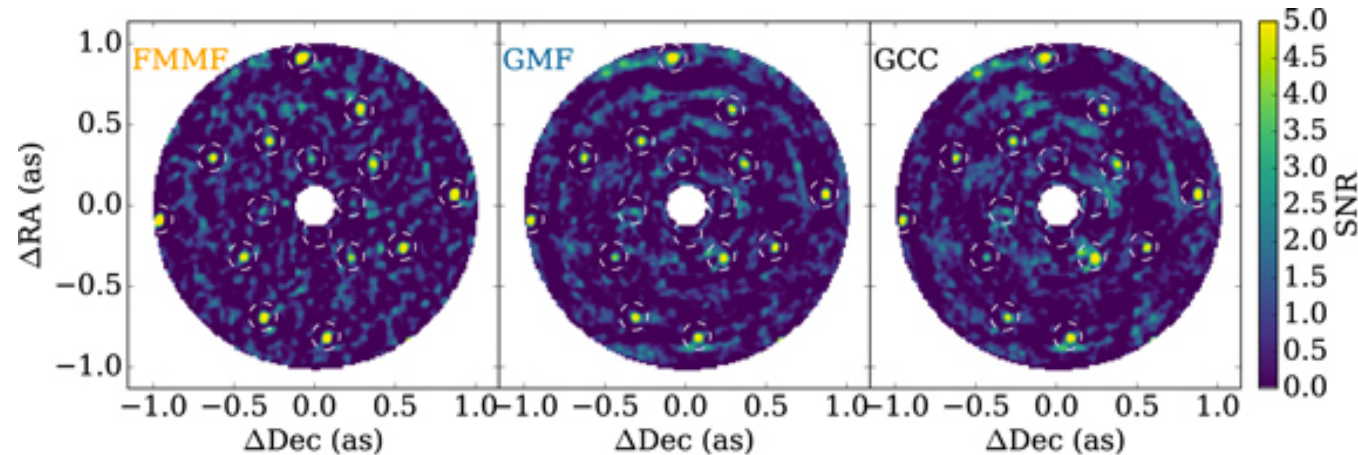
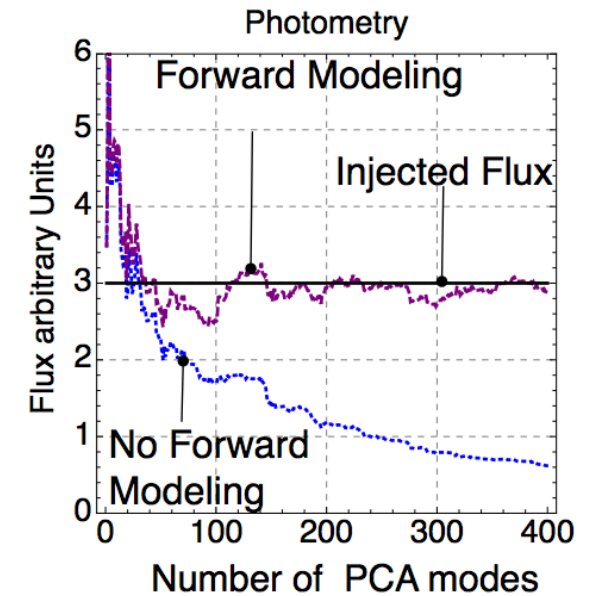
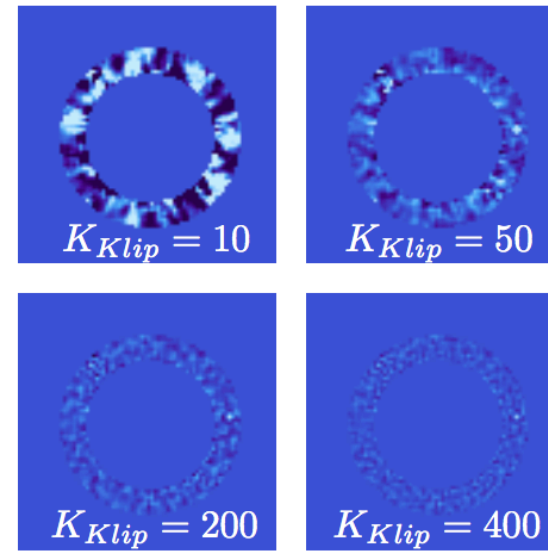
- The basis vectors can contain the planet signal!
  - Can try and exclude signal-containing frames, but lose fidelity
- **Projection onto principal components** remove flux from the planet.
  - Too many components--> no more planet
- How do you get uncertainties and errors? MCMC?



## Latest: Forward modeling PCA

$$R_i = I_i - \sum_j (Z_j \cdot I_i) Z_j$$

- Use a model of the (spatial) PSF to figure out how much the PSF gets eaten by the principal components
- Include and correct for the effects of the PSF in the covariance matrix calculation
- Run a match filter in the reduced datacube using the PCA-distorted PSF
- Need supercomputer to do this. Inference still challenging.





# Over-subtraction and self-subtraction

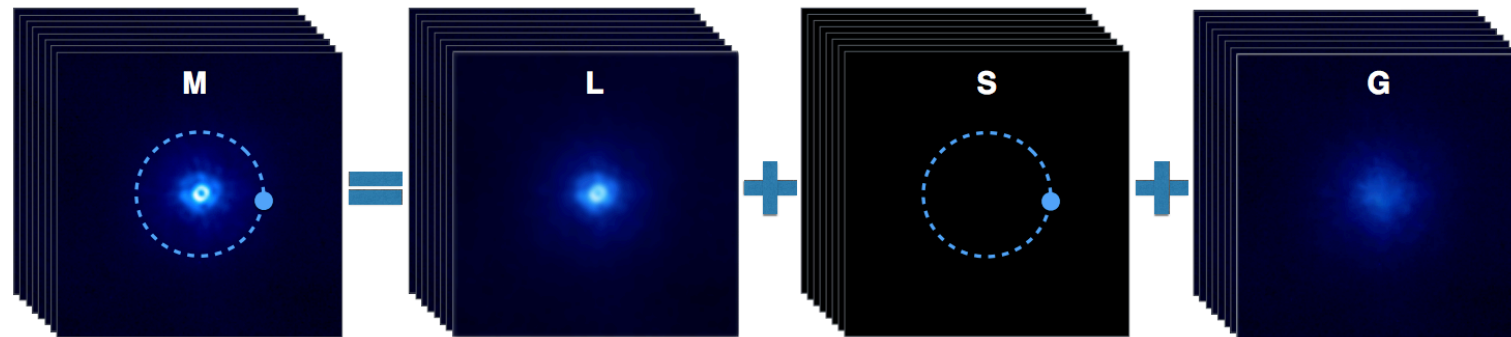
$$I_i(x, y) = \text{speckle}(x, y) + \text{planet}(x, y) + \text{noise}(x, y)$$

$$R_i = I_i - \sum_j (Z_j \cdot I_i) Z_j$$

- Over-subtraction
  - If the basis vectors “overlap” (in a multiplicative or dot-product sense), then some of the planet signal will be removed
  - This is nearly unavoidable
  - This is a linear operation
- Self-subtraction
  - If the basis vectors are generated from a dataset that contains the signal, then the basis vectors will contain the signal as well
  - This is avoidable using reference differential imaging or similar tricks
  - This is a nonlinear operation

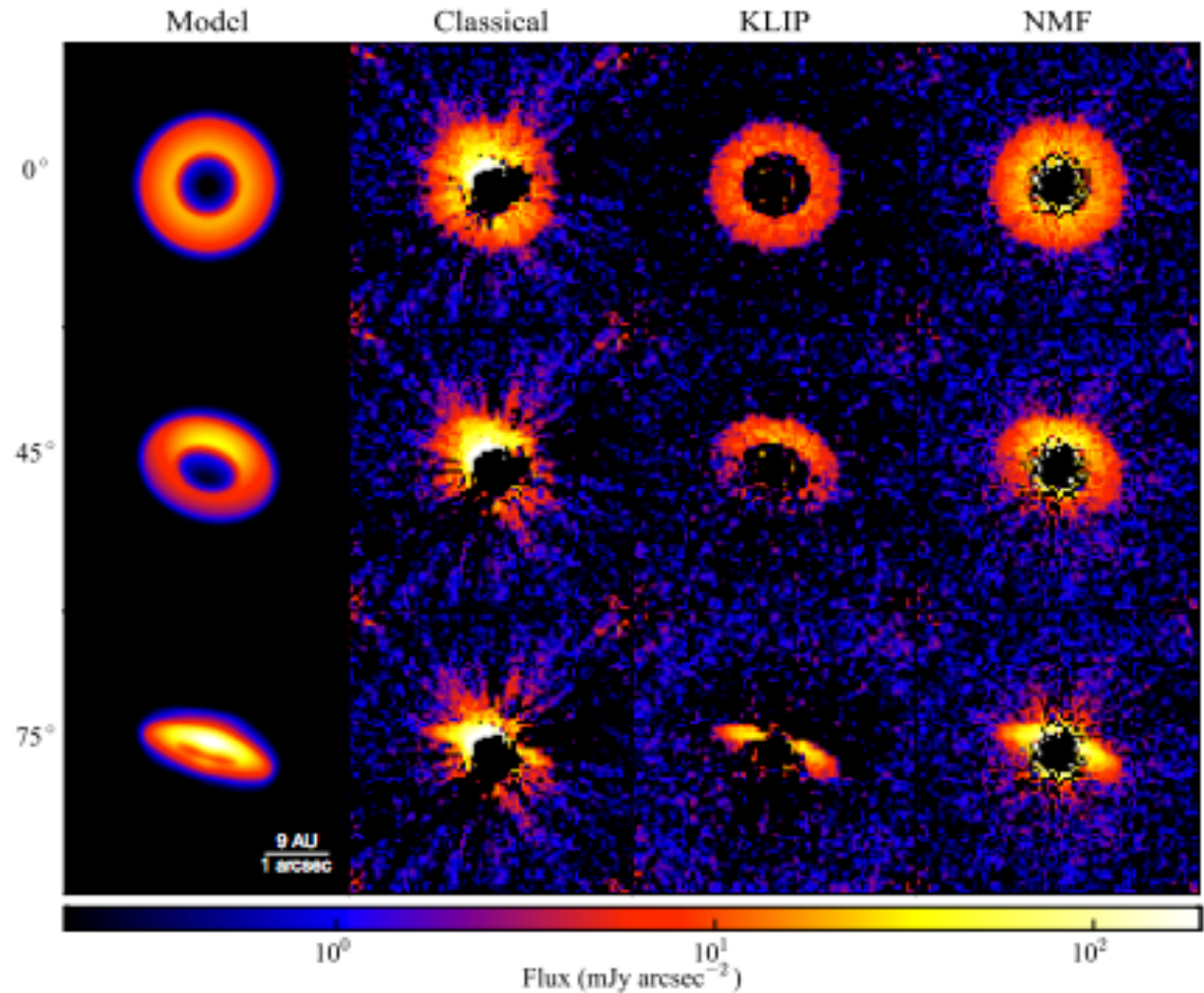
# Other recent developments

- **Local low rank, sparse, and Gaussian Decomposition (LLSG, Gomez-Gonzalez et al. 2016)**
  - Decomposes datacube into low-rank term (speckles), sparse term (planet), extra Gaussian noise
  - Seems to outperform PCA and nearly as fast
  - No planet model possible—no inference possible
  - Has extra free parameters



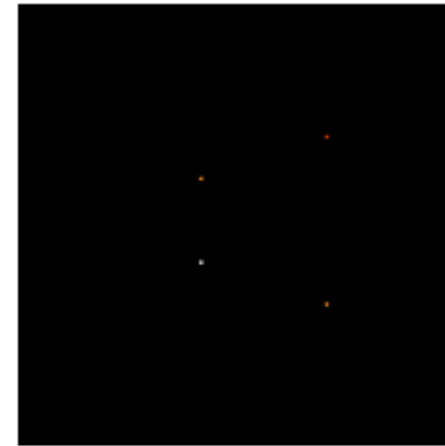
# Other recent developments

- **Non-negative matrix factorization (NMF, Rén et al. 2018)**
  - Decompose into a weight matrix and feature matrix, all positive
    - More physical for imaging data (intensity always positive)
  - More accurate for disks; much less over-subtraction
  - No way to include model of disk? —no inference possible
  - Useful for planets?

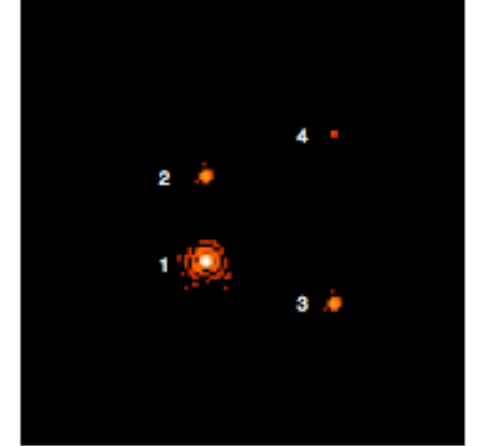


# Other recent developments

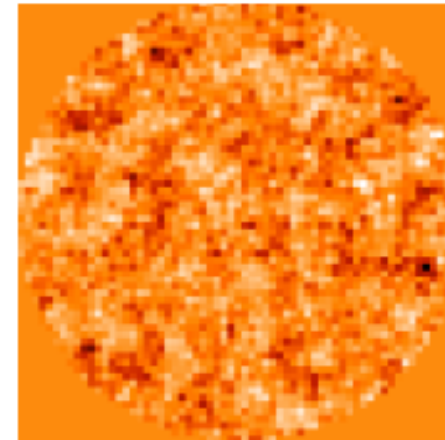
- **Simultaneous exoplanet detection and instrument aberration retrieval (Ygouf et al. 2013)**
  - Simultaneously model the image aberrations in the *pupil plane* and planet *in the focal plane* (!)
  - Uses real physics, makes sense, inference-friendly
  - Only works on simulated data; not clear how well it will work with horrible ground-based data
  - Requires multichannel data



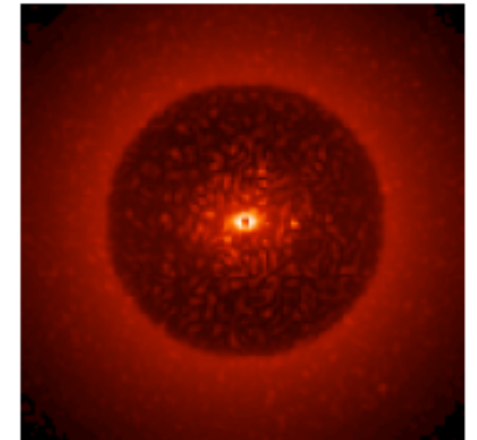
(a) Object map



(b) Image of the object map in the focal plane



(c) Aberration map



(d) Image of the speckle field in the focal plane

# Summary

- High contrast imaging data is awful
- Multiple algorithms exist to reduce the data
  - Most based on matrix factorization schemes
  - Few are based on physics
  - Inference hard
- Challenging to protect planet/disk signal from reduction algorithms
  - Removed by overfitting
  - Included in speckle components



# Joint data and model approach

- Joint model + systematics

$$y = A \cdot w + \mu(\theta) + \text{noise}$$

- Note this is for one frame, to keep things simple

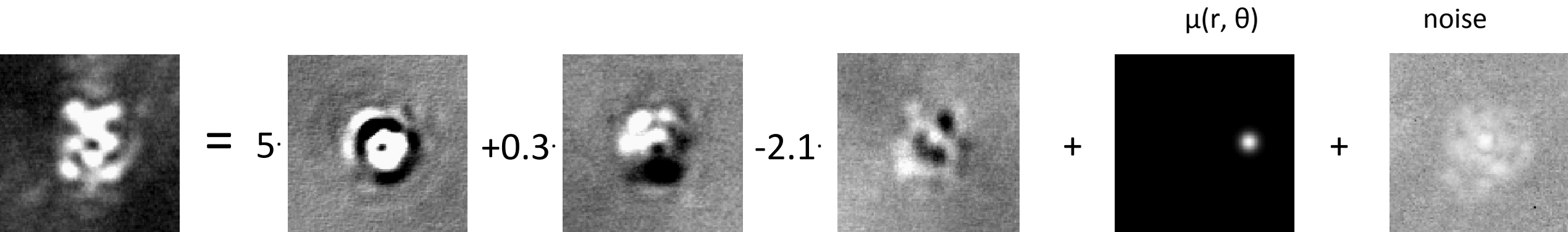
Parameter	Shape	Typical	Description
$y$	Npix x 1	10000 x 1	A single data frame
$A$	Npix x m	10000 x 10	The systematics design matrix
$\cdot$	N/A	N/A	Matrix dot product
$w$	m x 1	10	The weight applied to each vector in the design matrix to reproduce systematics
$\mu$	Npix	10000	Your model
$\theta$	?	3	The number of parameters in your model (x, y, amplitude)
noise	Npix	10000	Extra noise (such as read noise, sky noise, whatever)

# Data and model

- The systematics design matrix  $A$  is a collection of the systematics vectors  $\{V\}$ .  $\{V\}$  could be:
  - The principal components of your data
  - The sparse decomposition of your data (LLSG)
  - The non-negative matrix factorization data (NNMF)
  - ...or include it as free parameters
- $A \cdot w$  describes the systematics in a single dataframe
- $\mu(r, \theta)$  describes your signal

$$y = A \cdot w + \mu(\theta) + \text{noise}$$

$$A = \begin{pmatrix} \uparrow & \uparrow & \uparrow & \dots \\ v1 & v2 & v3 & \dots \\ \downarrow & \downarrow & \downarrow & \dots \end{pmatrix} \quad w = \begin{pmatrix} 5 \\ 0.3 \\ -2.1 \\ \vdots \end{pmatrix}$$



# Put into a probabilistic model

$$p(y|\theta, w) = \mathcal{N}(y; Aw + \mu(\theta), C)$$
$$= \frac{\exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{A} \cdot \mathbf{w} - \mu(\theta))^T \mathbf{C}^{-1}(\mathbf{y} - \mathbf{A} \cdot \mathbf{w} - \mu(\theta))\right)}{\sqrt{(2\pi)^k |\mathbf{C}|}}$$

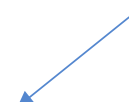
Parameter	Shape	Typical	Description
$p(y \theta, w)$	1 (scalar)	1	Probability of your data given your model parameters and weights
$y$	Npix x 1	10000 x 1	A single data frame
$A$	Npix x m	10000 x 10	The systematics design matrix
$w$	m x 1	10	The weight applied to each vector in the design matrix to reproduce systematics
$\mu$	Npix	10000	Your model
$\theta$	?	3	The number of parameters in your model (x, y, amplitude)
$C$	Npix x Npix	10000x 10000	Noise covariance matrix (may be assumed to be diagonal)

# Finally: inference

- The expression

$$p(y|\theta) = \mathcal{N}(y; \mu(\theta), C + A\Lambda A^T)$$

*weight covariance matrix  
incorporates uncertainties  
in the weights*



is almost what we want. For inference, we want  $p(\boldsymbol{\vartheta}|\mathbf{y})$ . You can get this by multiplying by a prior on your model parameters.

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

- For example, here is a model/prior I will use later

Shifted-scaled PSF model

$$\mu(x_0, y_0, a) = a \cdot \text{PSF}[x - x_0, y - y_0]$$

Uniform prior on position

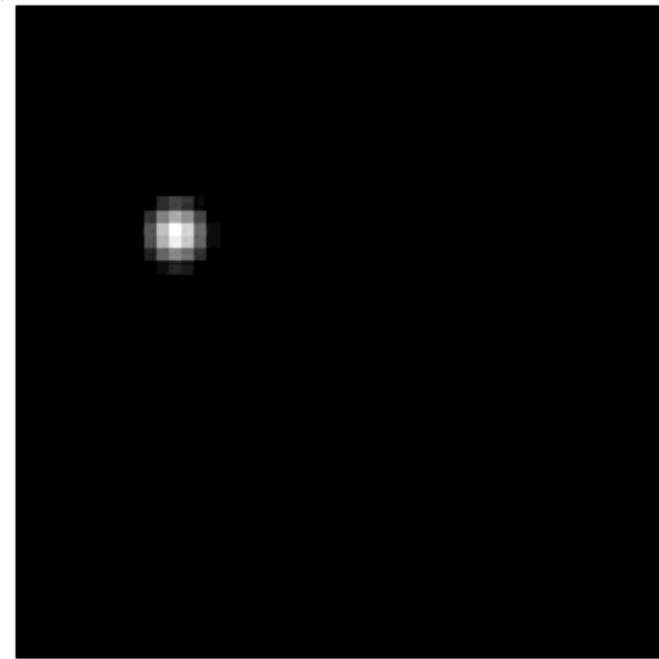
$$p(x_0), p(y_0) \propto 1$$

Scale-invariant prior on amplitude

$$p(a) \propto 1/a$$

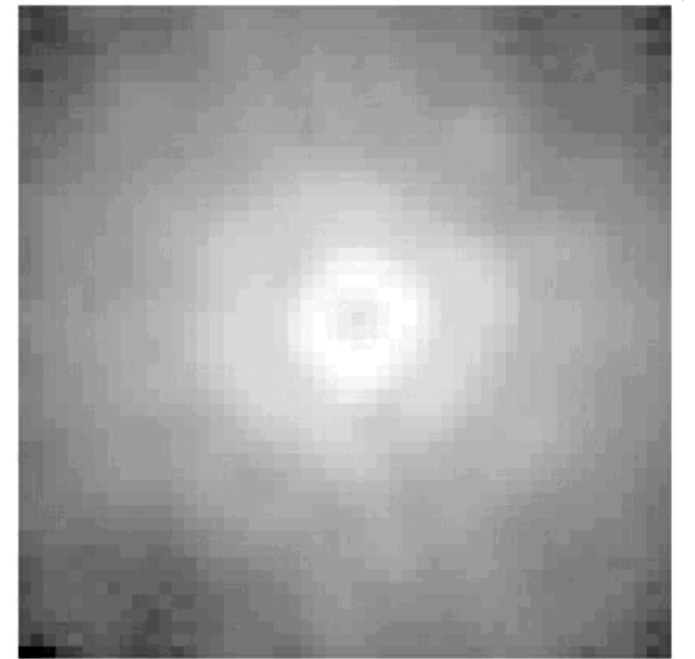
# 1) ADI with fake signal injected after

- “Perfect” ADI case
  - Calculate  $\mathbf{A}$ ,  $\mathbf{\Lambda}$ ,  $\mathbf{C}$  w/out planet in data
- Should perfectly reproduce injected signal (within uncertainties)



Injected signal

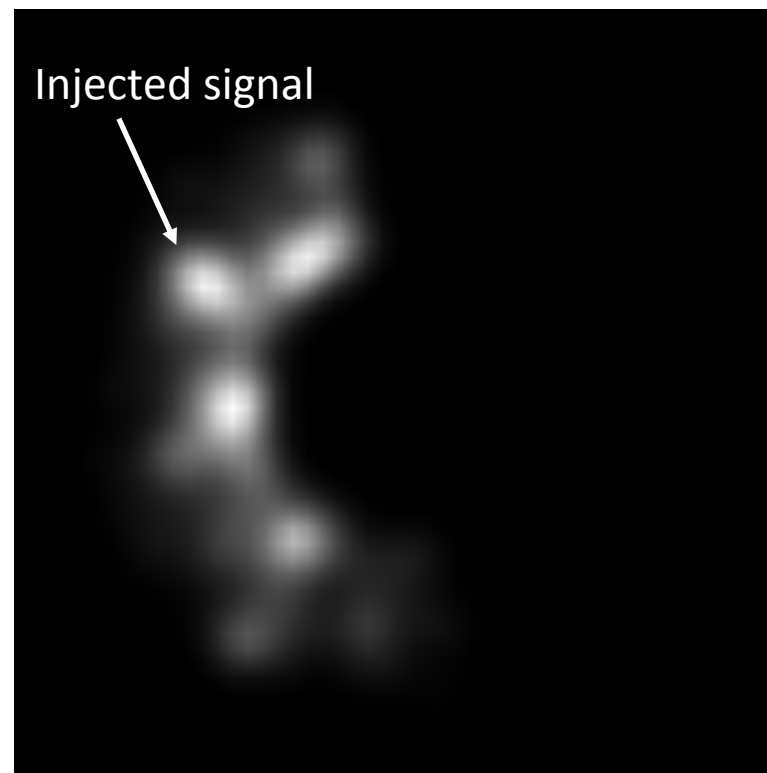
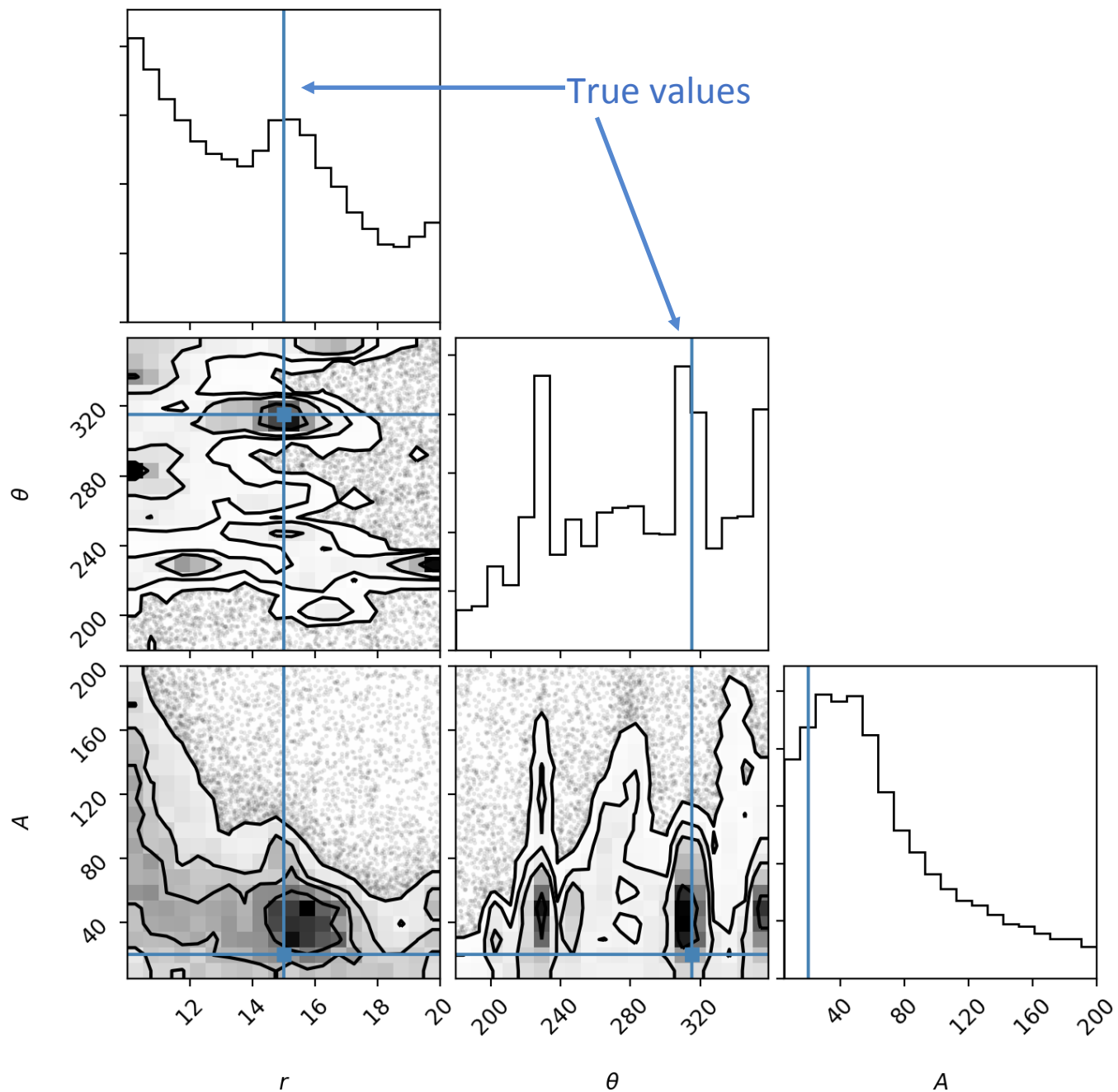
Beta Pictoris dataset



Signal + speckles  
(note signal not visible)

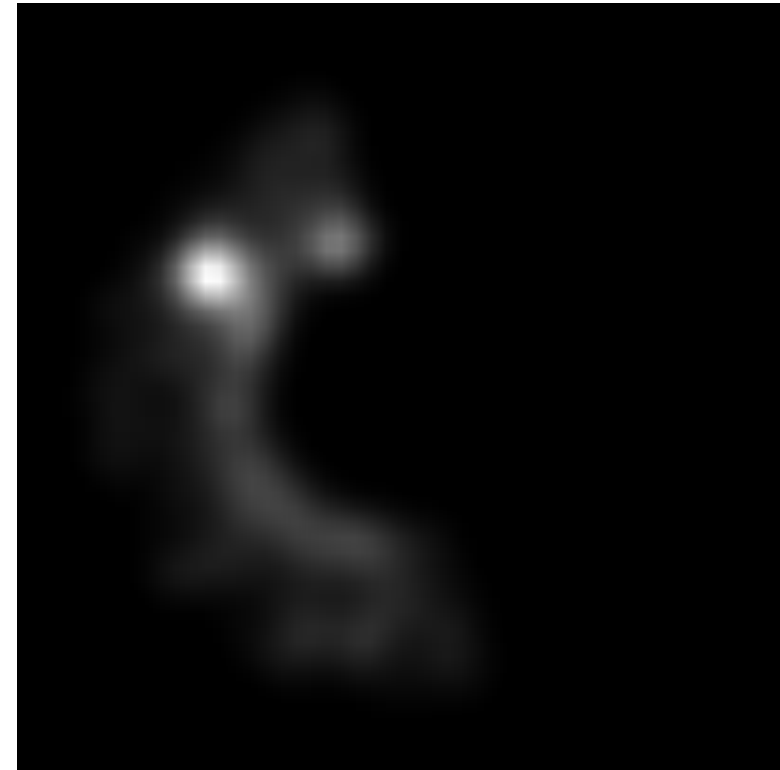
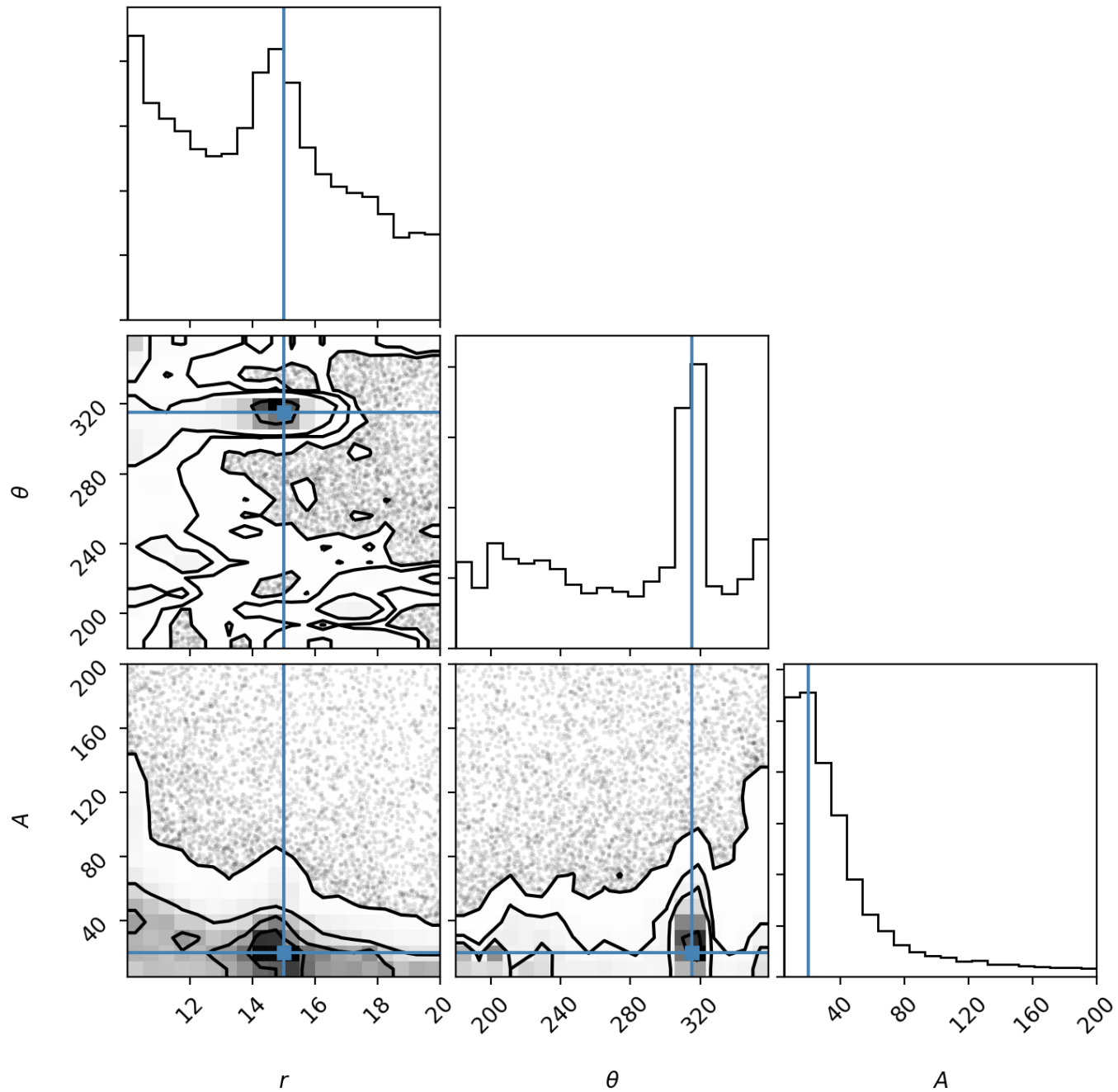


# 2 components



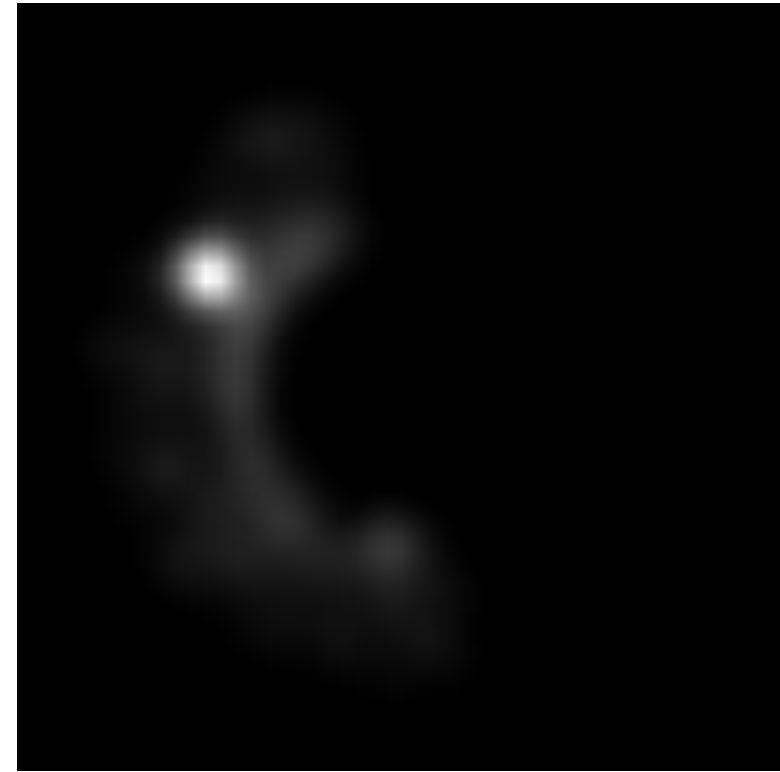
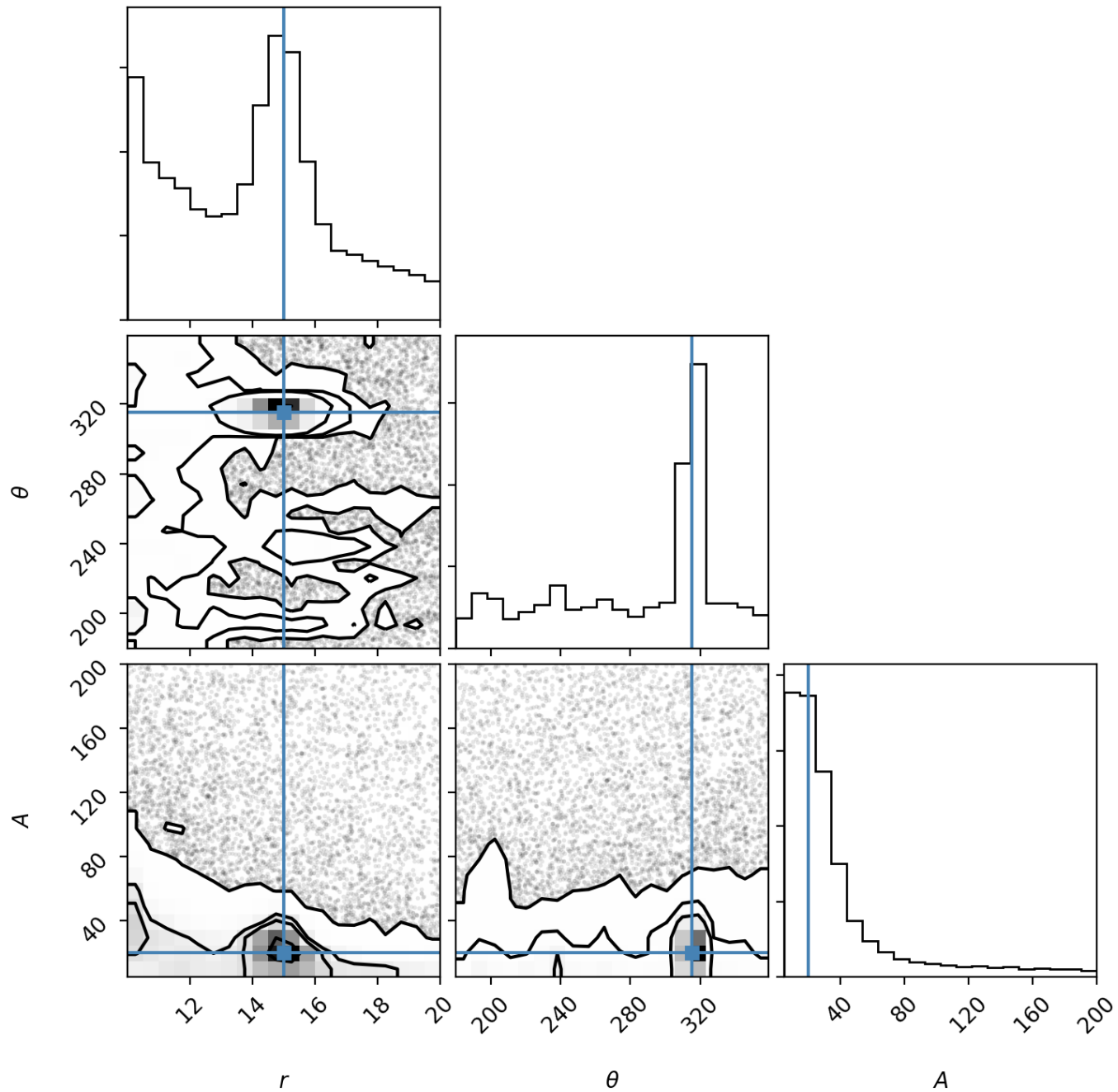
Probability map

5 components



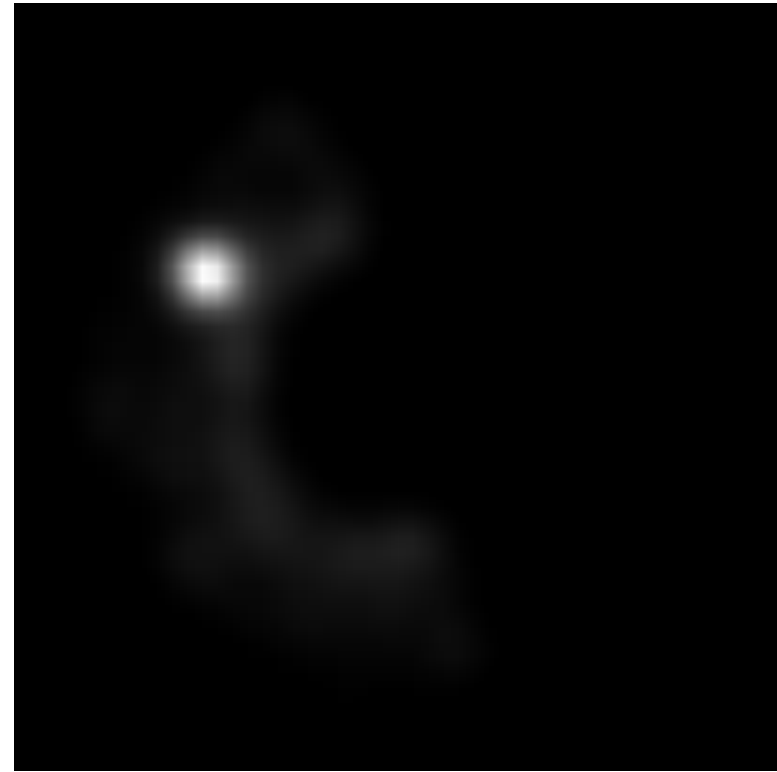
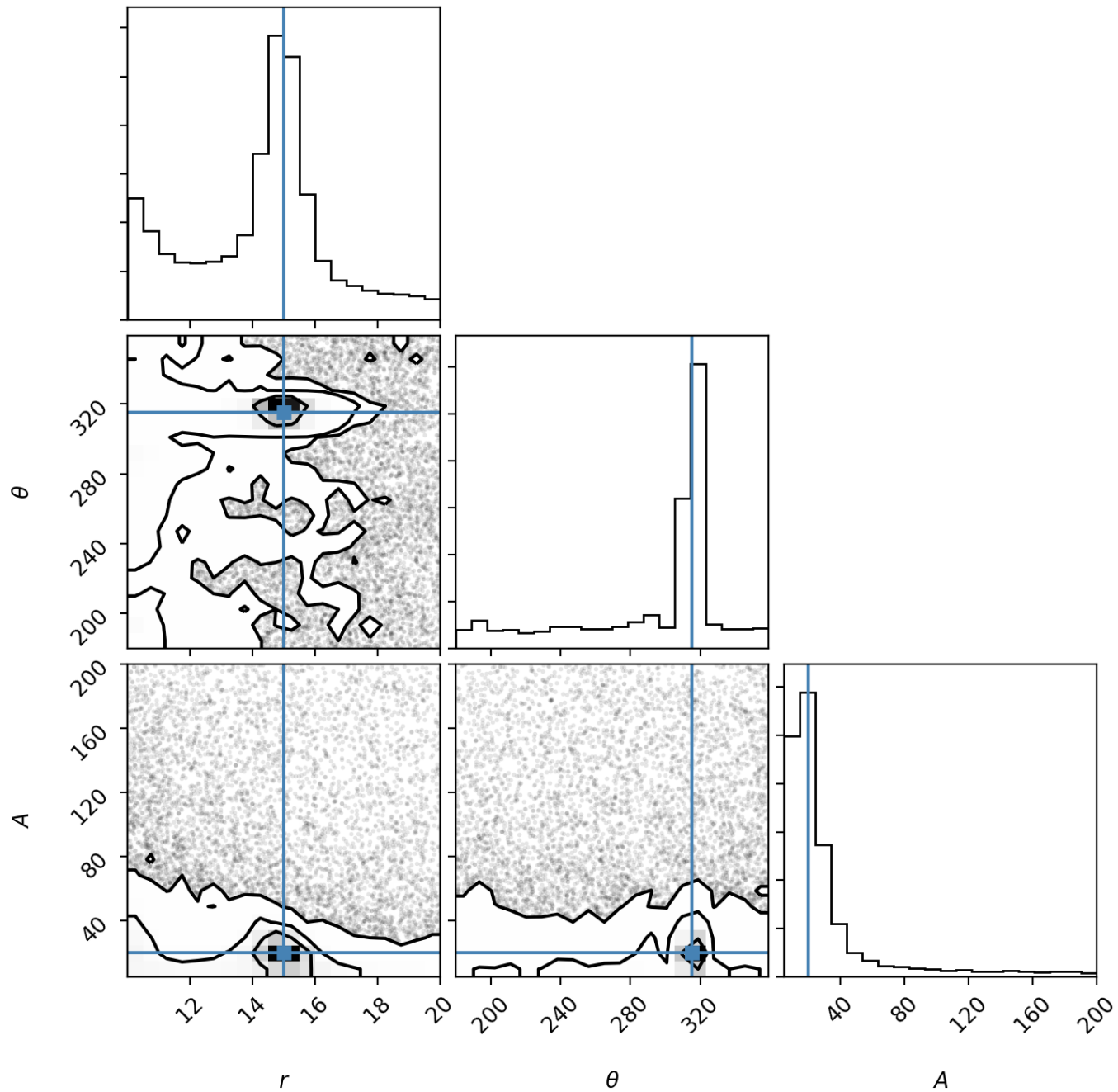
Probability map

10 components



Probability map

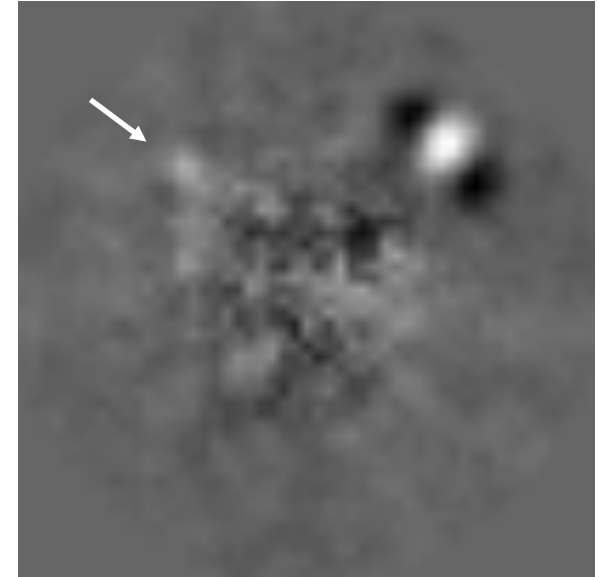
20 components



Probability map

# Review of common algorithms

- Sources in data will self-subtract if they overlap with the basis vectors
  - Have to inject fake companions to figure out how bad this effect is
- Where to stop? How many coefficients/components?
- Noise propagation? Error bars?
- Disks?
- What am I looking at? Is it real? Model selection?





# Issues with LOCI, PCA, etc

- Self subtraction
  - The basis vectors will “overlap” with the signal in the image, so as you increase the number of basis vectors the signal gets removed more and more
  - If the signal is rotating, low amounts of rotation will lead to “rotational” self-subtraction
- Throughput calibration
  - Fake signals need to be injected in the data to tell what the algorithm is doing
  - Negative fake signals are used to figure out photometry and astrometry
- Very hard to tell what you are looking at

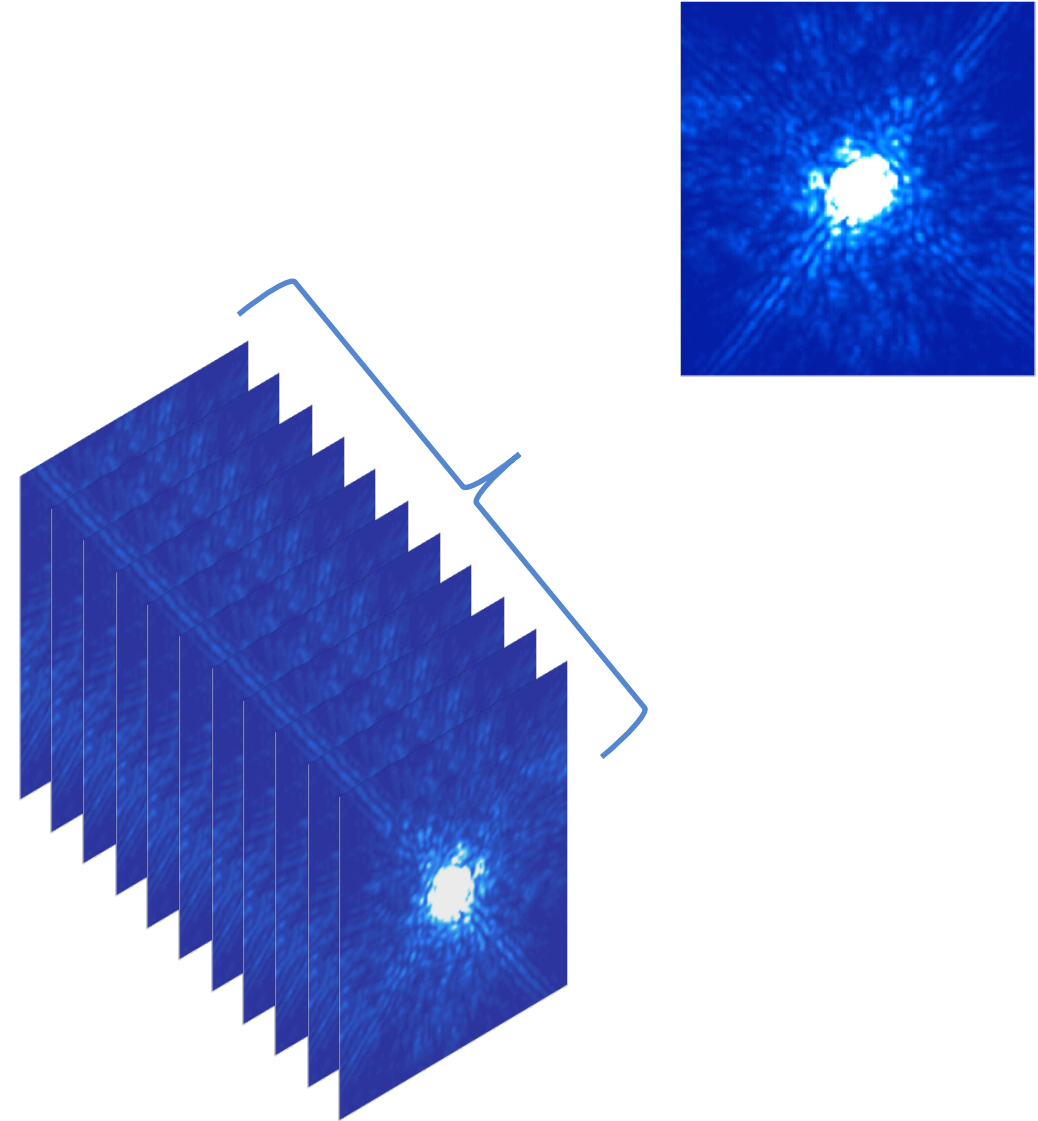
# Review of common algorithms: LOCI

- For each (sub)image  $I_i$  in your datacube
- Try to reproduce it by minimizing

$$R_i = I_i - \sum_j \alpha_{ij} I_j$$

using least-squares

- Replace  $I_j$  with a reference  $I'_j$  if using RDI
- Derotate the residuals  $R_i$  and median along the time axis



Bayesian inference provides a principled approach to the inference of a set of parameters,  $\Theta$ , in a model (or hypothesis),  $H$ , for data,  $\mathbf{D}$ . Bayes' theorem states that

$$\Pr(\Theta|\mathbf{D}, H) = \frac{\Pr(\mathbf{D}|\Theta, H) \Pr(\Theta|H)}{\Pr(\mathbf{D}|H)}, \quad (1)$$

where  $\Pr(\Theta|\mathbf{D}, H) \equiv P(\Theta|\mathbf{D})$  is the posterior probability density of the model parameters,  $\Pr(\mathbf{D}|\Theta, H) \equiv \mathcal{L}(\Theta)$  the likelihood of the data, and  $\Pr(\Theta|H) \equiv \pi(\Theta)$  the parameter prior. The final term,  $\Pr(\mathbf{D}|H) \equiv \mathcal{Z}$  (the Bayesian evidence), represents the factor required to normalize the posterior over the domain of  $\Theta$  given by:

$$\mathcal{Z} = \int_{\Omega_{\Theta}} \mathcal{L}(\Theta) \pi(\Theta) d\Theta. \quad (2)$$

Being independent of the parameters, however, this factor can be ignored in parameter inference problems which can be approximated by taking samples from the unnormalized posterior only, using standard MCMC methods (for instance).

Model selection between two competing models,  $H_0$  and  $H_1$ , can be achieved by comparing their respective posterior probabilities given the observed dataset as follows:

$$R = \frac{\Pr(H_1|\mathbf{D})}{\Pr(H_0|\mathbf{D})} = \frac{\Pr(\mathbf{D}|H_1) \Pr(H_1)}{\Pr(\mathbf{D}|H_0) \Pr(H_0)} = \frac{\mathcal{Z}_1 \Pr(H_1)}{\mathcal{Z}_0 \Pr(H_0)}. \quad (3)$$

# Joint data and model approach

- Joint model + systematics

$$y = A \cdot w + \mu(\theta) + \text{noise}$$

- Note this is for one frame, to keep things simple

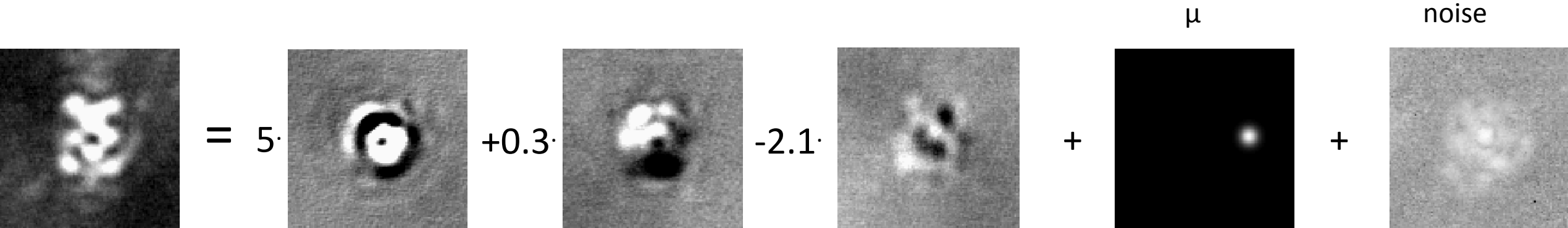
Parameter	Shape	Typical	Description
$y$	Npix x 1	10000 x 1	A single data frame
$A$	Npix x m	10000 x 10	The systematics design matrix
$\cdot$	N/A	N/A	Matrix dot product
$w$	m x 1	10	The weight applied to each vector in the design matrix to reproduce systematics
$\mu$	Npix	10000	Your model
$\theta$	?	3	The number of parameters in your model (x, y, amplitude)
noise	Npix	10000	Extra noise (such as read noise, sky noise, whatever)

# Data and model

- Joint model + systematics

$$y = A \cdot w + \mu(\theta) + \text{noise}$$

$$w = [5, 0.3, 2.1]$$



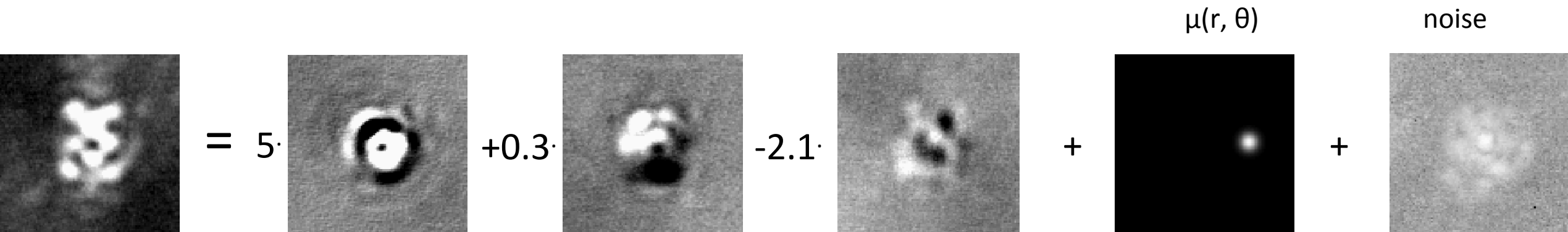


# Data and model

- The systematics design matrix  $A$  is a collection of the systematics vectors  $\{V\}$ .  $\{V\}$  could be:
  - The principal components of your data
  - The sparse decomposition of your data (LLSG)
  - The non-negative matrix factorization data (NNMF)
  - ...or include it as free parameters
- $A \cdot w$  describes the systematics in a single dataframe
- $\mu(r, \theta)$  describes your signal

$$y = A \cdot w + \mu(\theta) + \text{noise}$$

$$A = \begin{pmatrix} \uparrow & \uparrow & \uparrow & \dots \\ v1 & v2 & v3 & \dots \\ \downarrow & \downarrow & \downarrow & \dots \end{pmatrix} \quad w = \begin{pmatrix} 5 \\ 0.3 \\ -2.1 \\ \vdots \end{pmatrix}$$



# Put into a probabilistic model

$$p(y|\theta, w) = \mathcal{N}(y; Aw + \mu(\theta), C)$$
$$= \frac{\exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{A} \cdot \mathbf{w} - \mu(\theta))^T \mathbf{C}^{-1}(\mathbf{y} - \mathbf{A} \cdot \mathbf{w} - \mu(\theta))\right)}{\sqrt{(2\pi)^k |\mathbf{C}|}}$$

Parameter	Shape	Typical	Description
$p(y \theta, w)$	1 (scalar)	1	Probability of your data given your model parameters and weights
$y$	Npix x 1	10000 x 1	A single data frame
$A$	Npix x m	10000 x 10	The systematics design matrix
$w$	m x 1	10	The weight applied to each vector in the design matrix to reproduce systematics
$\mu$	Npix	10000	Your model
$\theta$	?	3	The number of parameters in your model (x, y, amplitude)
$C$	Npix x Npix	10000x 10000	Noise covariance matrix (may be assumed to be diagonal)

# Put into a probabilistic model

$$p(y|\theta, w) = \mathcal{N}(y; Aw + \mu(\theta), C)$$

- Notice this is not making an assumption about “speckle noise being normally distributed” or anything like that which is obviously wrong. The speckles are described by some combination  $w$  of feature vectors  $A$
- The assumption is the “rest of the noise” after removal of speckles can be described by a Gaussian with a covariance matrix  $C$

# No one cares about the weights

$$p(y|\theta, w) = \mathcal{N}(y; Aw + \mu(\theta), C)$$

- We don't care about  **$p(\mathbf{y}|\boldsymbol{\vartheta}, \mathbf{w})$** . We really want  **$p(\mathbf{y}|\boldsymbol{\vartheta})$** 
  - The weights  **$\mathbf{w}$**  do not contain anything remotely interesting
- Would be nice to get rid of them...let's invent a prior

$$p(w) = \mathcal{N}(w; 0, \Lambda)$$

- Assume a Gaussian prior on the weights with *zero mean* and covariance matrix  **$\Lambda$** 
  - **$\Lambda$**  is an *m*×*m* matrix
- Then marginalize them out

$$p(y|\theta) = \int_{-\infty}^{\infty} p(w)p(y|\theta, w)dw$$

...a miracle occurs

$$p(y|\theta, w) = \mathcal{N}(y; Aw + \mu(\theta), C)$$

$$p(y|\theta) = \int_{-\infty}^{\infty} p(w)p(y|\theta, w)dw$$

- This is a nasty **m**-dimensional integral and bad news if you try and do it with a computer. But it actually has an analytic solution!

$$p(y|\theta) = \int_{-\infty}^{\infty} p(w)p(y|\theta, w)dw = \mathcal{N}(y; \mu(\theta), C + A\Lambda A^T)$$

- This takes about a page of linear algebra to show. See Luger et al 2016

# Finally: inference

- The expression

$$p(y|\theta) = \mathcal{N}(y; \mu(\theta), C + A\Lambda A^T)$$

is almost what we want. For inference, we want  $p(\boldsymbol{\vartheta}|\mathbf{y})$ . You can get this by multiplying by a prior on your model parameters.

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

- For example, here is a model/prior I will use later

Shifted-scaled PSF model

$$\mu(x_0, y_0, a) = a \cdot \text{PSF}[x - x_0, y - y_0]$$

Uniform prior on position

$$p(x_0), p(y_0) \propto 1$$

Scale-invariant prior on amplitude

$$p(a) \propto 1/a$$



# How to calculate everything I talked about

- The design vectors and such depend on the particular reduction approach you use. Here is how to calculate everything *when you take the design vectors from PCA*.
- ***A (design matrix)***
  - Calculate the first  $m$  principal components of your datacube
- ***$\Lambda$  (weight covariance matrix)***
  - For each frame in your datacube, reduce it using ***A***
  - (this is just a dot product for PCA.)
  - Then find the covariances of each component weight (the means are automatically zero for PCA, and off-diagonal elements are also  $\sim 0$ )
- ***C (noise covariance matrix)***
  - Reduce the datacube using ***A***
  - don't derotate; make a pixel-by-pixel diagonal matrix of the variances
  - Multiply by  $\sim 2$  if you are paranoid about extra noise (fudge factor)

# Algorithm layout

- Calculate  **$\mathbf{A}$ ,  $\mathbf{C}$ ,  $\mathbf{\Lambda}$** ; choose a PSF model/priors
- For each frame of the datacube:
  - Look up the sky angle,  $\alpha$
  - Offset the PSF model  $(r, \theta, a) \rightarrow (r, \theta + \alpha, a)$ 
    - This tracks excess flux at a moving position
    - **No derotation here**
  - Sample from  $p(\theta|y) \propto p(y|\theta)p(\theta)$ 
    - This is the Gaussian from before
    - Only 3 parameters due to marginalization trick
    - ***Use nested sampling***
  - Save the samples
- Combine all the samples

# Complaints about this approach

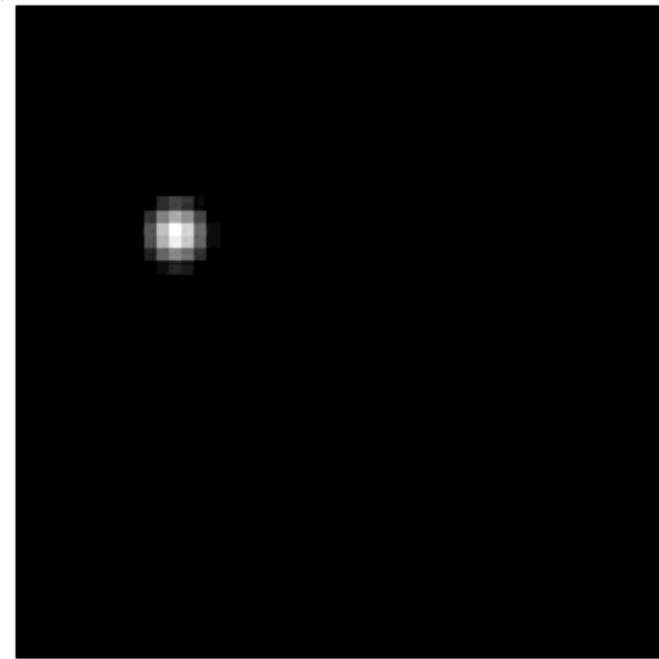
- *This is just PCA repackaged in confusing Bayesianism; there's nothing new here*
  - You can use PCA to generate the design vectors, or you can use anything you want. The point is that you can *include uncertainties in your systematics model* in your data analysis.
- *Other people already do forward modeling; nothing to see here folks*
  - No, other people forward model the effects of self subtraction on the PSF and then run an MCMC chain over the *reduced* data. This is operating on the *raw datacube*.
- *This will only work for point sources since you need to know the PSF*
  - If you don't have a model for your data, you're not doing inference. This will work on disks if you have a disk model.

# Comparisons

	Primary use	Data model needed?	Extra free parameters	Fast	Compatible with MCMC	Self-subtraction	<i>Can tell you whether a planet is there</i>
PCA, LLSG, NNMF, etc	Blind discovery	No	No	Yes	No	Always	<b>No</b>
With linear model	Discovery, characterization	Yes	Yes—as many as your model	Yes	Yes	Not really	<b>Yes</b>

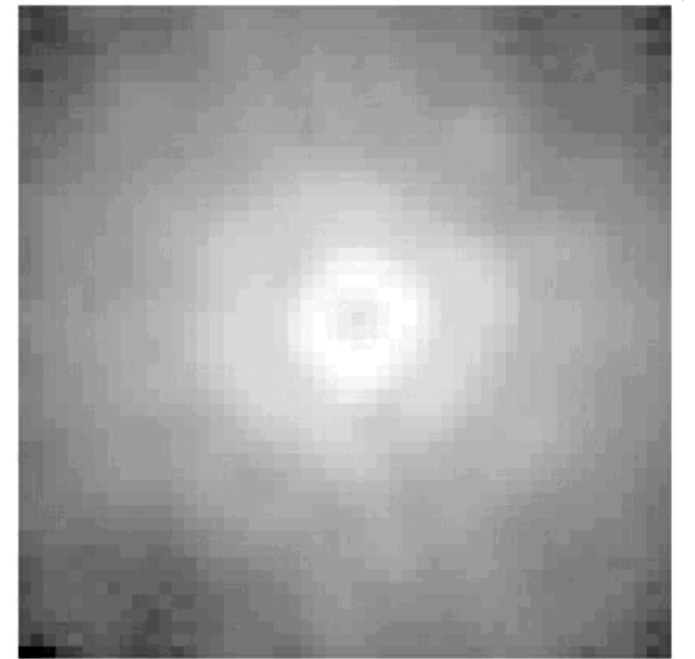
# 1) ADI with fake signal injected after

- “Perfect” ADI case
  - Calculate  $\mathbf{A}$ ,  $\mathbf{\Lambda}$ ,  $\mathbf{C}$  w/out planet in data
- Should perfectly reproduce injected signal (within uncertainties)



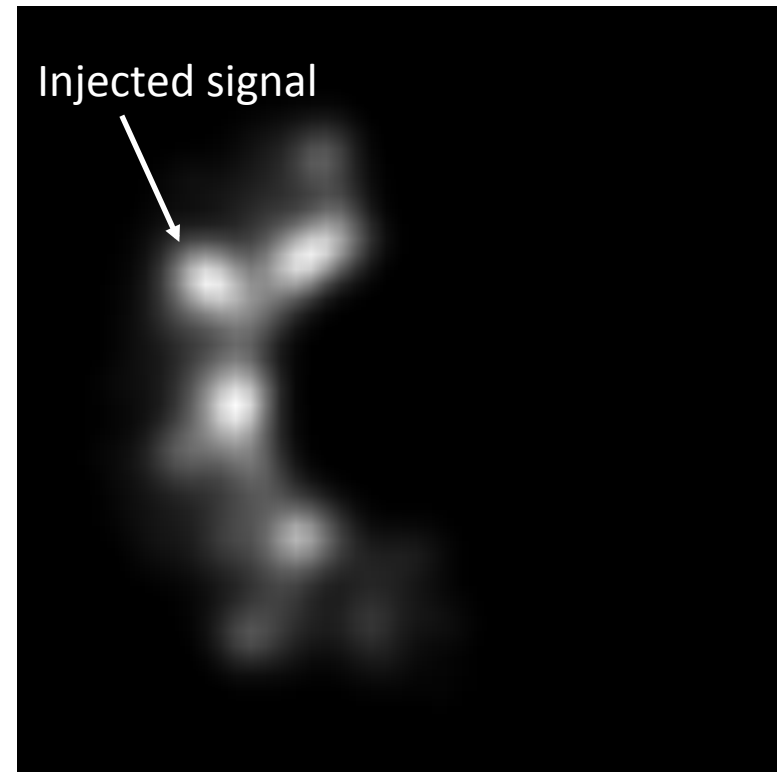
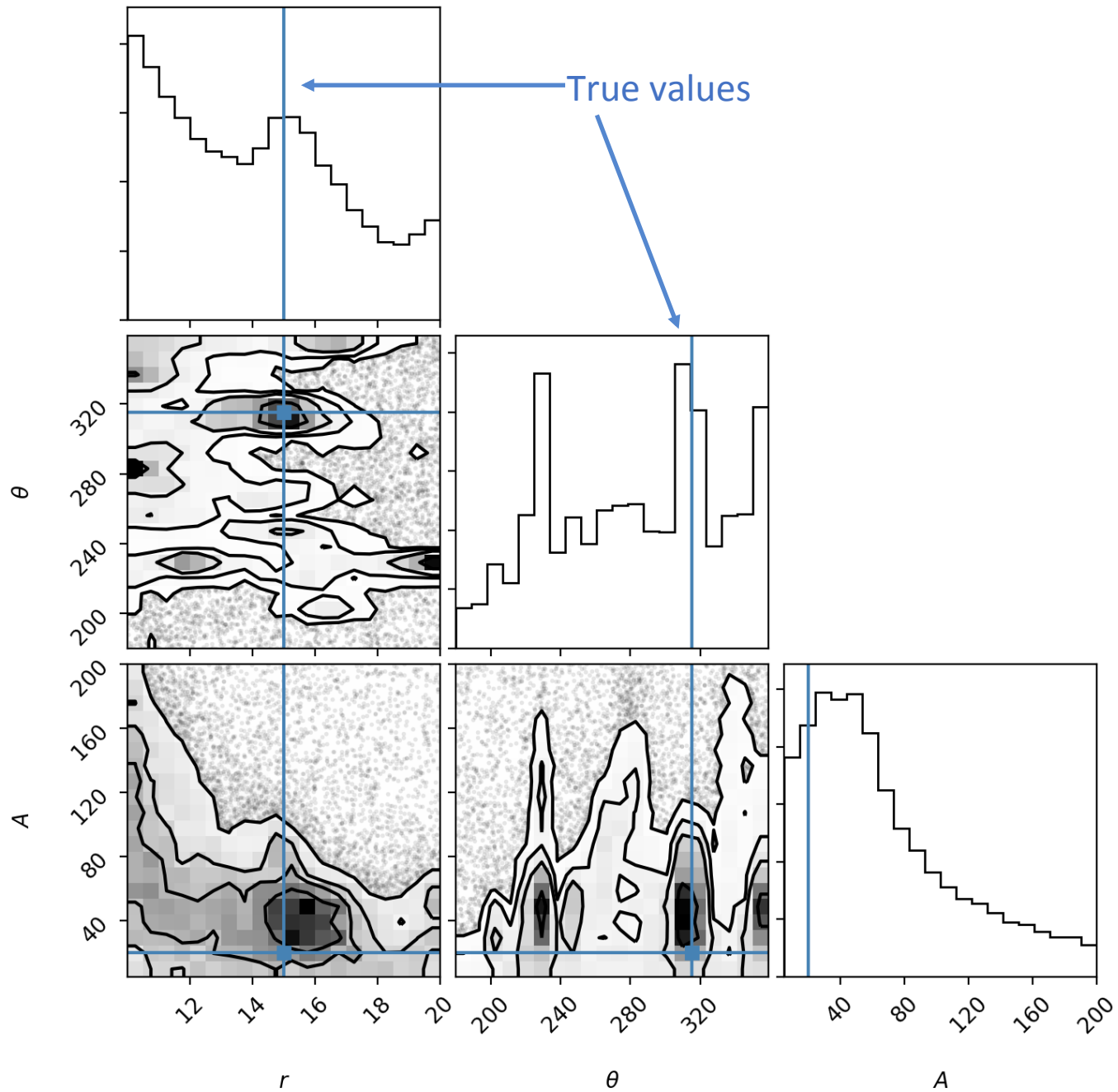
Injected signal

Beta Pictoris dataset



Signal + speckles  
(note signal not visible)

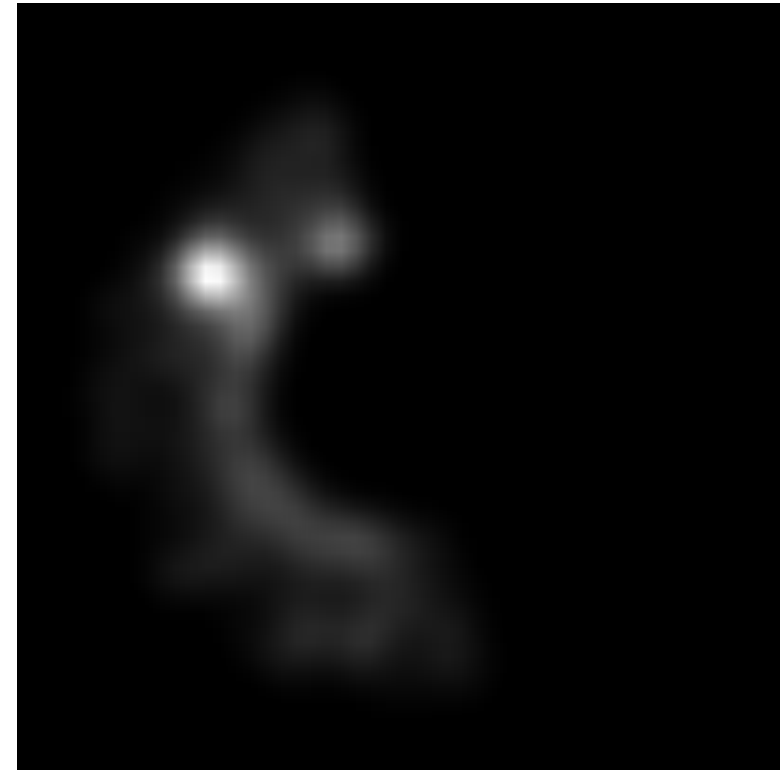
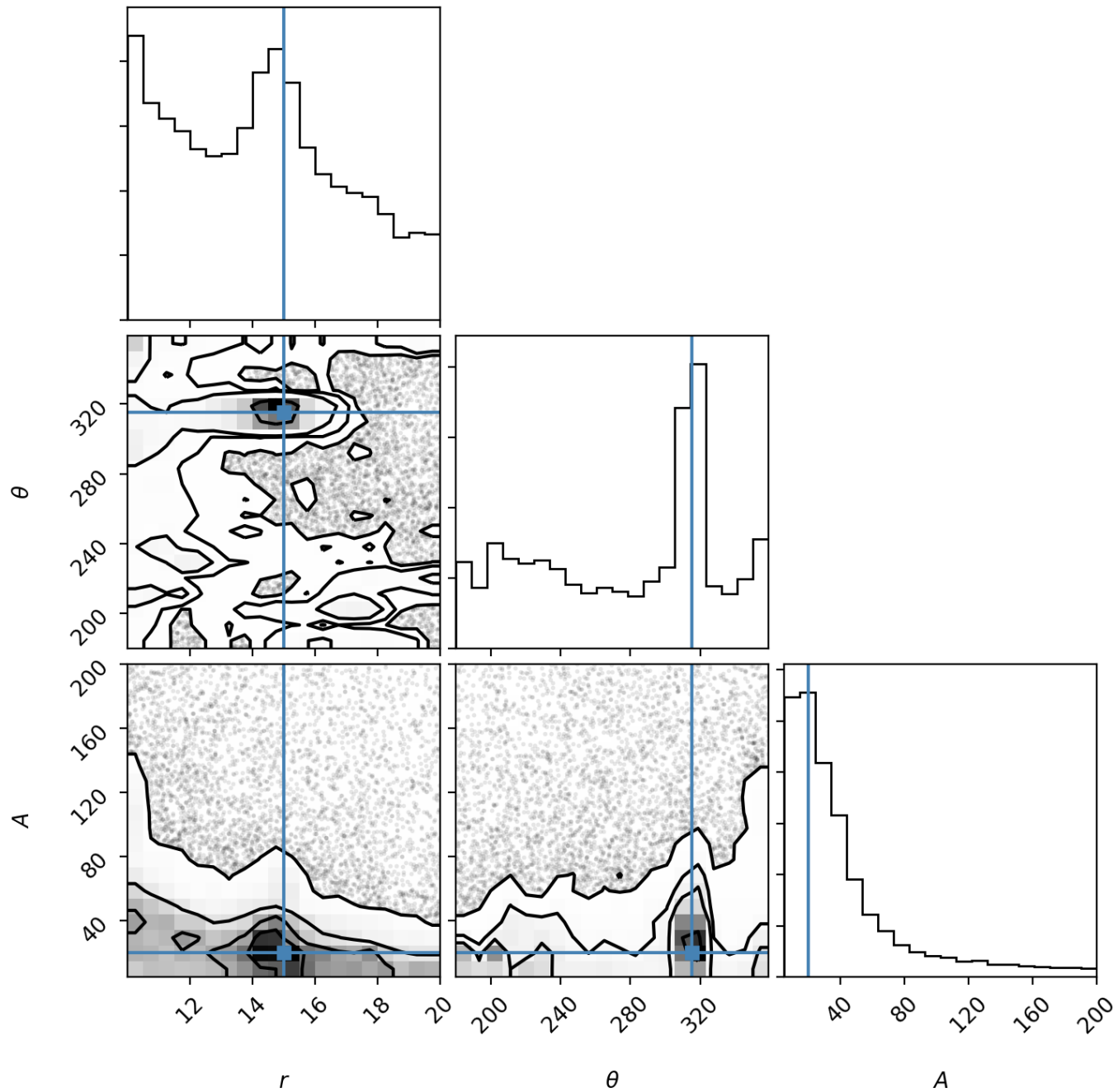
# 2 components



Probability map

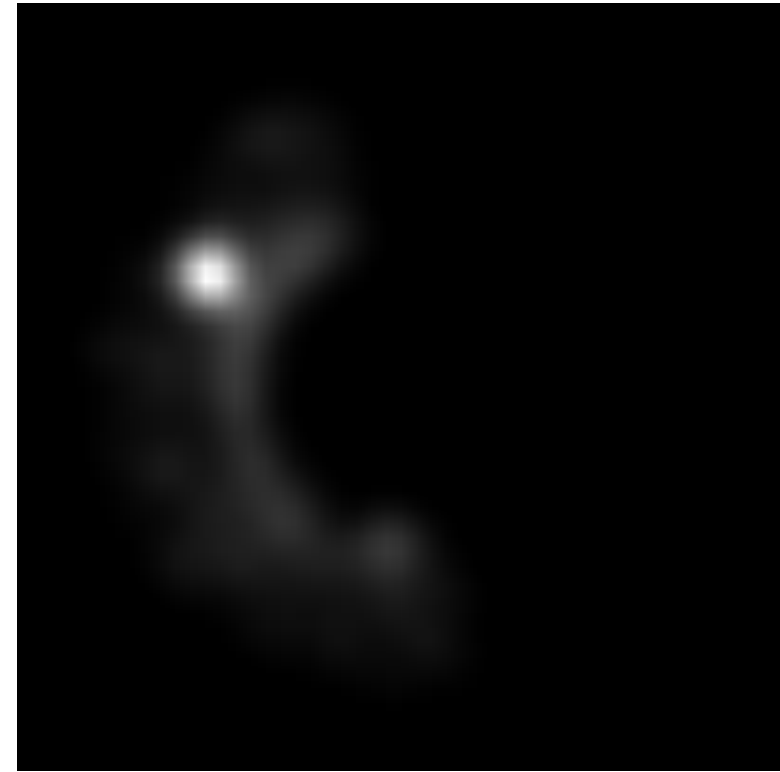
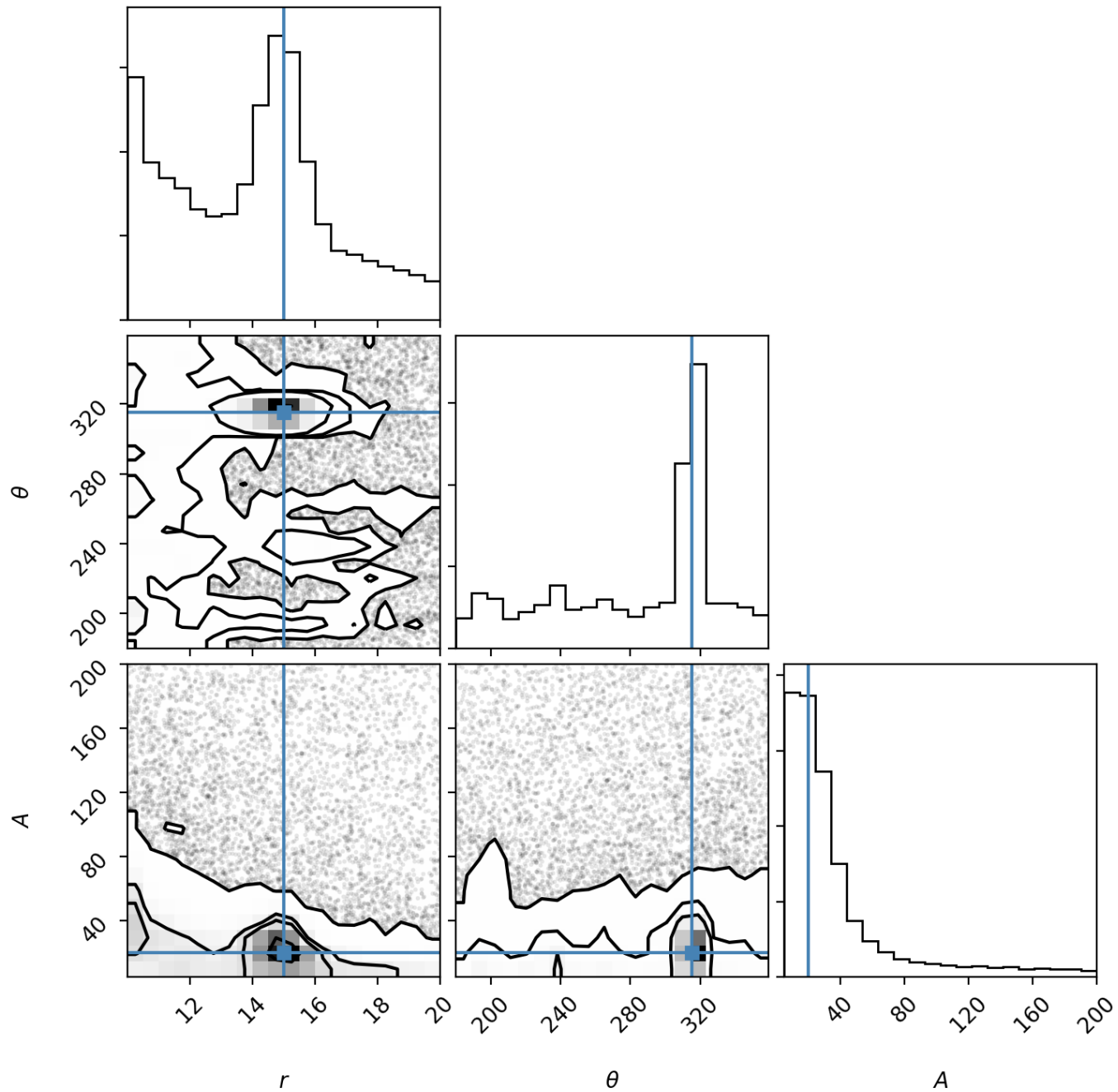


5 components



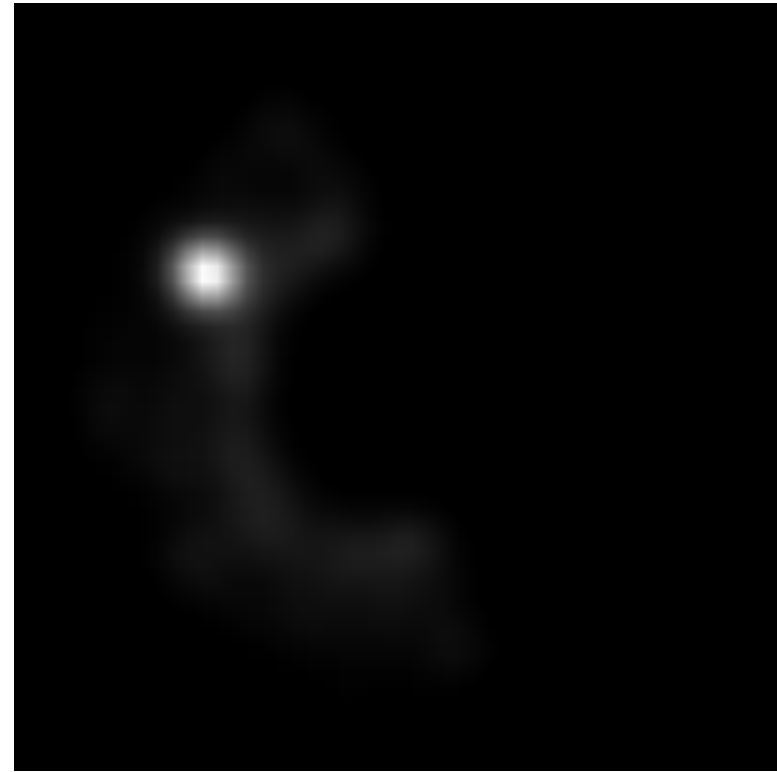
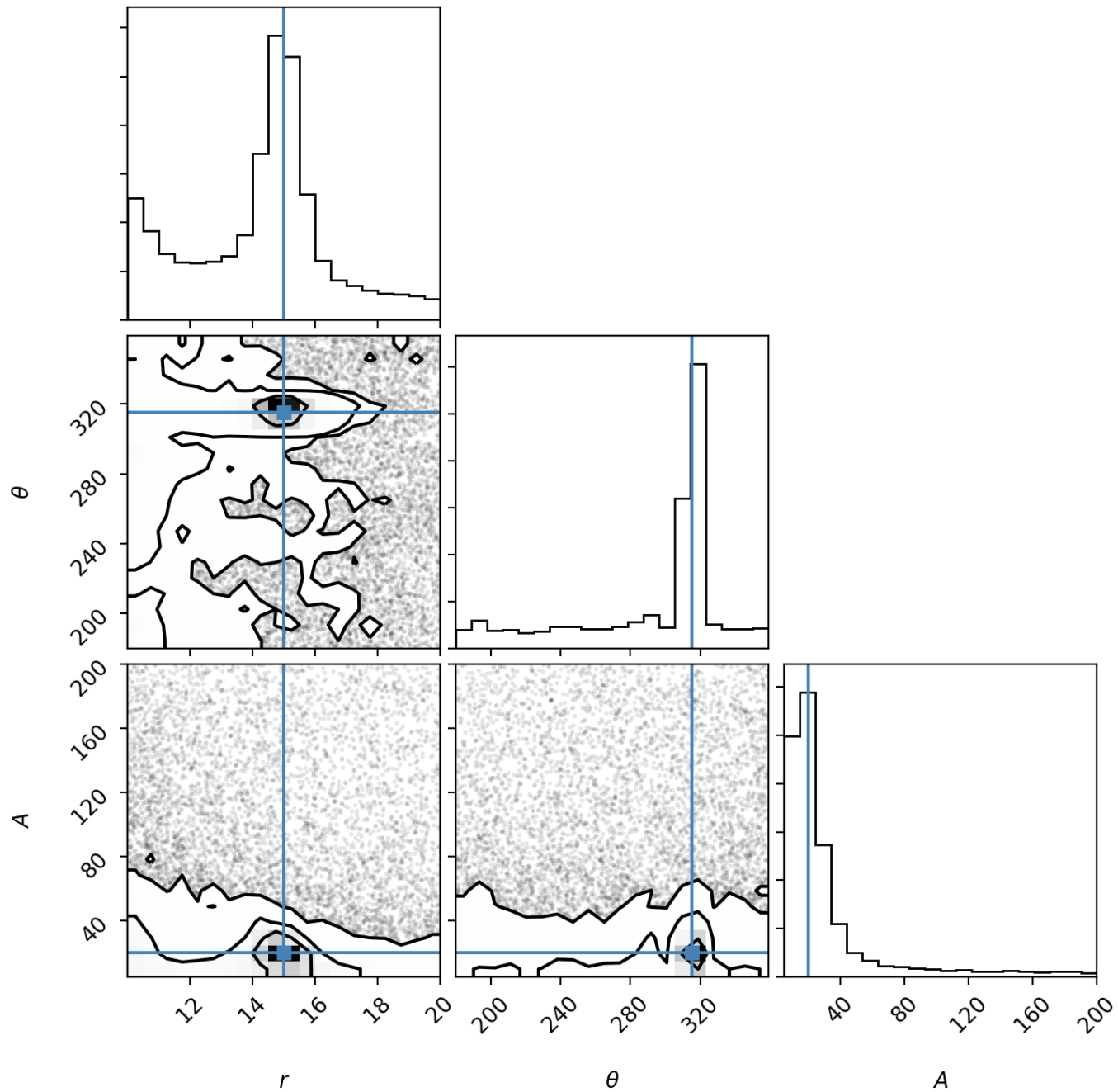
Probability map

10 components



Probability map

# 20 components



Probability map

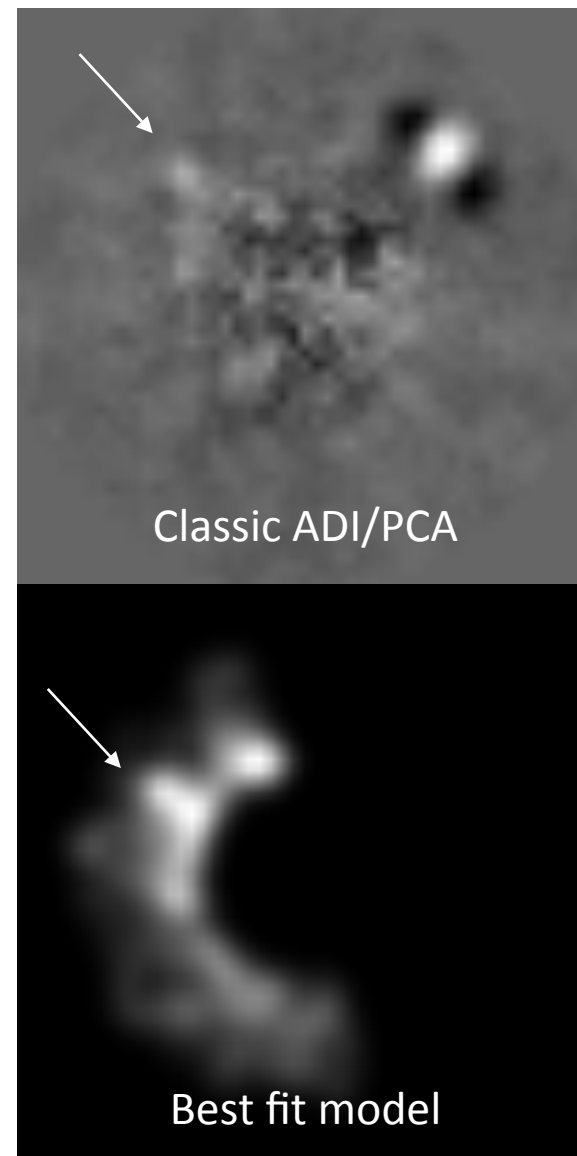
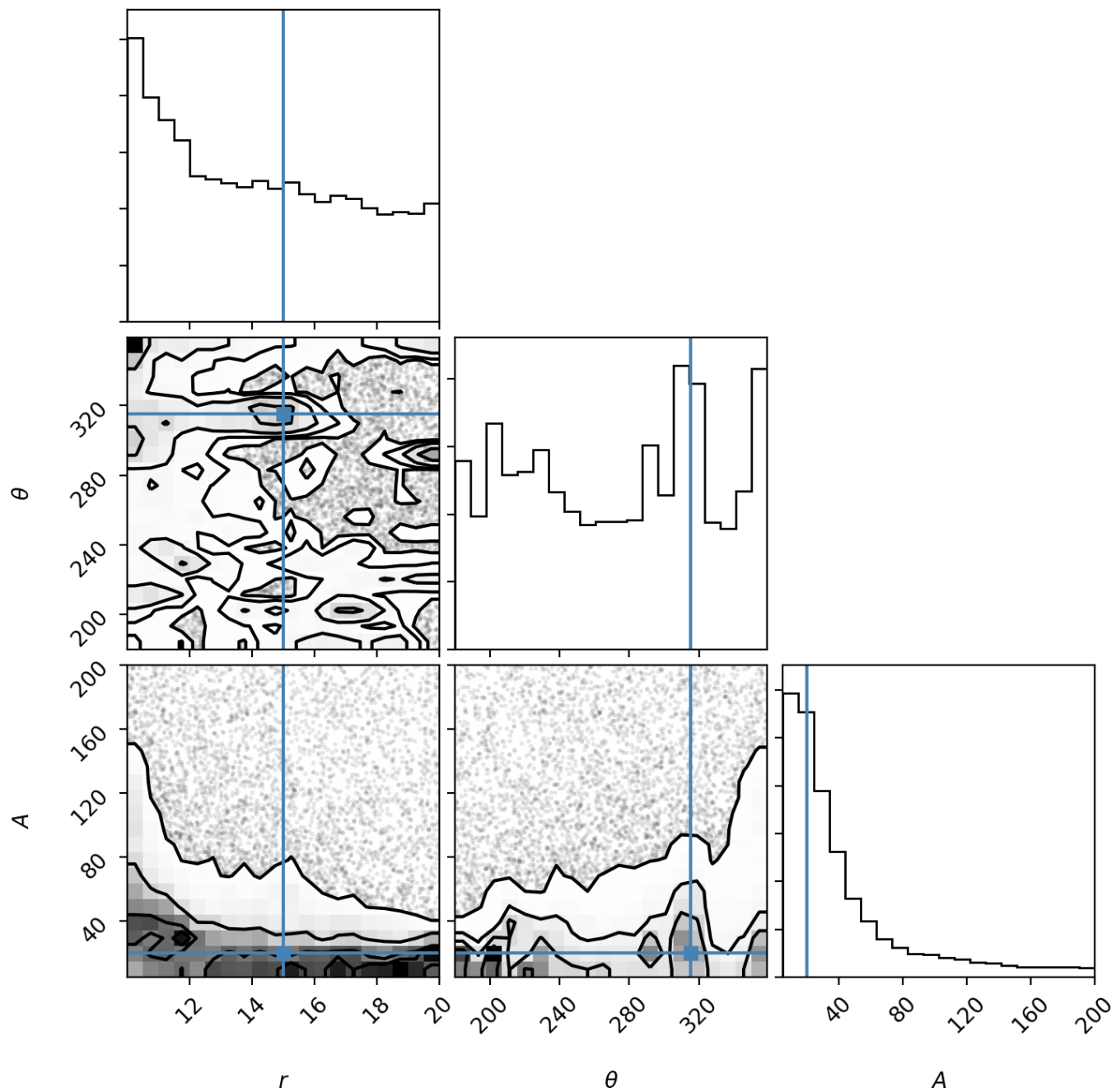
# 1) ADI with fake signal injected after

- Adding more components shrinks uncertainties
- Position and amplitude are perfectly recovered by probabilistic model
- **No self-subtraction even though the design vectors overlap with the PSF to some degree.**
- Remember:
  - **Raw data frames fed into algorithm;** not operating on reduced data
  - Model and systematics jointly fit

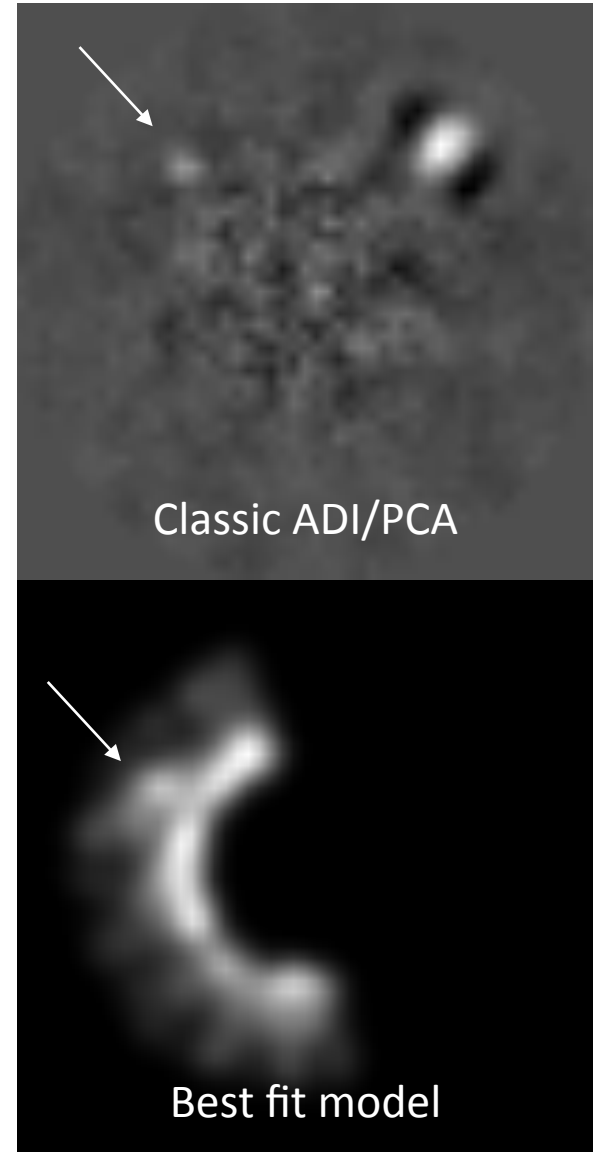
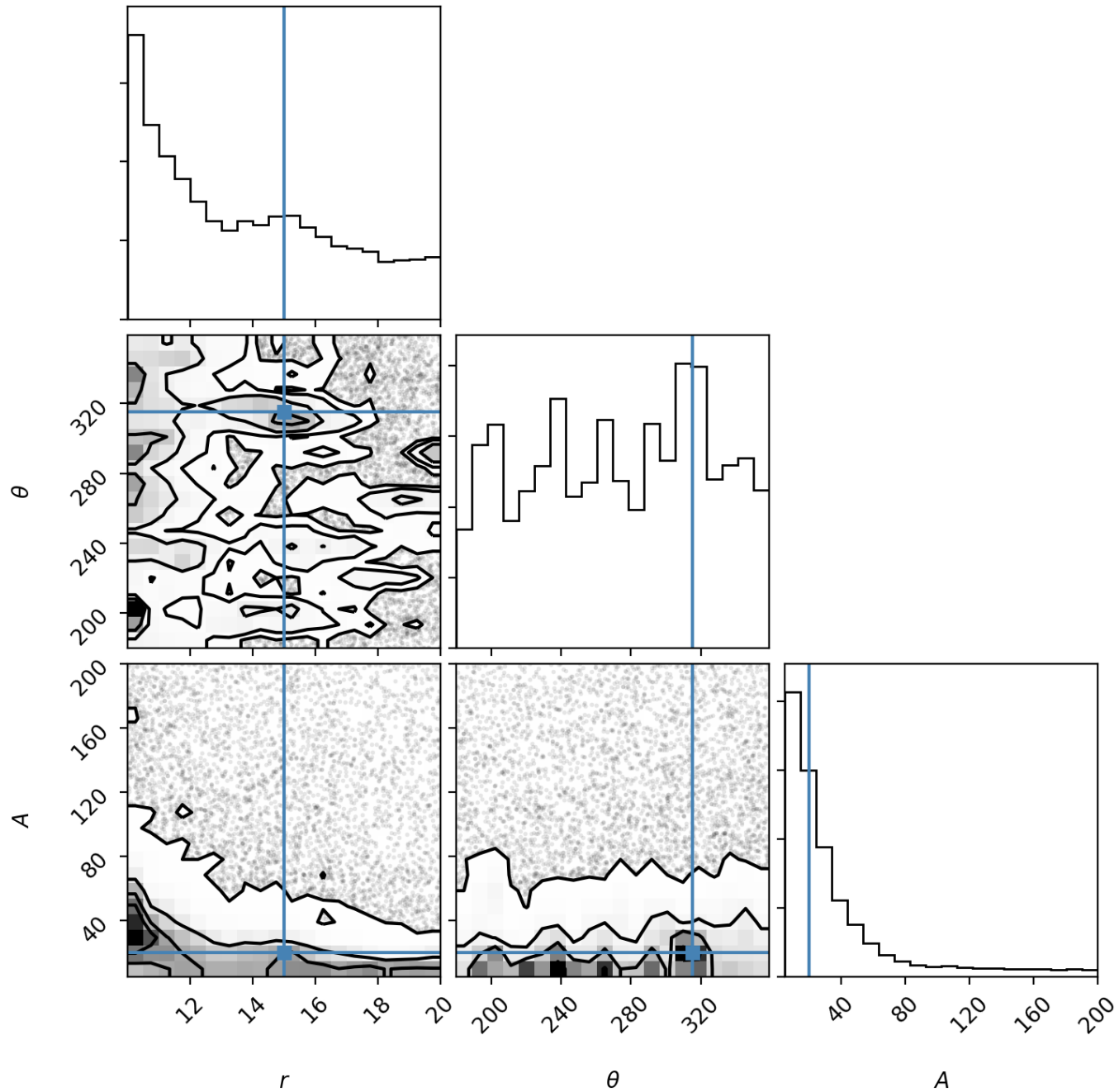
## 2) ADI with signal in the data

- “Realistic” ADI case
  - Calculate  $\mathbf{A}$ ,  $\mathbf{\Lambda}$ ,  $\mathbf{C}$  w/planet in data
  - Self-subtraction will be partially accounted for in error budget?
- Corresponds to what you get out of a telescope if you’re too lazy to do RDI

5 components

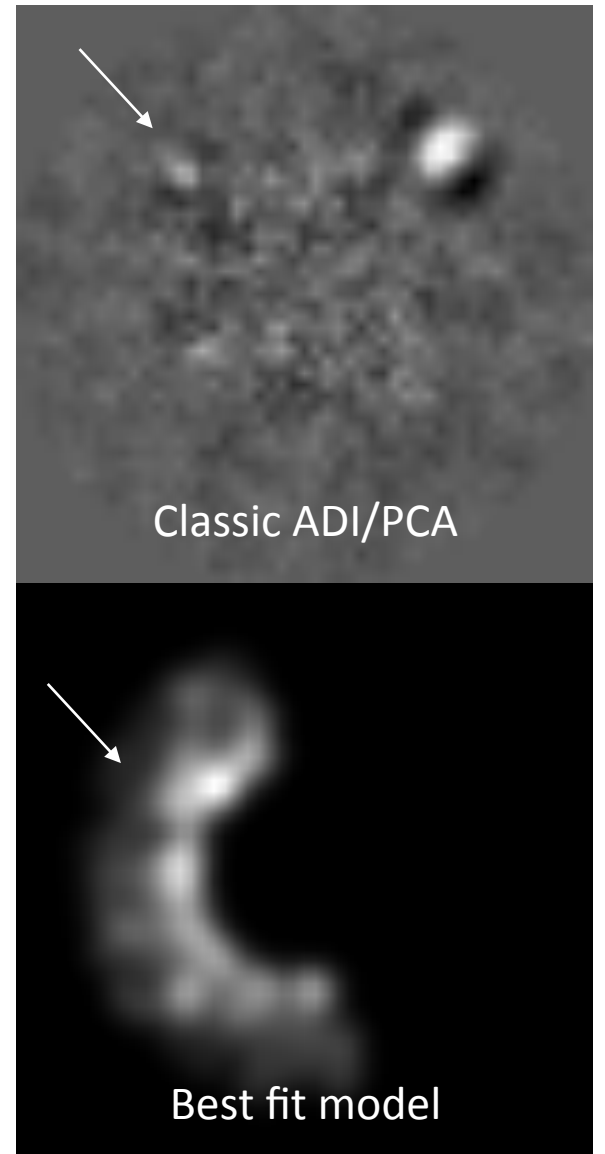
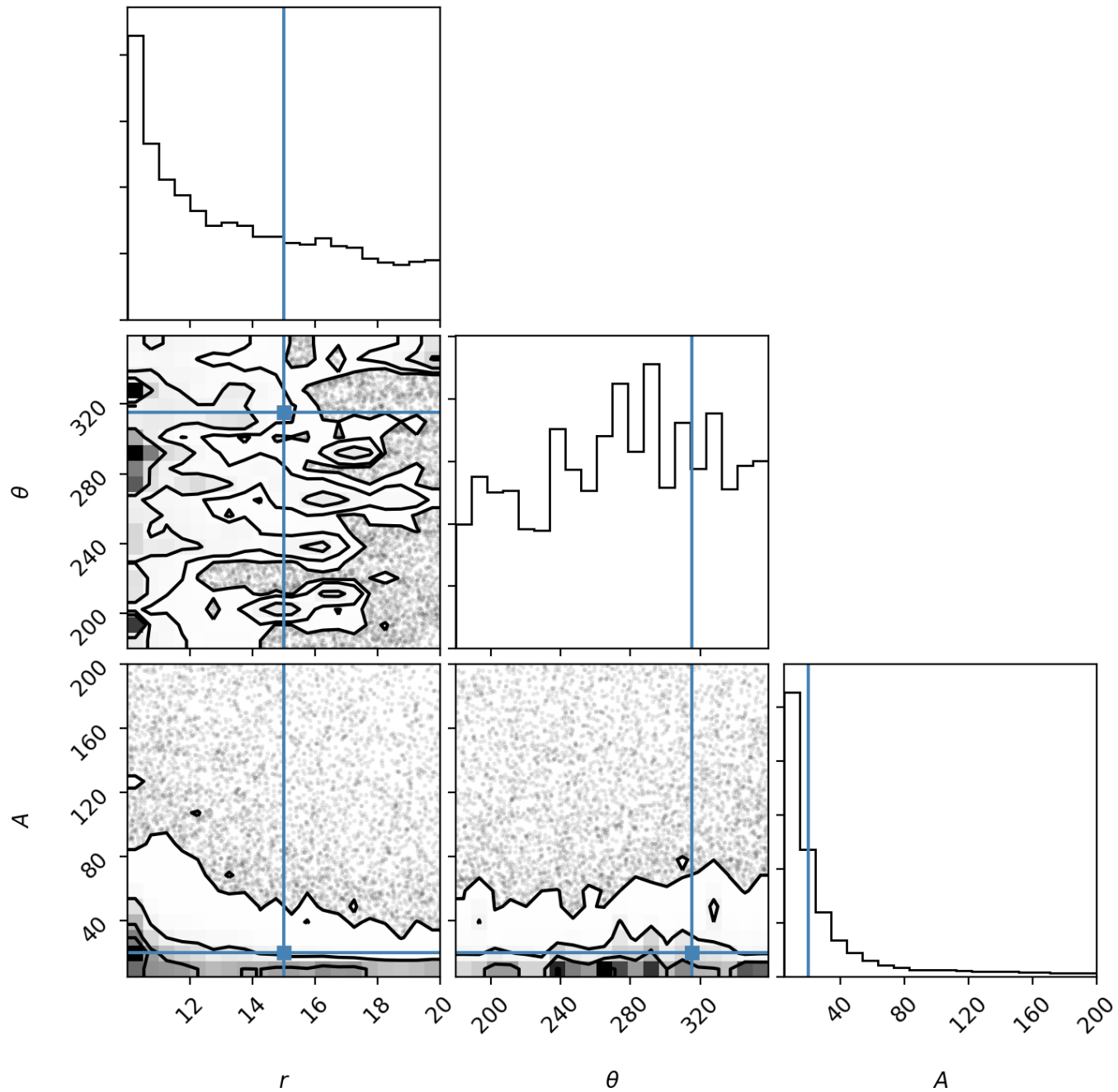


10 components

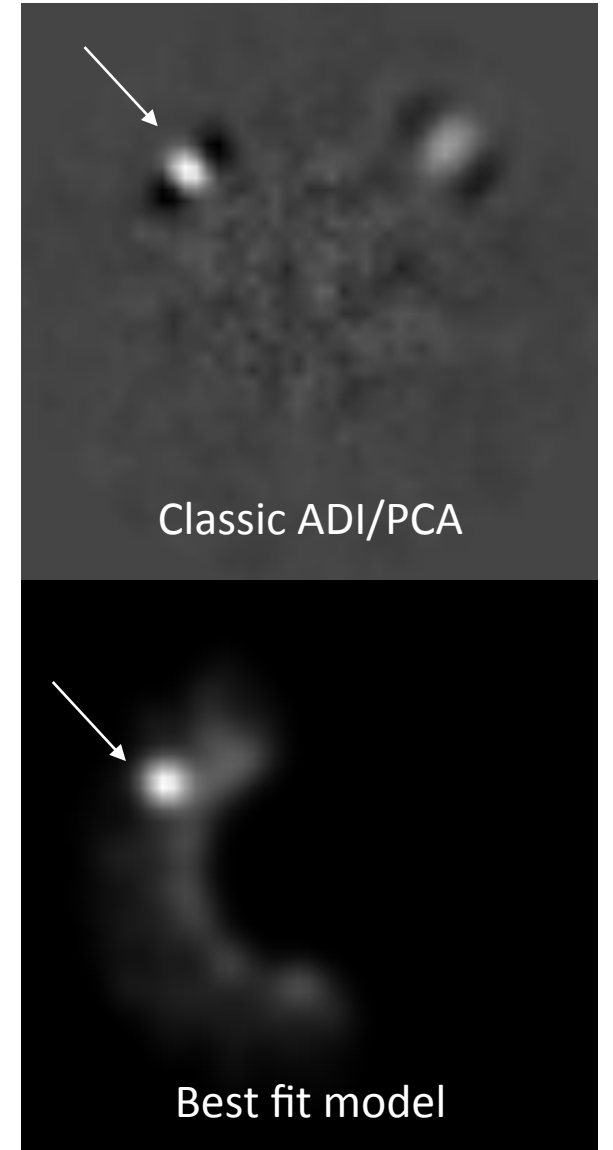
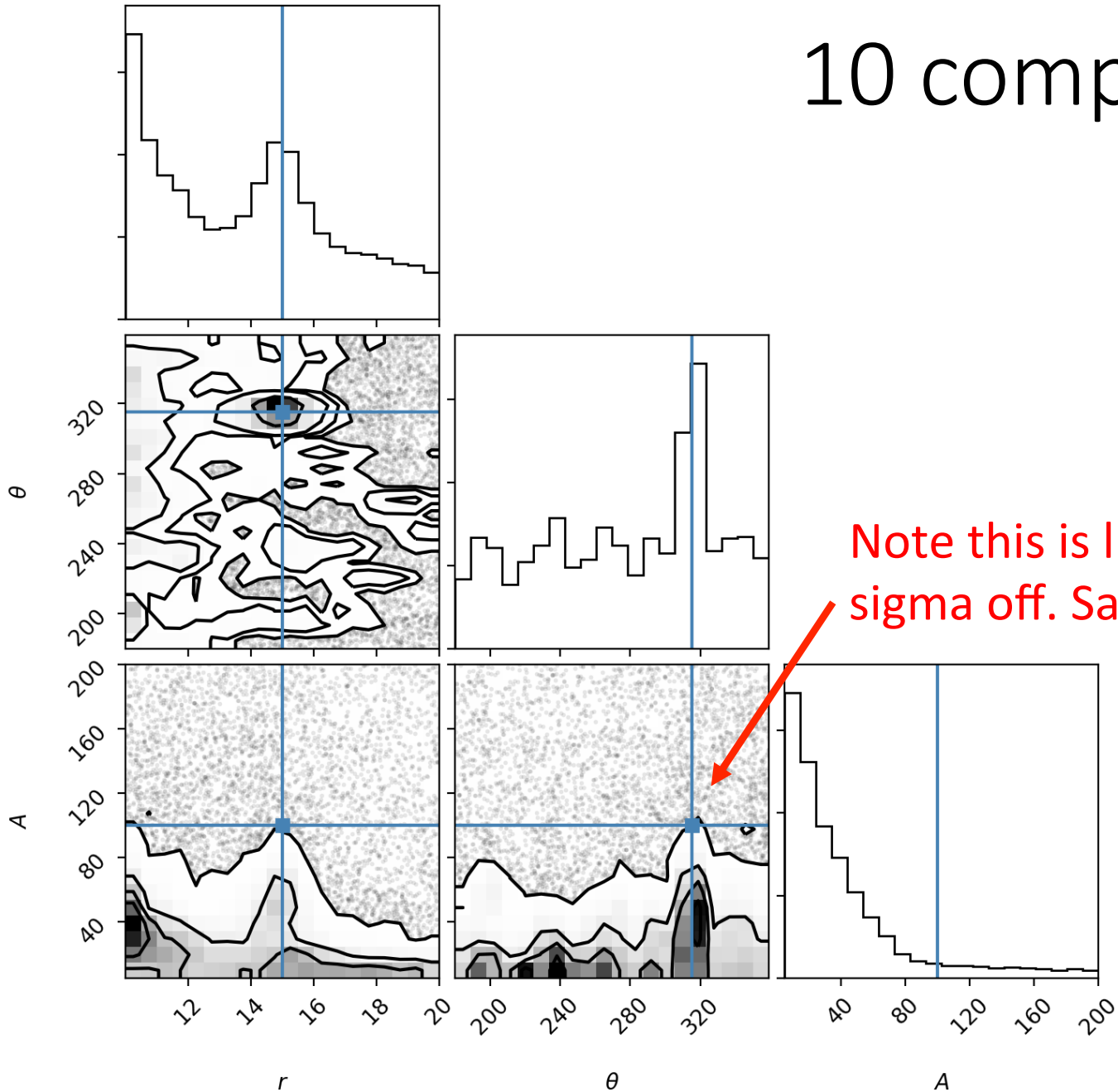




# 20 components



# 10 components, bright signal



# Analyzing why ADI isn't working for bright stuff

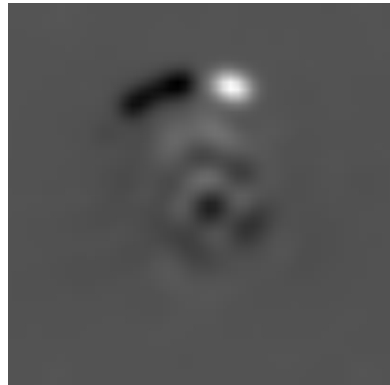
- The joint fit approach works fine on getting positions
- But the photometry is bad, bad, bad for brighter planets.
- The reason for this is because *the signal is included in the design matrix*

Difference between design  
matrix when signal is included  
vs not included in datacube

$\Delta A_1$



$\Delta A_2$



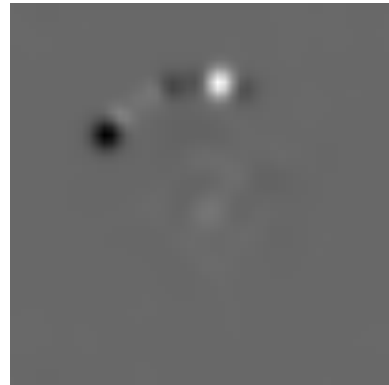
$\Delta A_3$



$\Delta A_4$



$\Delta A_5$



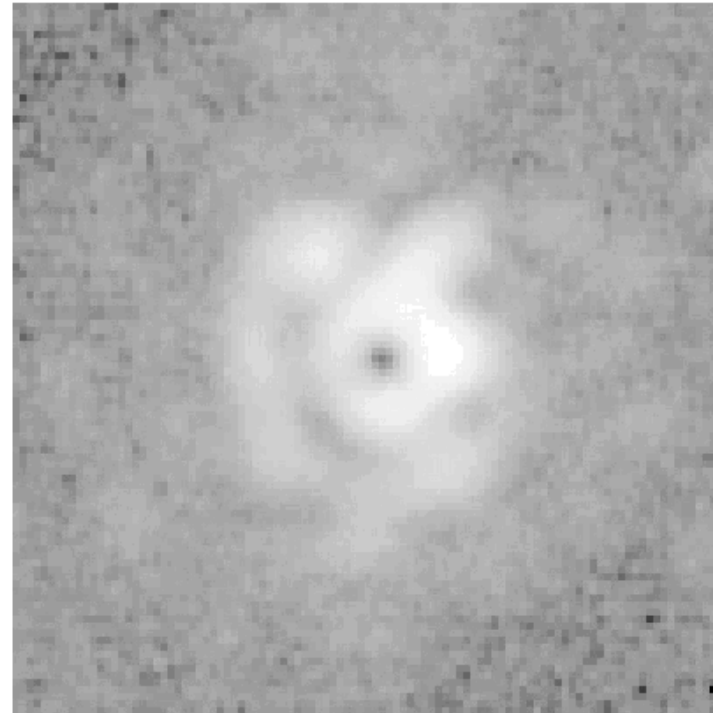
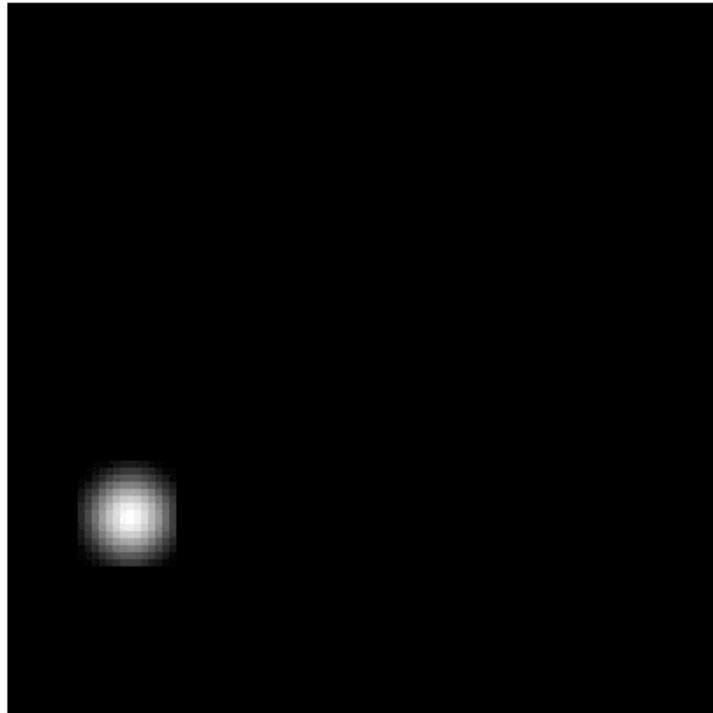
# How to fix this?

- Just do RDI
  - seems depressing to be forced into this
  - Already implemented
- Use temporal rather than spatial correlations
  - Interesting idea but concerned about computational time
- Nuclear option: Do a full inference on the design matrix as well as the model
  - This is the goal for the end of the year
  - Too much freedom?
  - Math hard
  - Computationally insane
- Help me?

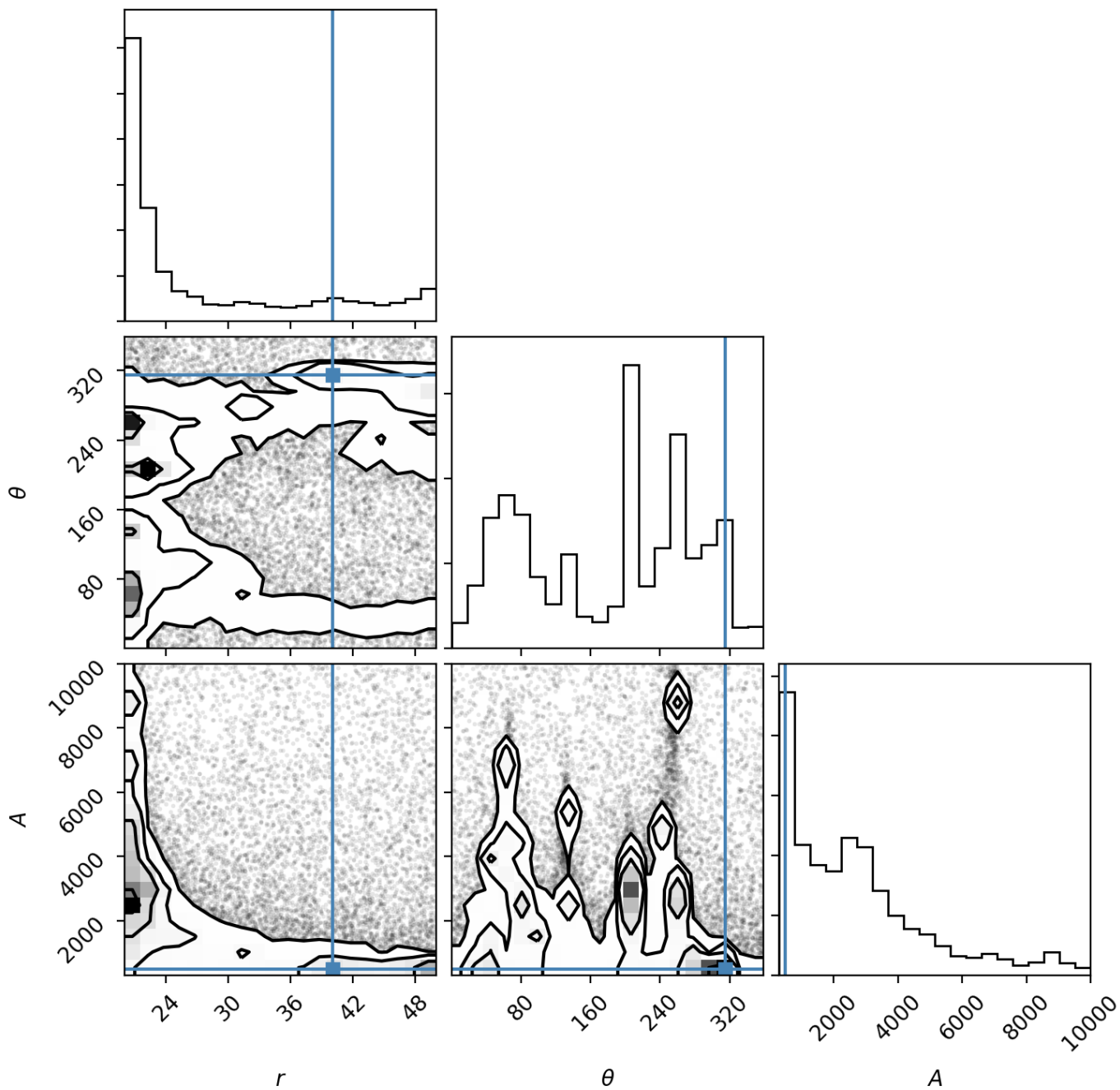


# RDI examples

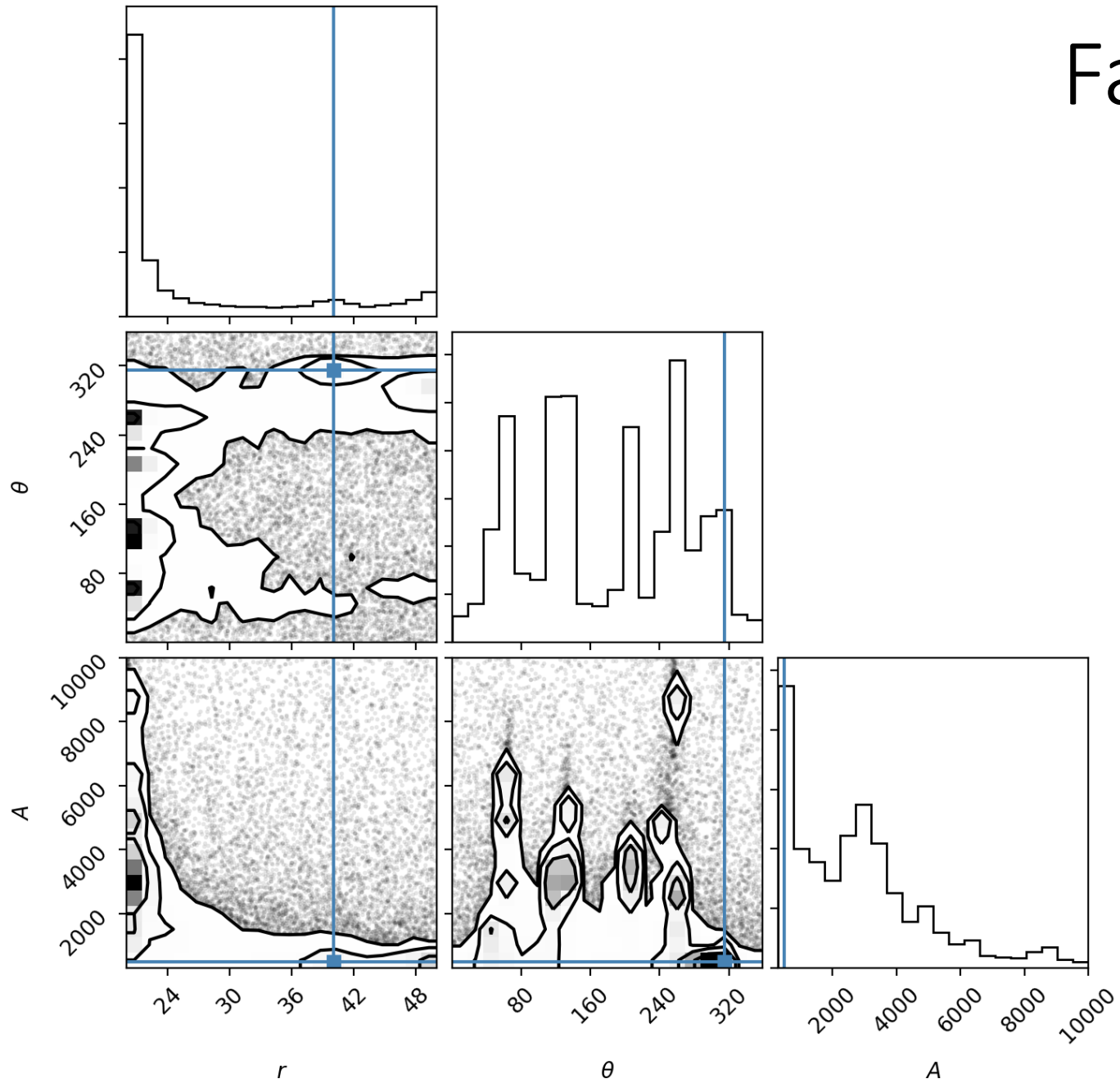
- Using Keck Vortex data
- No change to algorithm
- Still injecting fake signals tho



Faint signal, 5 comps

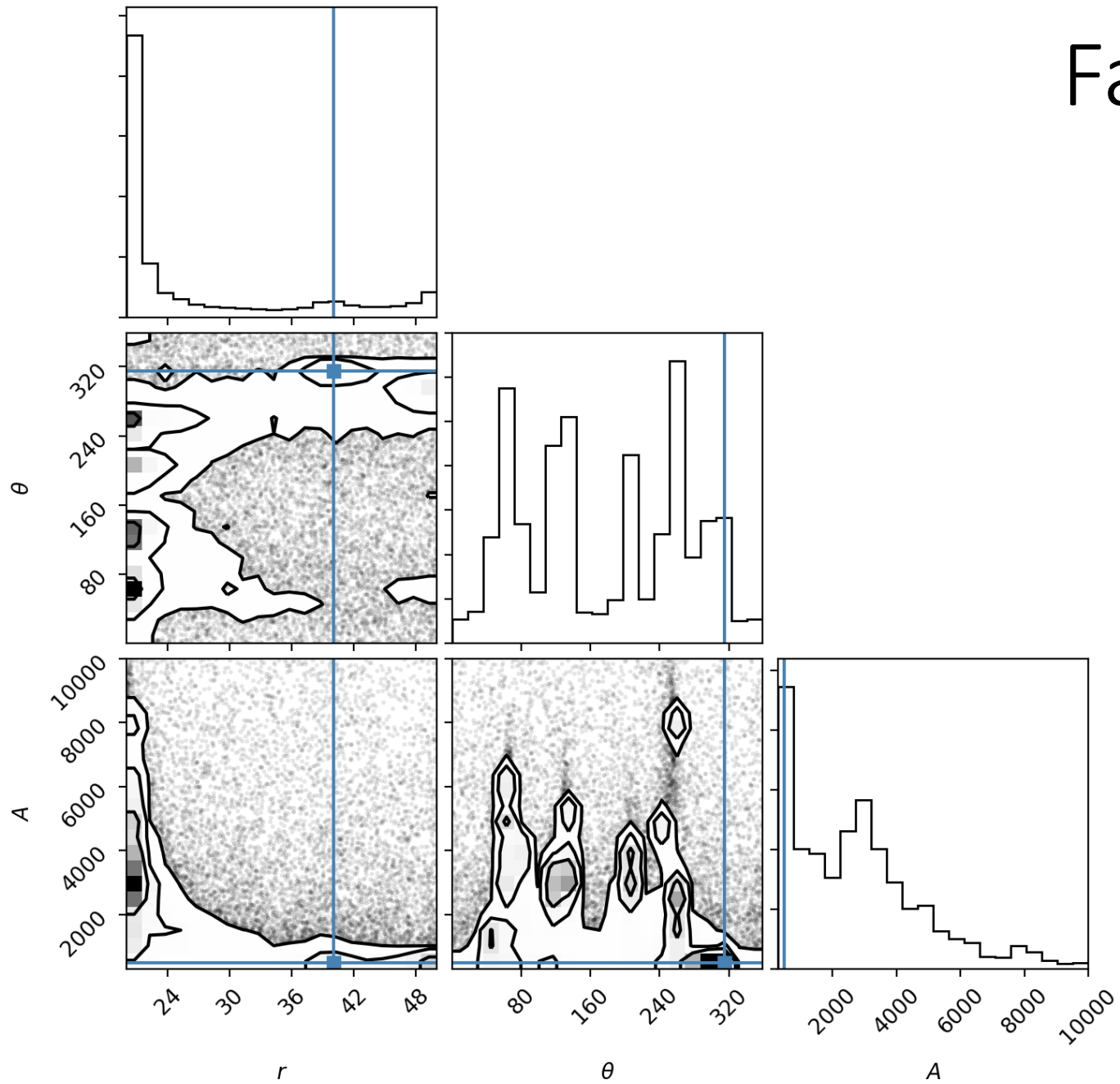


Faint signal, 10 comps

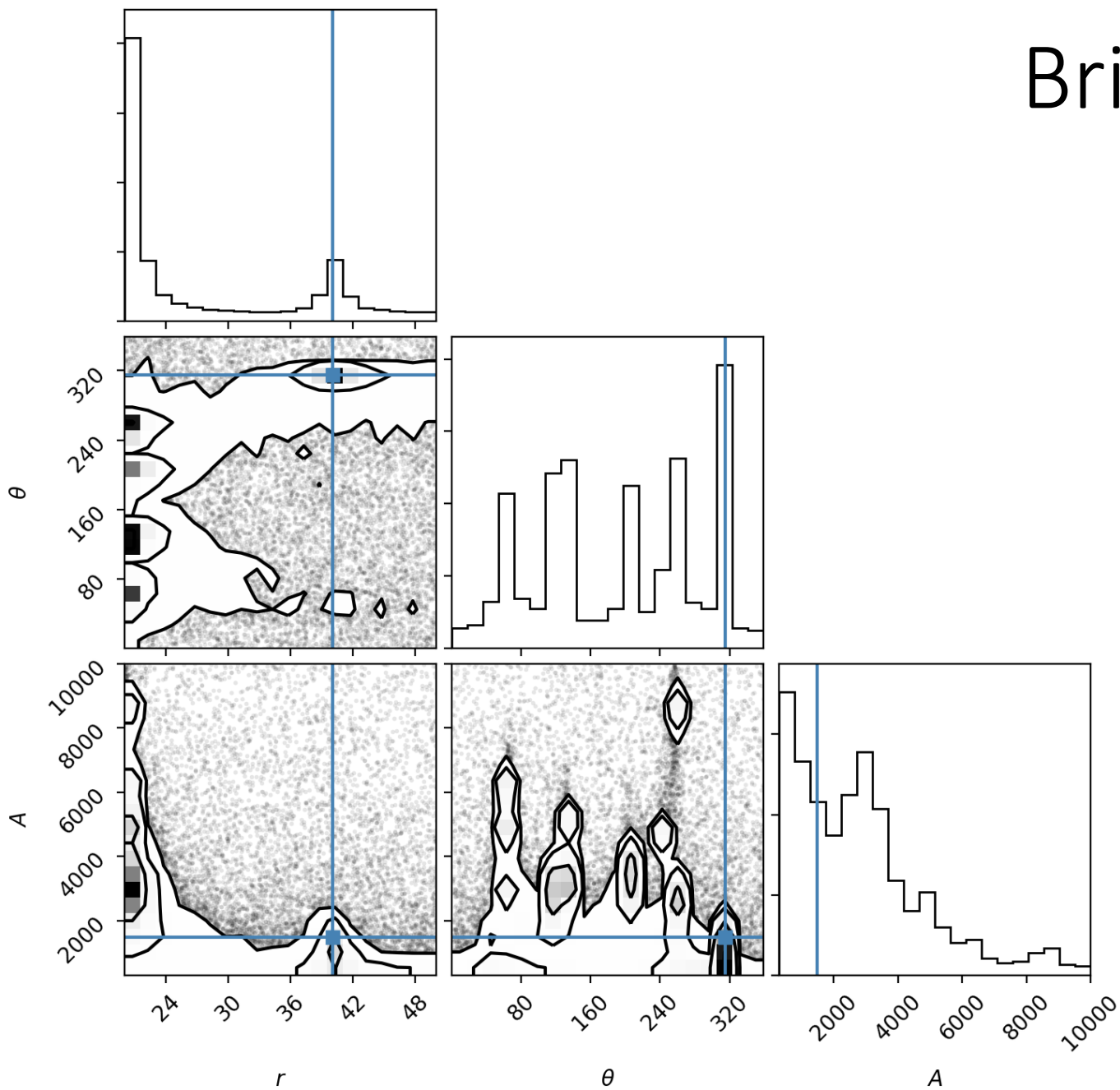




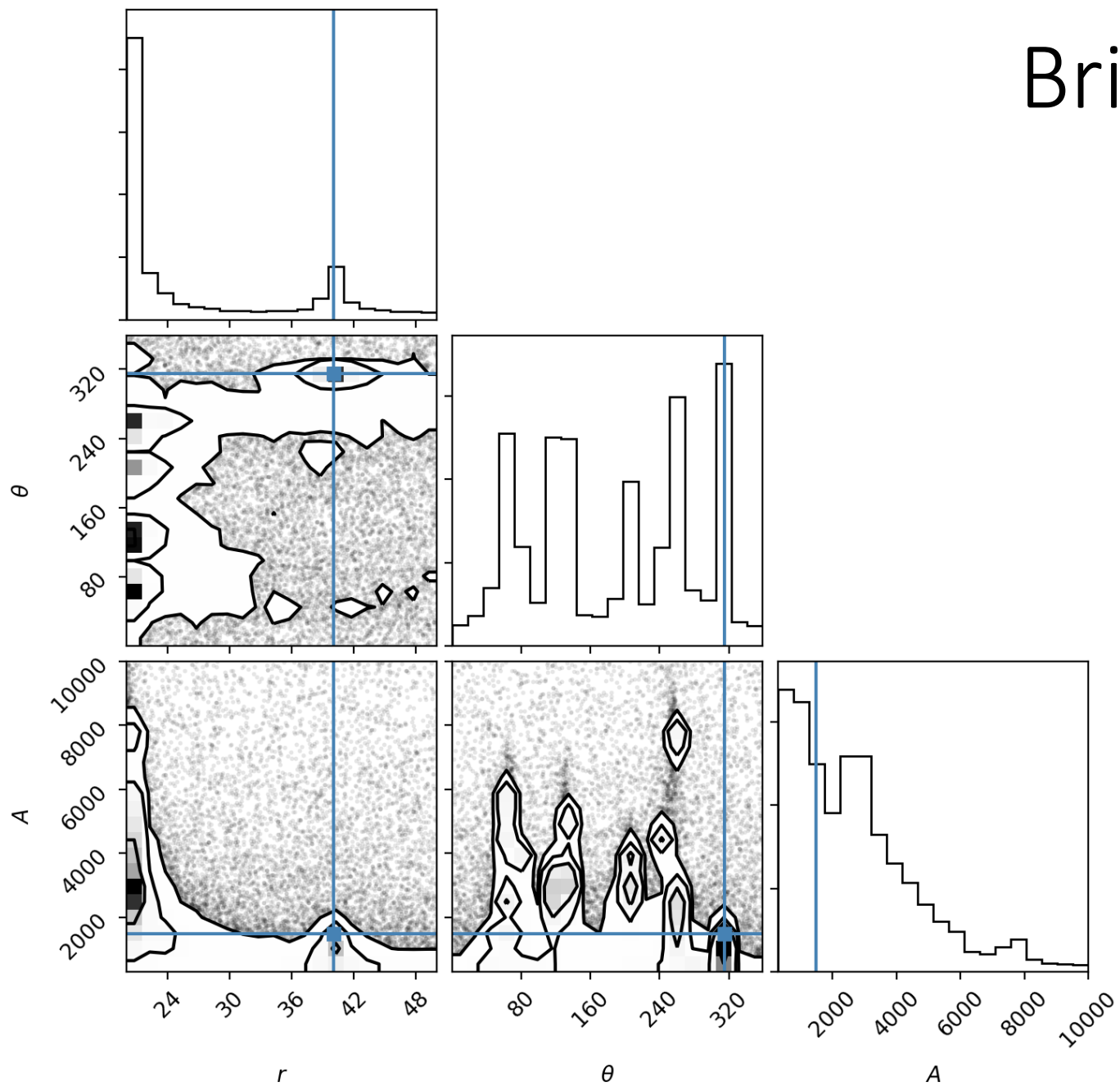
Faint signal, 15 comps



Bright signal, 10 comps



Bright signal, 15 comps



# Summary

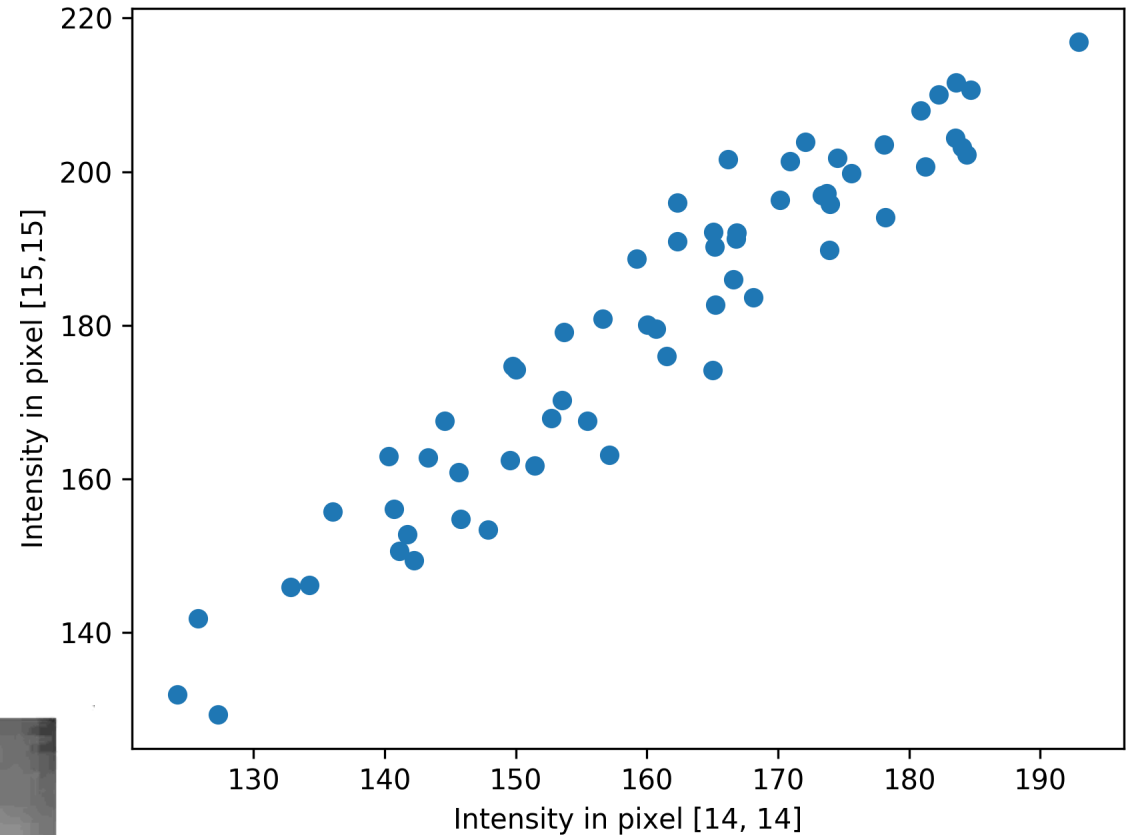
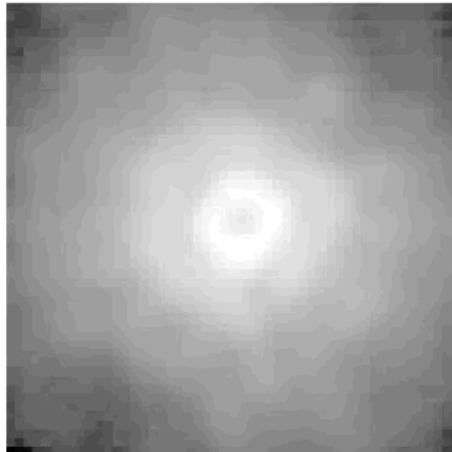
- Developing an approach to jointly model signal and systematics in high contrast imaging data
- Works on raw data frames
- Current work promising
  - No self-subtraction
  - Self-consistent, includes uncertainties in systematics
  - plays well with Bayes, MCMC
  - Allows inference

# Future work

- Try with more datasets
  - Disks
  - Multiple companions
  - Different telescopes
- Generalize to include datacube all at once rather than frame-by-frame
- Include planet noise
- Temporal correlation approach?
- Fully infer design matrix and signal

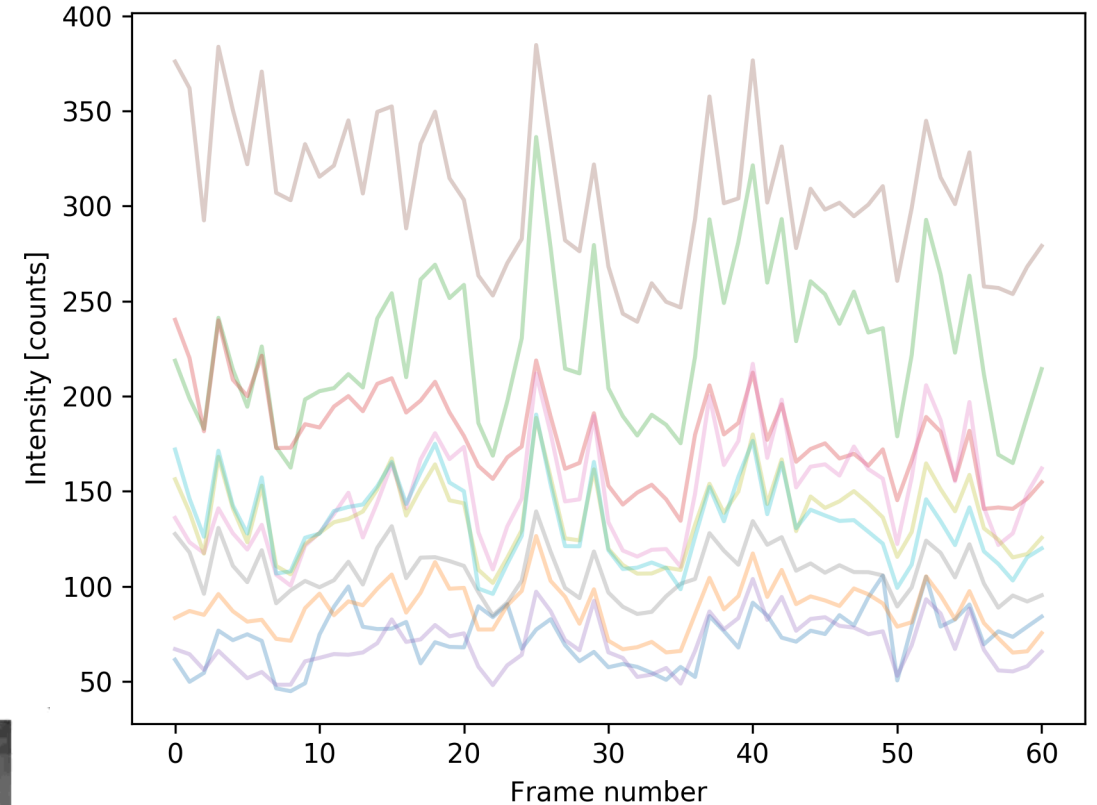
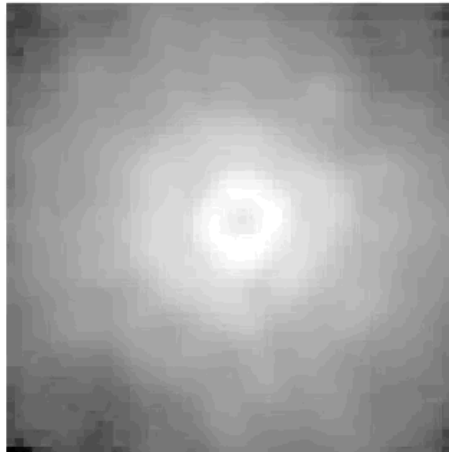
# Using temporal covariances instead

- PCA, LLSG, LOCI, etc are all based on spatial covariances
  - Diagonalizing covariance matrix
- But you can also think of the data cube as  $N_{\text{pix}}$  time series
- Some advantages
  - More datapoints than dimensions
  - Can generate basis vectors while *guaranteeing* signal is not in the vectors
  - Bright basis vectors can be used for faint points (less noise)



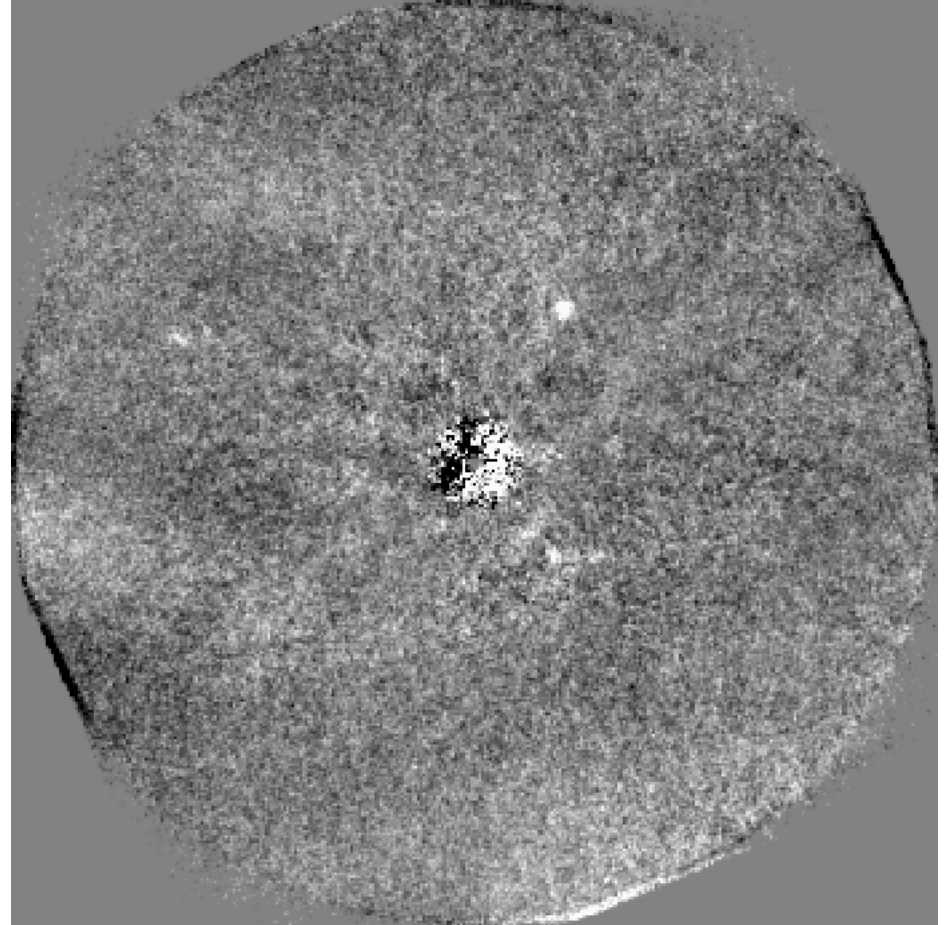
# Using temporal covariances instead

- PCA, LLSG, LOCI, etc are all based on spatial covariances
  - Diagonalizing covariance matrix
- But you can also think of the data cube as  $N_{\text{pix}}$  time series
- Some advantages
  - More datapoints than dimensions
  - Can generate basis vectors while *guaranteeing* signal is not in the vectors
  - Bright basis vectors can be used for faint points (less noise)



# Using temporal covariances instead

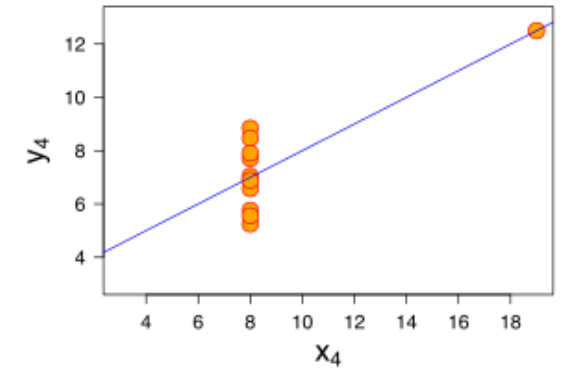
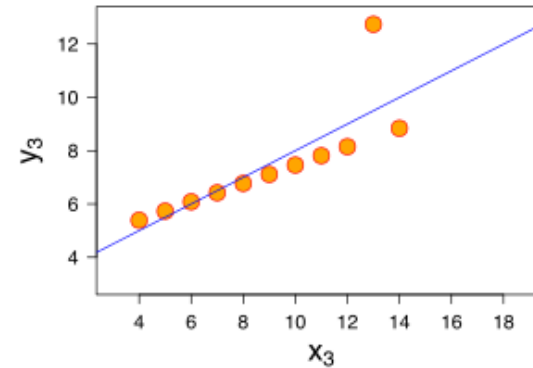
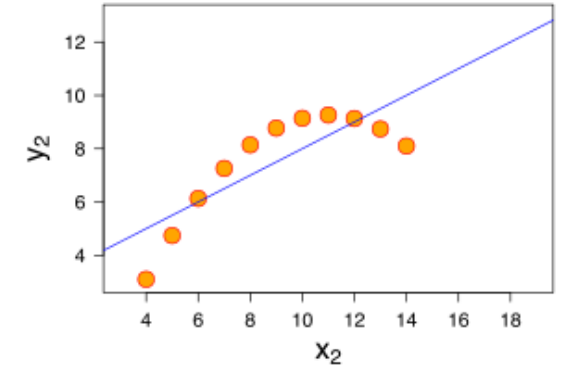
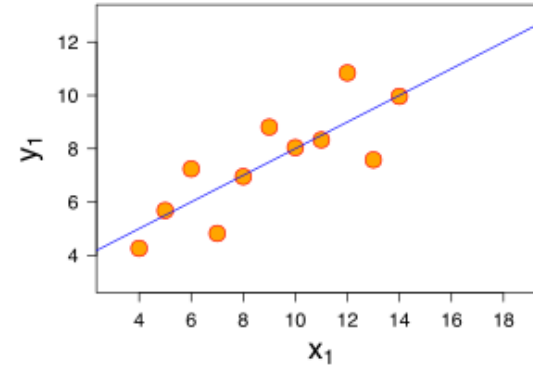
- Current work is semi-promising
- You can at least see planets but it's not that fast and I don't like how sensitive the algorithm is to different “regularization” parameters
- How to do cross-validation?
- Not clear how feasible with MCMC/Bayes





# Nested sampling

- Model fitting
  - multimodal parameter spaces
  - high-dimensional parameter spaces
  - Can run until convergence
- **Model selection**
  - Select between different models
  - ie, “is there a planet in my data?”



All these datasets have the same mean and stdev. All these fits are least square. But some of these models are pretty bad!!!

Bayesian inference provides a principled approach to the inference of a set of parameters,  $\Theta$ , in a model (or hypothesis),  $H$ , for data,  $\mathbf{D}$ . Bayes' theorem states that

$$\Pr(\Theta|\mathbf{D}, H) = \frac{\Pr(\mathbf{D}|\Theta, H) \Pr(\Theta|H)}{\Pr(\mathbf{D}|H)}, \quad (1)$$

where  $\Pr(\Theta|\mathbf{D}, H) \equiv P(\Theta|\mathbf{D})$  is the posterior probability density of the model parameters,  $\Pr(\mathbf{D}|\Theta, H) \equiv \mathcal{L}(\Theta)$  the likelihood of the data, and  $\Pr(\Theta|H) \equiv \pi(\Theta)$  the parameter prior. The final term,  $\Pr(\mathbf{D}|H) \equiv \mathcal{Z}$  (the Bayesian evidence), represents the factor required to normalize the posterior over the domain of  $\Theta$  given by:

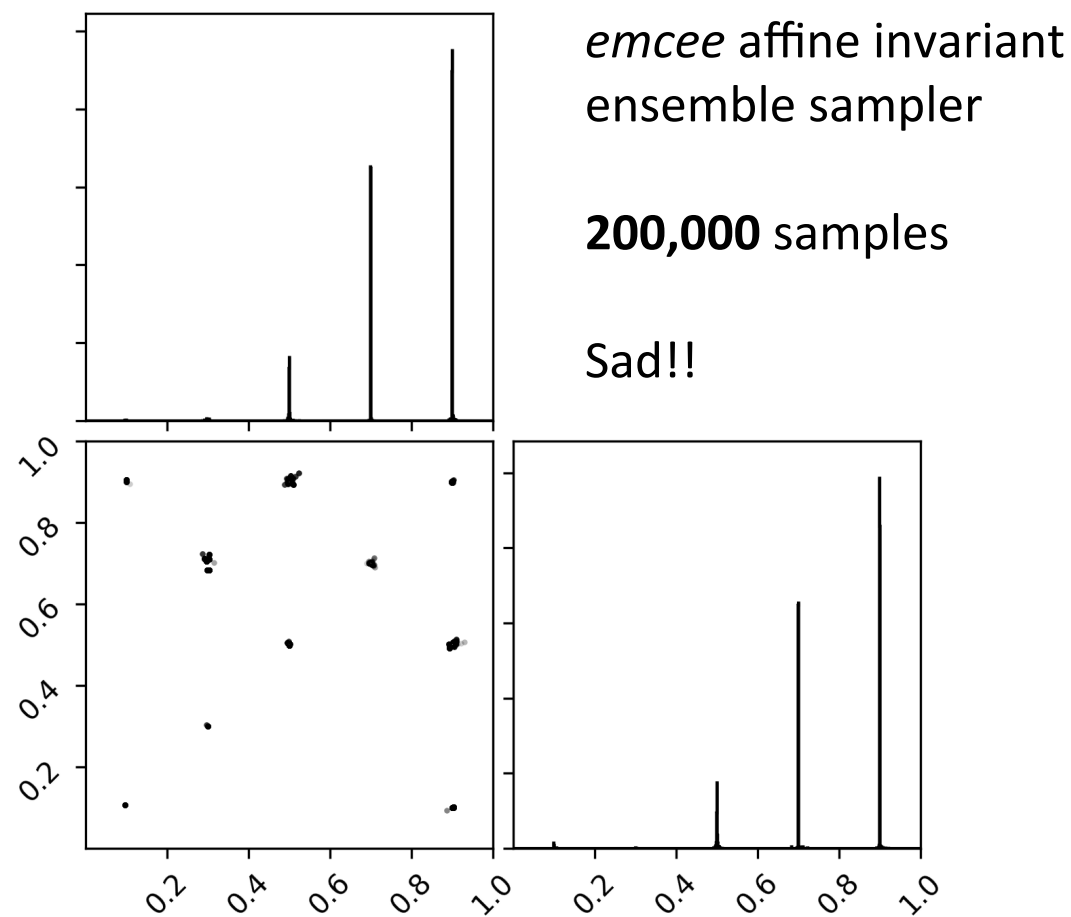
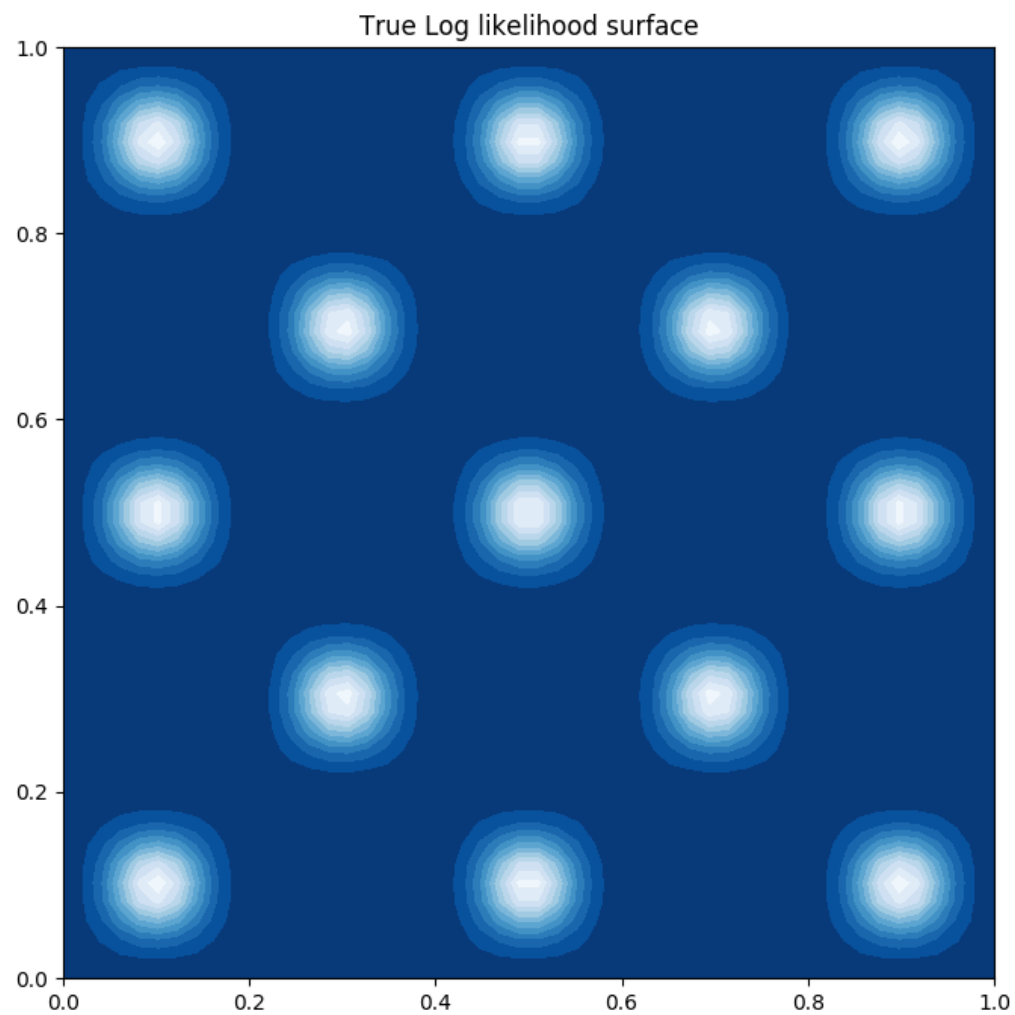
$$\mathcal{Z} = \int_{\Omega_{\Theta}} \mathcal{L}(\Theta) \pi(\Theta) d\Theta. \quad (2)$$

Being independent of the parameters, however, this factor can be ignored in parameter inference problems which can be approximated by taking samples from the unnormalized posterior only, using standard MCMC methods (for instance).

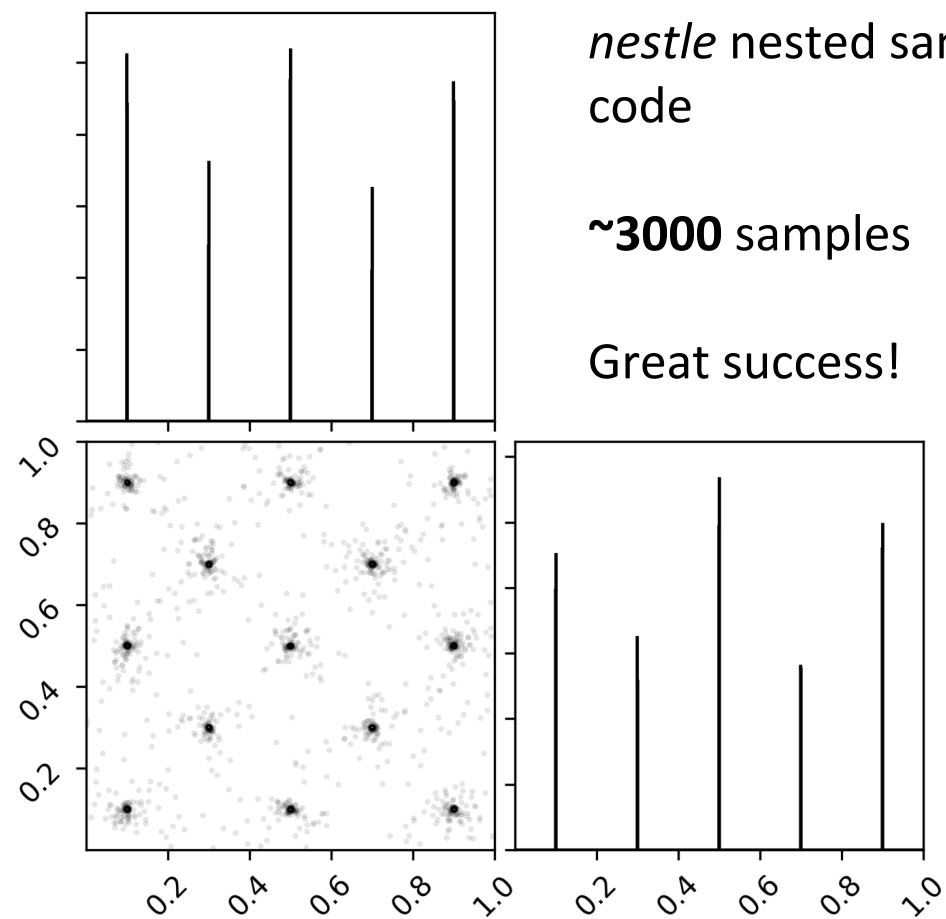
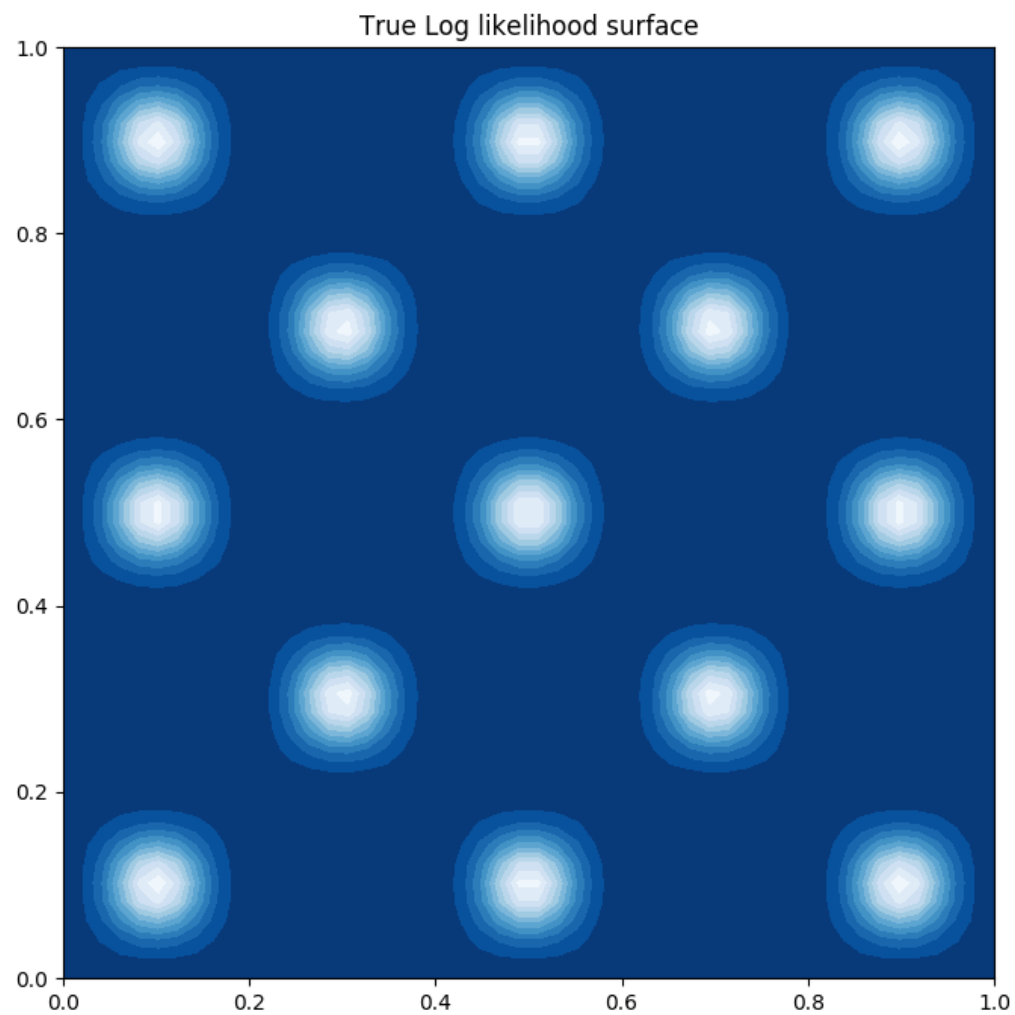
Model selection between two competing models,  $H_0$  and  $H_1$ , can be achieved by comparing their respective posterior probabilities given the observed dataset as follows:

$$R = \frac{\Pr(H_1|\mathbf{D})}{\Pr(H_0|\mathbf{D})} = \frac{\Pr(\mathbf{D}|H_1) \Pr(H_1)}{\Pr(\mathbf{D}|H_0) \Pr(H_0)} = \frac{\mathcal{Z}_1 \Pr(H_1)}{\mathcal{Z}_0 \Pr(H_0)}. \quad (3)$$

*Note:* Ensemble samplers don't work on high contrast imaging data



*Note:* Ensemble samplers don't work on high contrast imaging data



*Note:* Ensemble samplers don't work on high contrast imaging data

## DNest4: Diffusive Nested Sampling in C++ and Python

Brendon J. Brewer

Department of Statistics  
The University of Auckland

Daniel Foreman-Mackey

Sagan Fellow  
University of Washington

Package	Easy to implement models?	High dimensions?	Multimodal distributions?	Dependent distributions?	Phase changes?	Computes $Z$ ?
<b>DNest4</b>	✗	✓	✓	✓	✓	✓
<b>emcee</b>	✓	✗	✗	✓	✗	✗
<b>JAGS</b>	✓	✓	✗	✗	✗	✗
<b>MultiNest</b>	✓	✗	✓	✓	✓	✓
<b>Stan</b>	✓	✓	✗	✓	✗	✗

Table 1: A simplified summary of the advantages and disadvantages of some Bayesian computation software packages.