Curve Fitting

JERRY JEYACHANDRA

BLOHM LAB

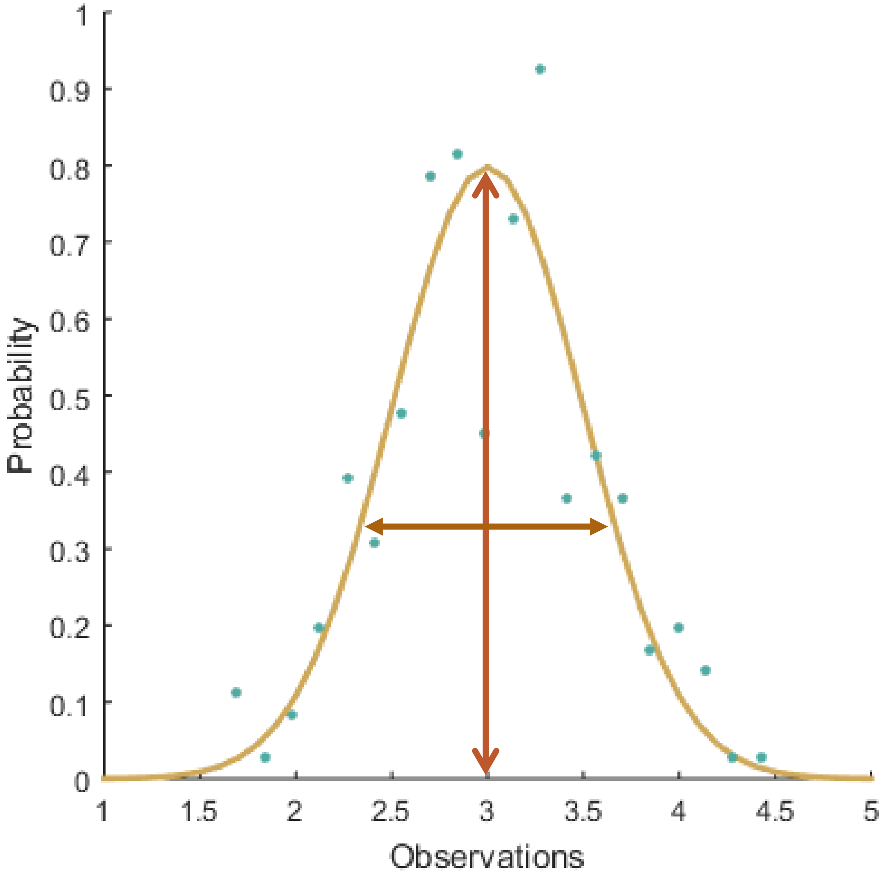
What is Curve-Fitting?

Fitting empirical data with a mathematical function.

Simple: Best-fit line

Complex: Multi-stage model

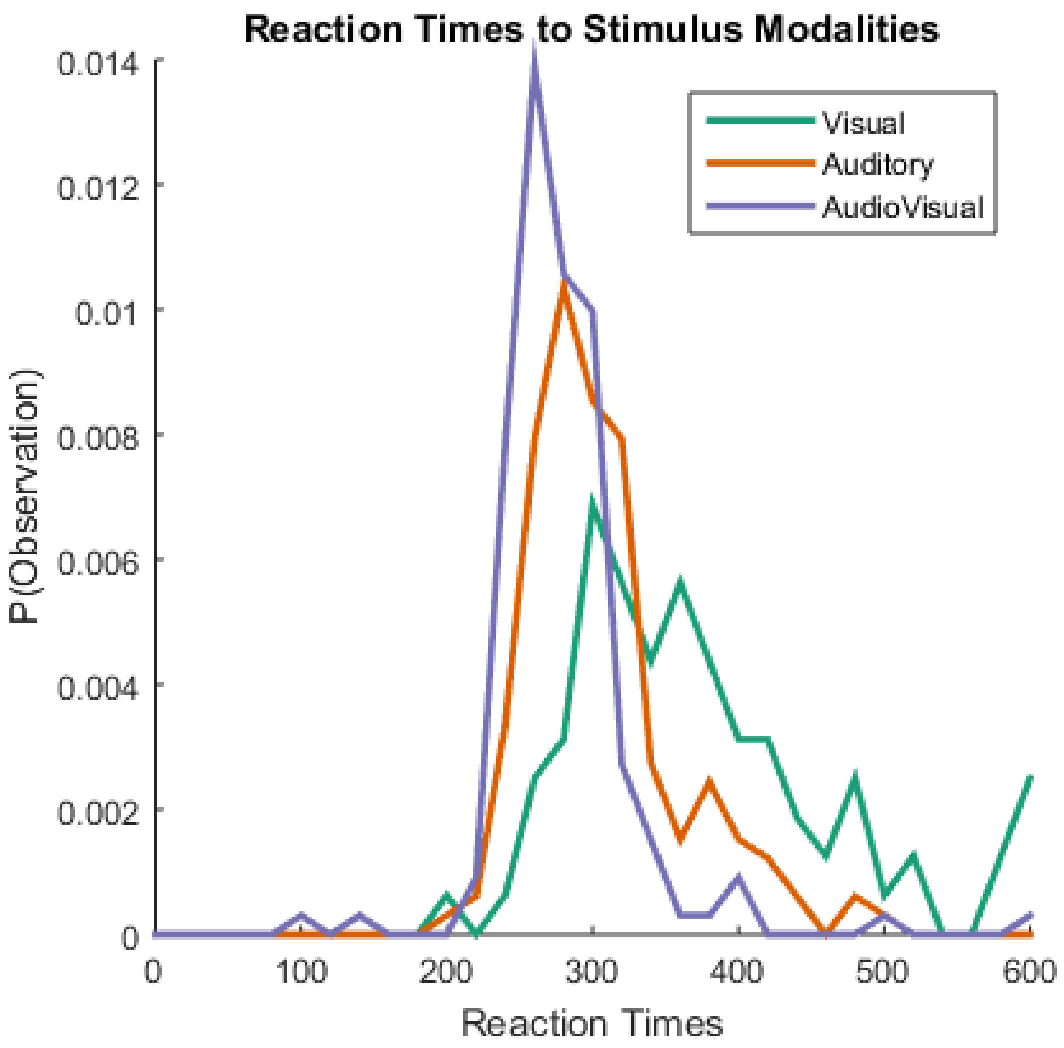
The underlying process?



Why use Curve Fitting?

1. Succinctly and quantitatively describe the relationships within the data
2. Making predictions outside your dataset
3. Estimate meaningful parameters for your data
4. Testing model predictions

Example: Reaction Times



http://www.neural-code.com/index.php/tutorials/action/saccade/58-saccade-introduction

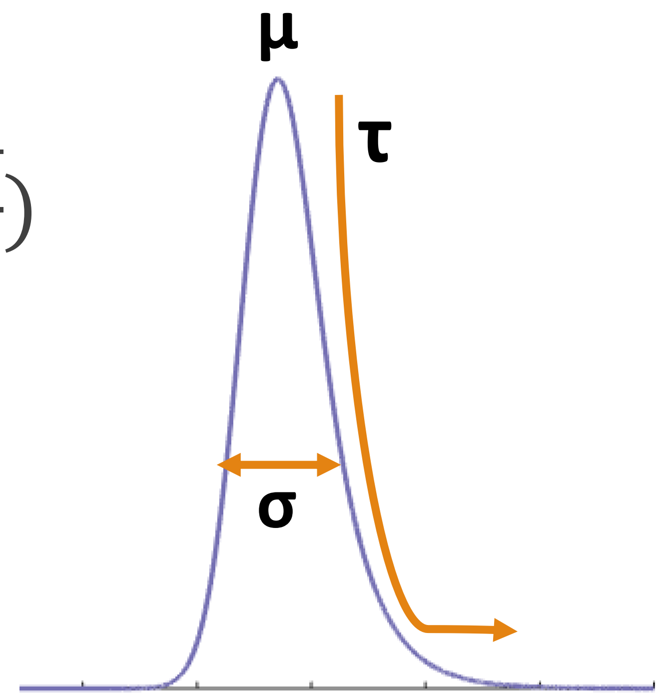
Model 1: Ex-Gaussian Fit

Model: Ex-Gaussian

$$f(x|\mu,\sigma,\tau) = \frac{1}{\tau}\exp\left(\frac{\mu}{\tau} + \frac{\sigma^2}{2\tau^2} - \frac{x}{\tau}\right)\varphi\left(\frac{x-\mu-\frac{\sigma^2}{\tau}}{\sigma}\right)$$

Goal: Find the parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\tau}$ that best quantifies the data.

How?



Method 1: Maximum Likelihood Estimation

Use the *likelihood value* in order to determine the most likely parameters to the data

Given a density function:

$f(x|\theta)$ where θ defined as our fitting parameters.

The likelihood function is defined as:

$$L(\theta|X) = \prod_{i=1}^{N} f(x_i|\theta)$$

Method 1: Maximum Likelihood Estimation

Use log-likelihood to prevent floating errors

$LogL(\theta|X) = \sum_{i=1}^{N} \ln[f(x_i|\theta)]$ ← Maximize!

Optimization problem: Use an iterative algorithm.

Method 1: Maximum Likelihood Estimation

Use log-likelihood to prevent floating errors

$LogL(\theta|X) = -\sum_{i=1}^{N} \ln[f(x_i|\theta)]$ ← Minimize!

Optimization problem: Use an iterative algorithm.

MATLAB Implementation: fminsearch, fminsearchbd, fmincg, gradient methods

Method 1: Maximum Likelihood Estimation – **MATLAB Implementation**

1. Setting up the Model function

```
function [ F ] = ExGauss( data, params )

%Set parameter associations
mu = params(1);
sigma = params(2);
tau = params(3);

%Model (Ex-Gaussian)
phi = normcdf((data-mu-sigma.^2./tau)./sigma)./tau;
F = exp(-data./tau + mu./tau + sigma.^2./2./tau.^2).*phi;

%Safety for 0 divisions
F(F == Inf)=zeros(length(F(F==Inf)),1);
if (tau < 0 || sigma < 0)
    F = zeros(length(F),1);
end


end
```

$$f(x|\mu,\sigma,\tau) = \frac{1}{\tau}\exp(\frac{\mu}{\tau} + \frac{\sigma^2}{2\tau^2} - \frac{x}{\tau})\varphi(\frac{x-\mu-\frac{\sigma^2}{\tau}}{\sigma})$$

Method 1: Maximum Likelihood Estimation – **MATLAB Implementation**

2. Setting up the Maximum Likelihood Function

```
function [ LogP ] = MaxLkh(params,data)

%Compute Ex-Gauss values for data
p = ExGauss(data,params);

%Compute log-likelihood for given data
LogP = -sum(log(p));

end
```

$$LogL(\theta|X) = -\sum_{i=1}^{N} \ln[f(x_i|\theta)]$$

Method 1: Maximum Likelihood Estimation – **MATLAB Implementation**

3. Implementing optimization algorithm - **fminsearch**

```
%Set initial parameter values
tauV = std(visual).*0.8;
muV = mean(visual) - skewness(visual);
sigmaV = sqrt(var(visual)-(tauV^2));

%Perform search algorithm
visParams = fminsearch(@MaxLkh,[muV sigmaV tauV],[],visual);
```

Max # Iterations

Function Pointer
Function to minimize

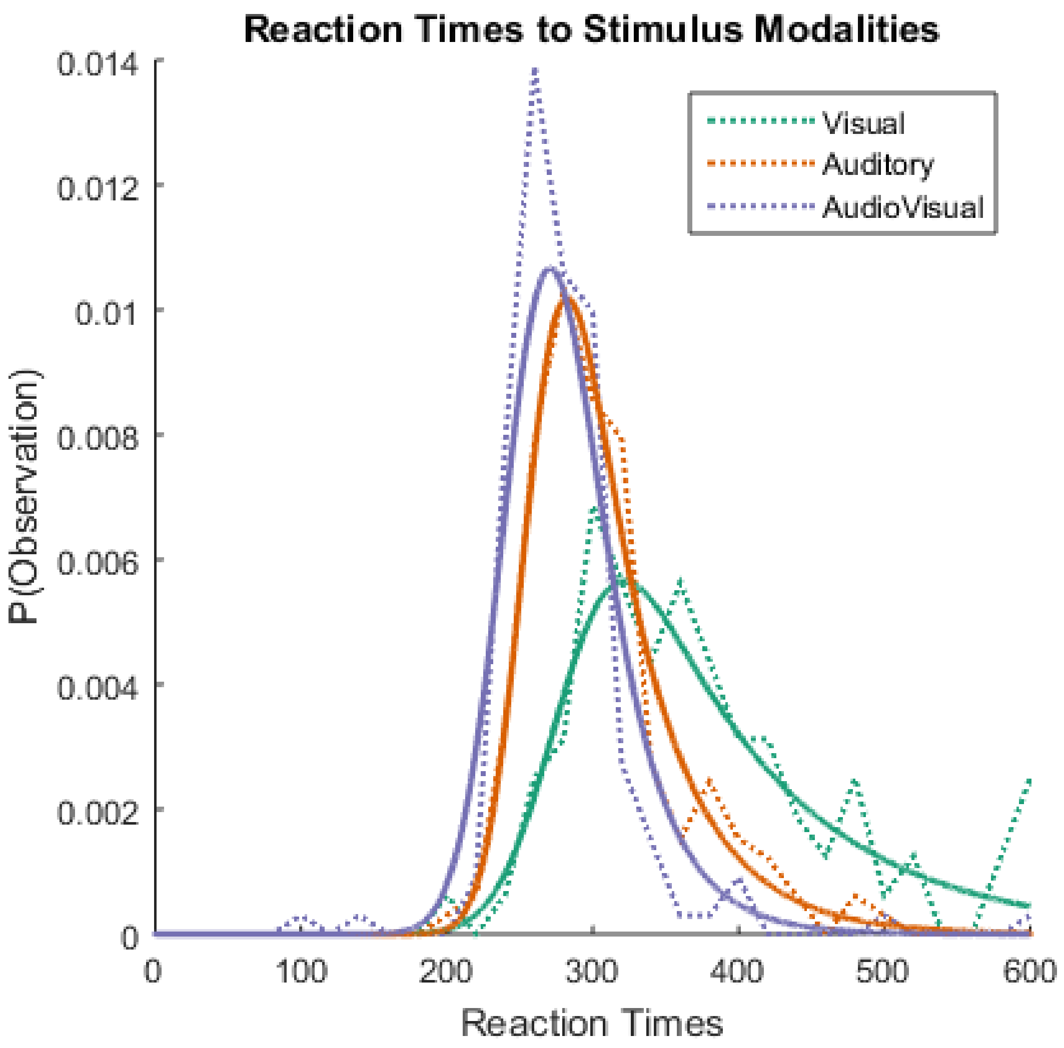
Initial Values
First input to MaxLkh

Data
Second input to MaxLkh

Method 1: Maximum Likelihood Estimation – **MATLAB Implementation**

4. Visualizing the Fit

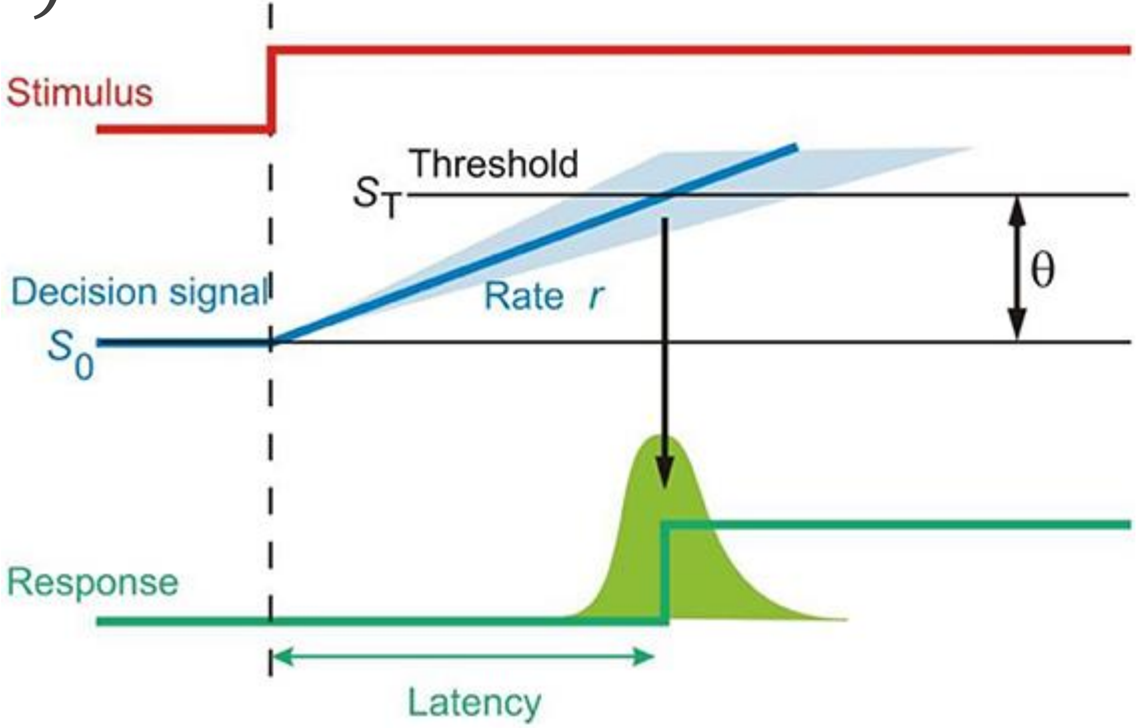
Remember: output of fminsearch is the parameters we care about!



Model 2: Linear Approach to Threshold with Ergodic Rate (LATER)

Model: LATER

$T = \frac{S_T - S_0}{r}$ where $r \sim \mathcal{N}(\mu, \sigma^2)$



Model 2: Linear Approach to Threshold with Ergodic Rate (LATER)

Model: LATER

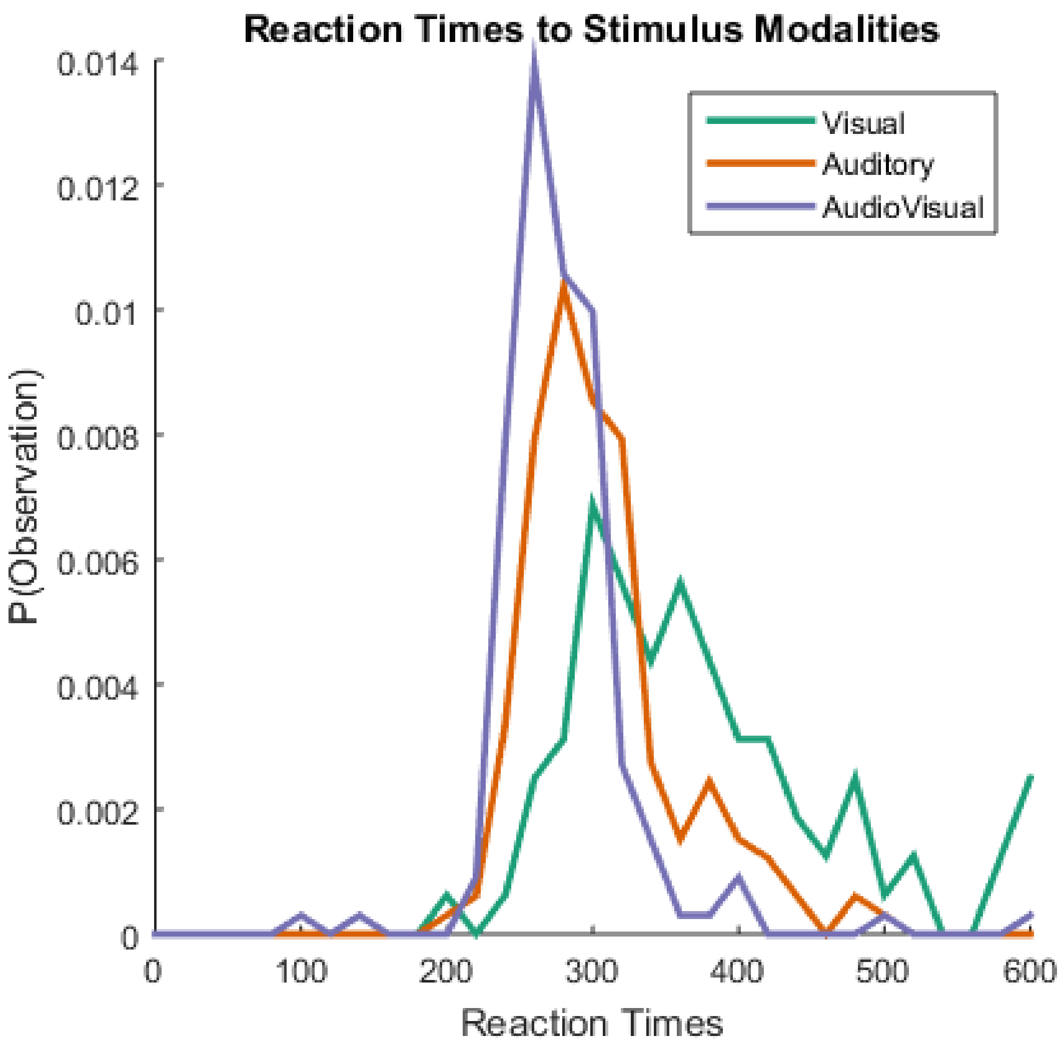
$T = \frac{S_T - S_0}{r}$ where $r \sim \mathcal{N}(\mu, \sigma^2)$

Goal: Find the parameters **ΔS**, **μ**, **σ** that best quantifies the data

- Recinormal fitting → "Reciprocal Normal"
  $\frac{1}{T} = \frac{r}{\Delta S}$ where $r \sim \mathcal{N}(\mu, \sigma^2)$

  Assumptions!

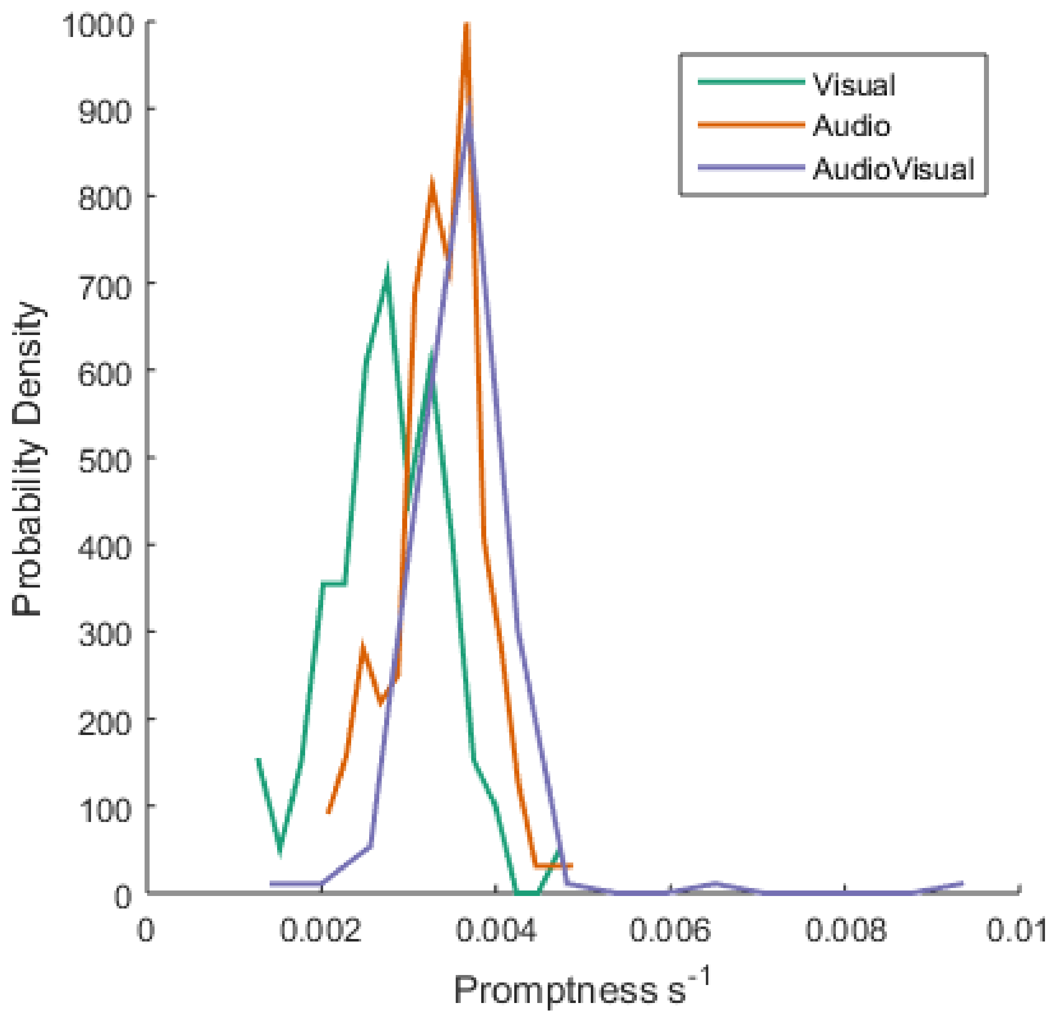
Model 2: LATER Working with the Data



http://www.neural-code.com/index.php/tutorials/action/saccade/58-saccade-introduction

Model 2: LATER Working with the Data

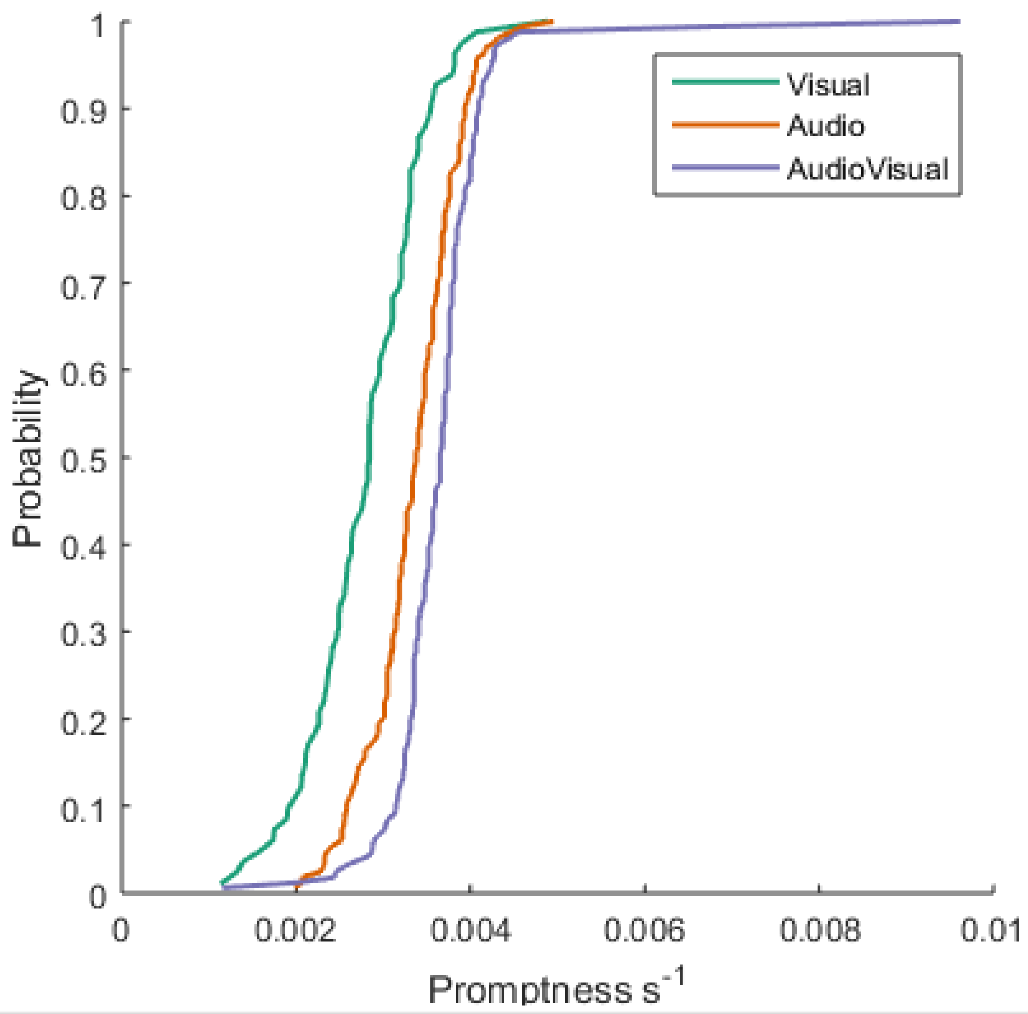
Reciprocal Transform




```
%Get reciprocal of data
visual = 1./visual;
audio = 1./audio;
audVic = 1./audVis;
```

Model 2: LATER Working with the Data

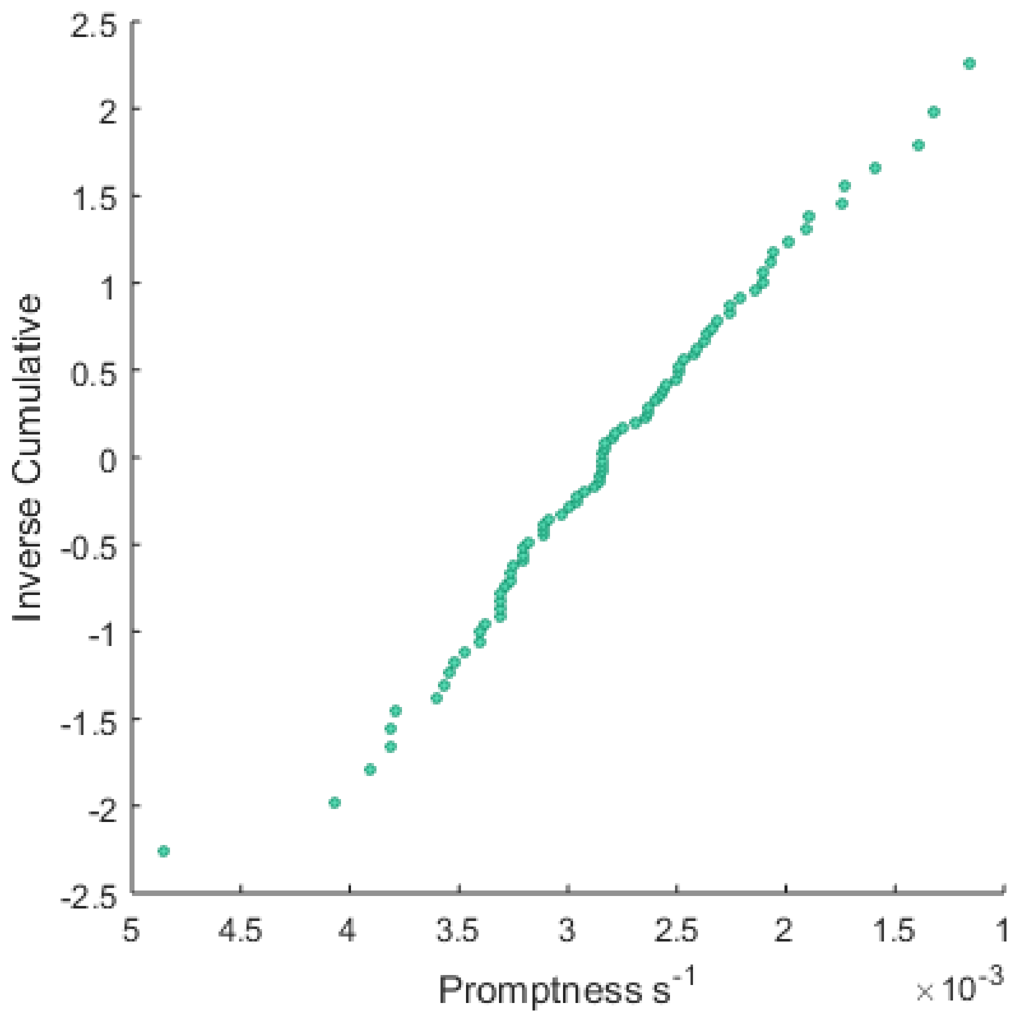
Cumulative Distribution




```
%Compute cumulative
sVisual = sort(visual);
n = numel(sVisual);
cumulativeY =(1:n)./(n+1);
plot(sVisual,cumulativeY);
```

Model 2: LATER Working with the Data

Inverse Cumulative Distribution




```
vYInv = norminv(cumulativeY,0,1);
plot(sVisual,vyInv);
```

Method 2: Ordinary Least Squares

Use *residuals* in order to compute the error between the model and the empirical data.

Given the data *y*, pick parameters for $f(X;\theta)$ that **minimizes** the sum of the residuals (J).

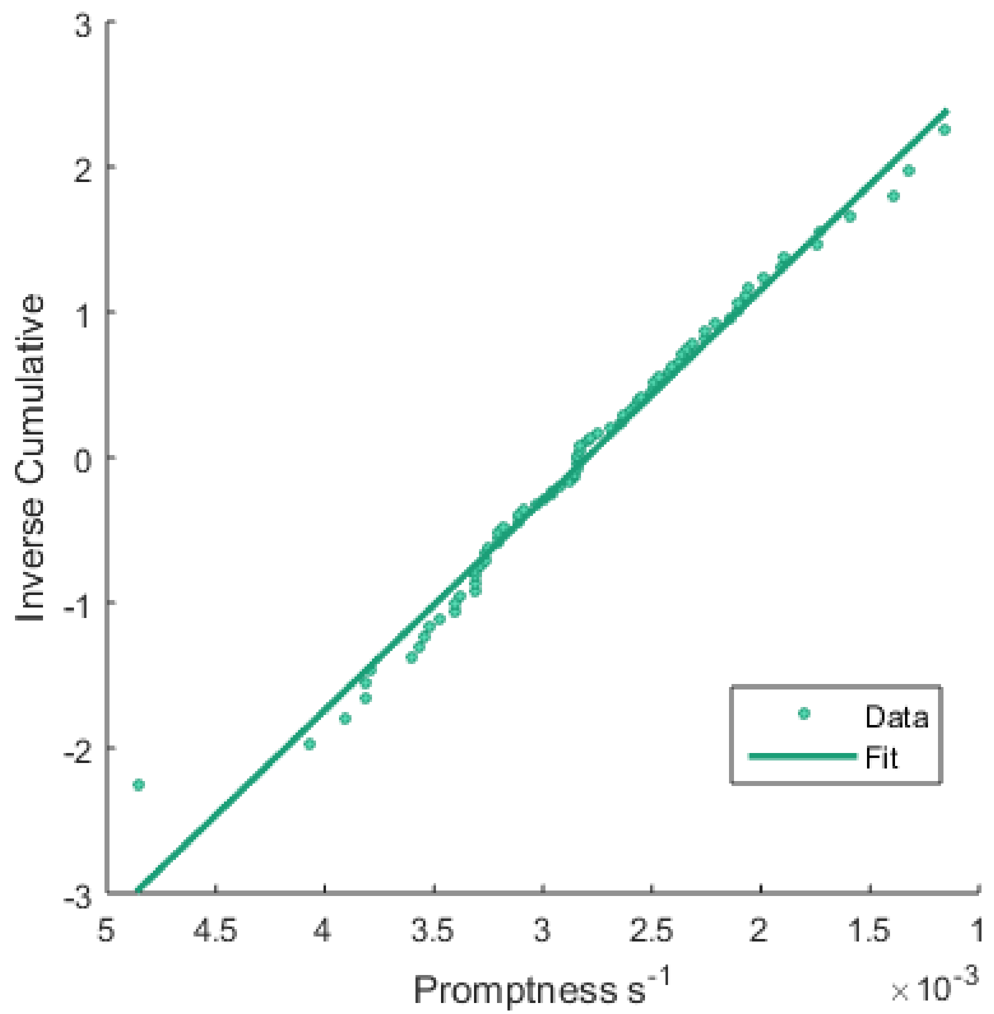
$J = \sum_{i=1}^{N}(f(X;\theta) - y_i)^2$ *where* $f(X;\theta) = \beta_0 + X^T\beta_1$

Optimization problem: Minimize *J*

MATLAB Implementation: regress, closed form : $(X^TX)\backslash X^TY$

Method 2: Ordinary Least Squares MATLAB Implementation

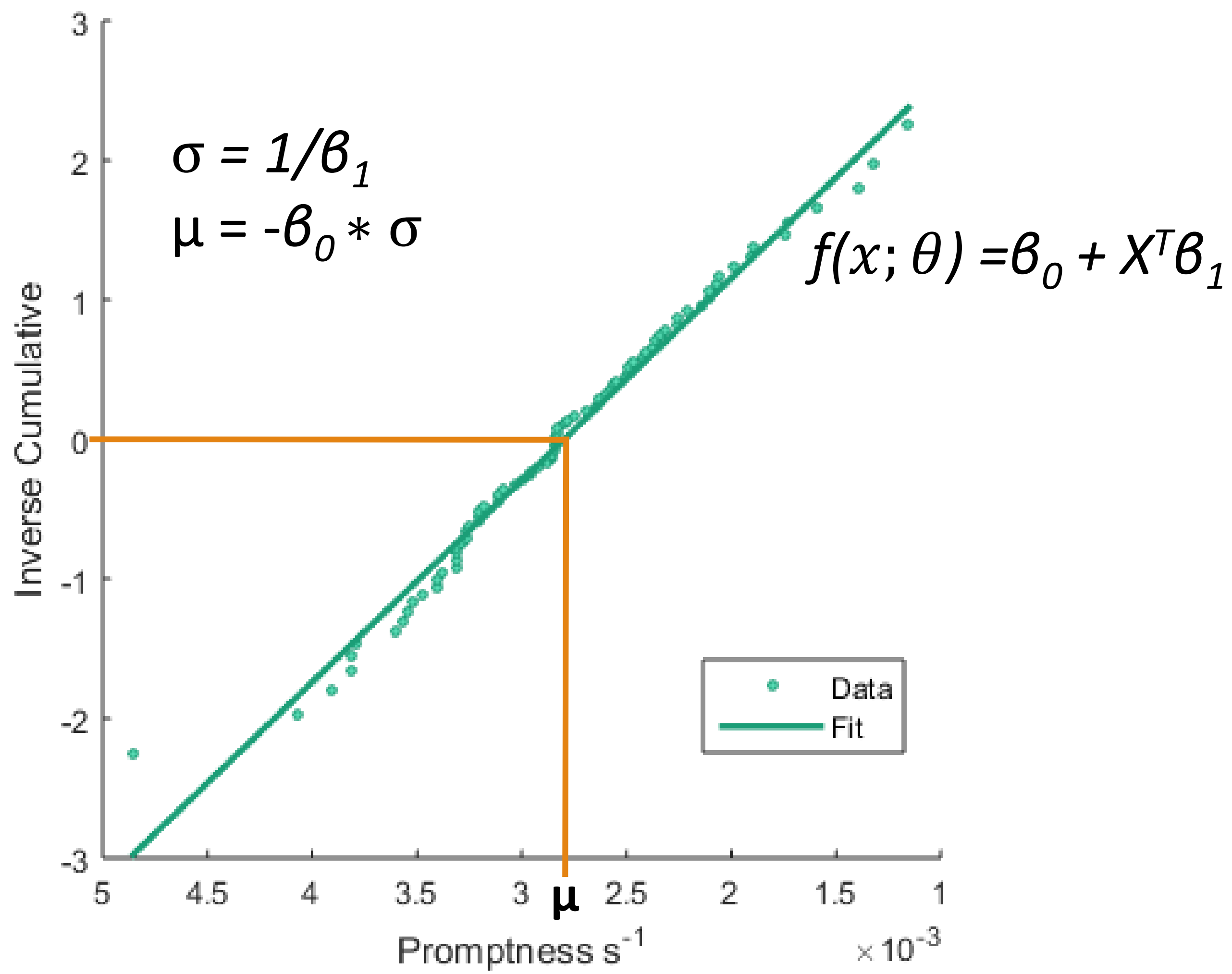
Ordinary Least Squares Regression




```
vF = [ones(size(visual,1)) visual]; %bias
bV = regress(vYInv',vF); %regression
```

Method 2: Ordinary Least Squares
MATLAB Implementation

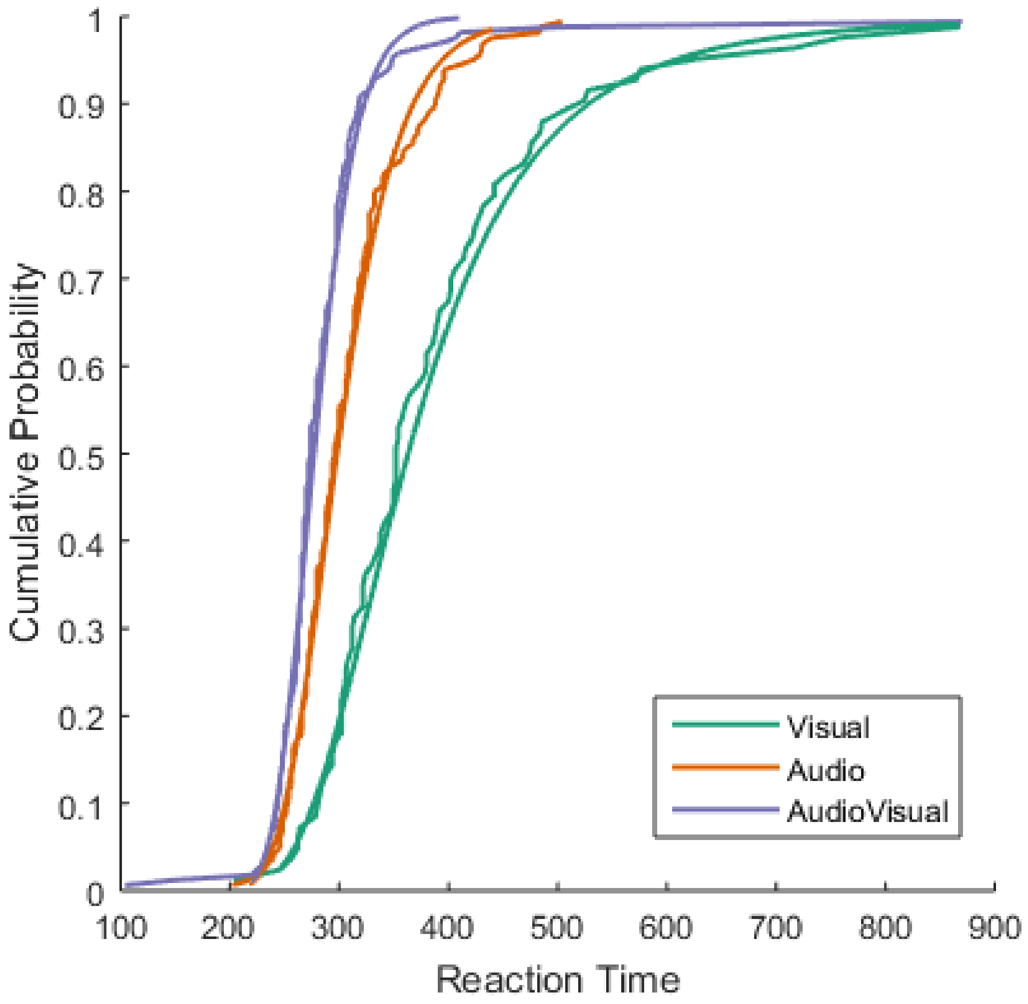
Making sense of parameters



Recall: $\frac{1}{T} = \frac{r}{\Delta S}$ where $r \sim \mathcal{N}(\mu, \sigma^2)$

Method 2: Ordinary Least Squares
MATLAB Implementation

Visualizing the Fit




```
%Set values to evaluate on
xFitRcip = (min(visual):((max(visual)-min(visual))/1000):max(visual))';
yp = 1-normcdf(xFitRcip,mu,sig); %Flip cumulative of reciprocal dist
xFit = 1./xFitRcip; %Take inverse promptness --> RT
```

How Much Confidence do we have in our Parameters?

Visual inspection isn't enough…!

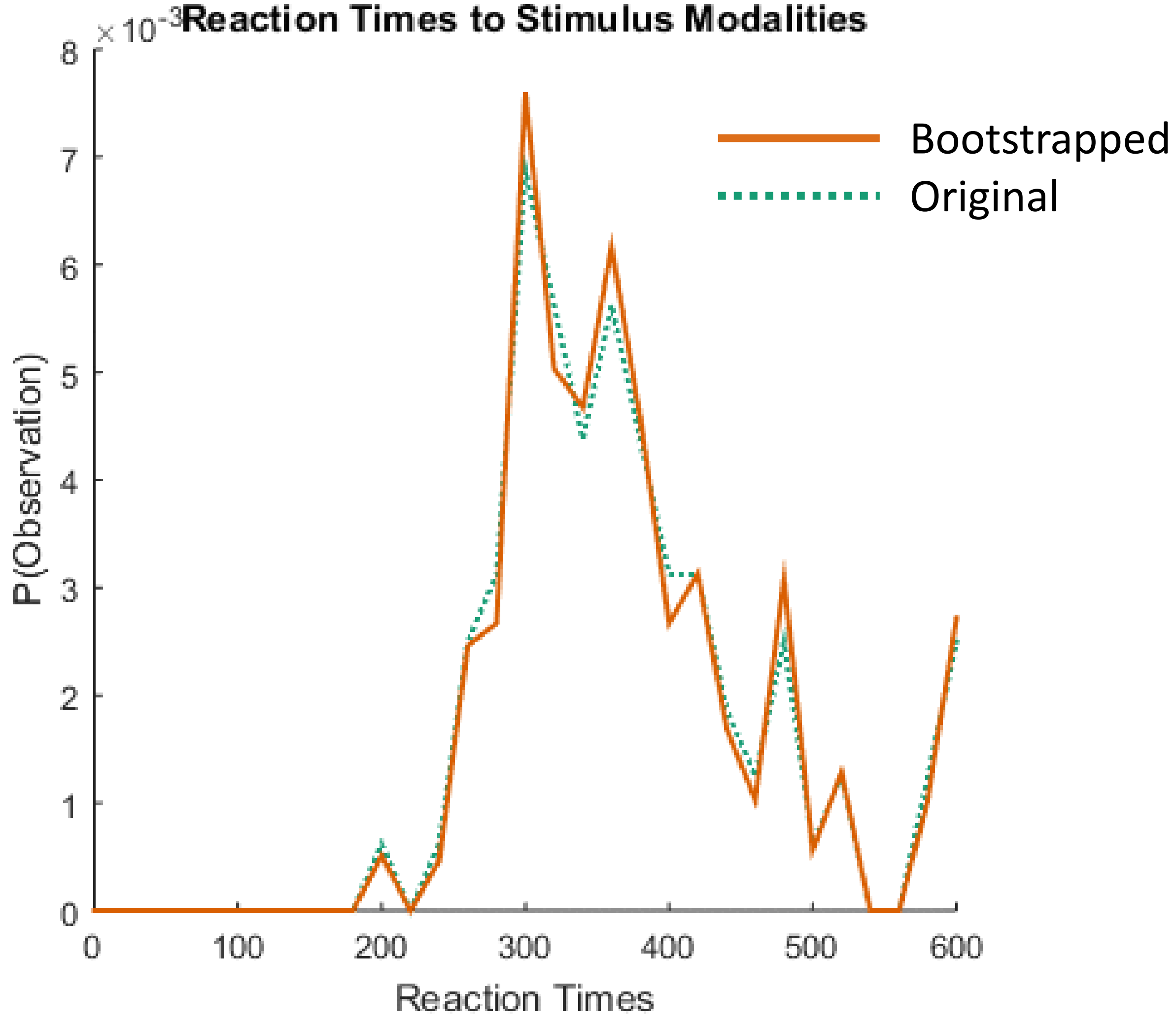
- Fit may have been biased by sample
- Quantify this!

How?

Bootstrapping!

Bootstrapping

The process of re-sampling with replacement to get a better approximation of the true distribution

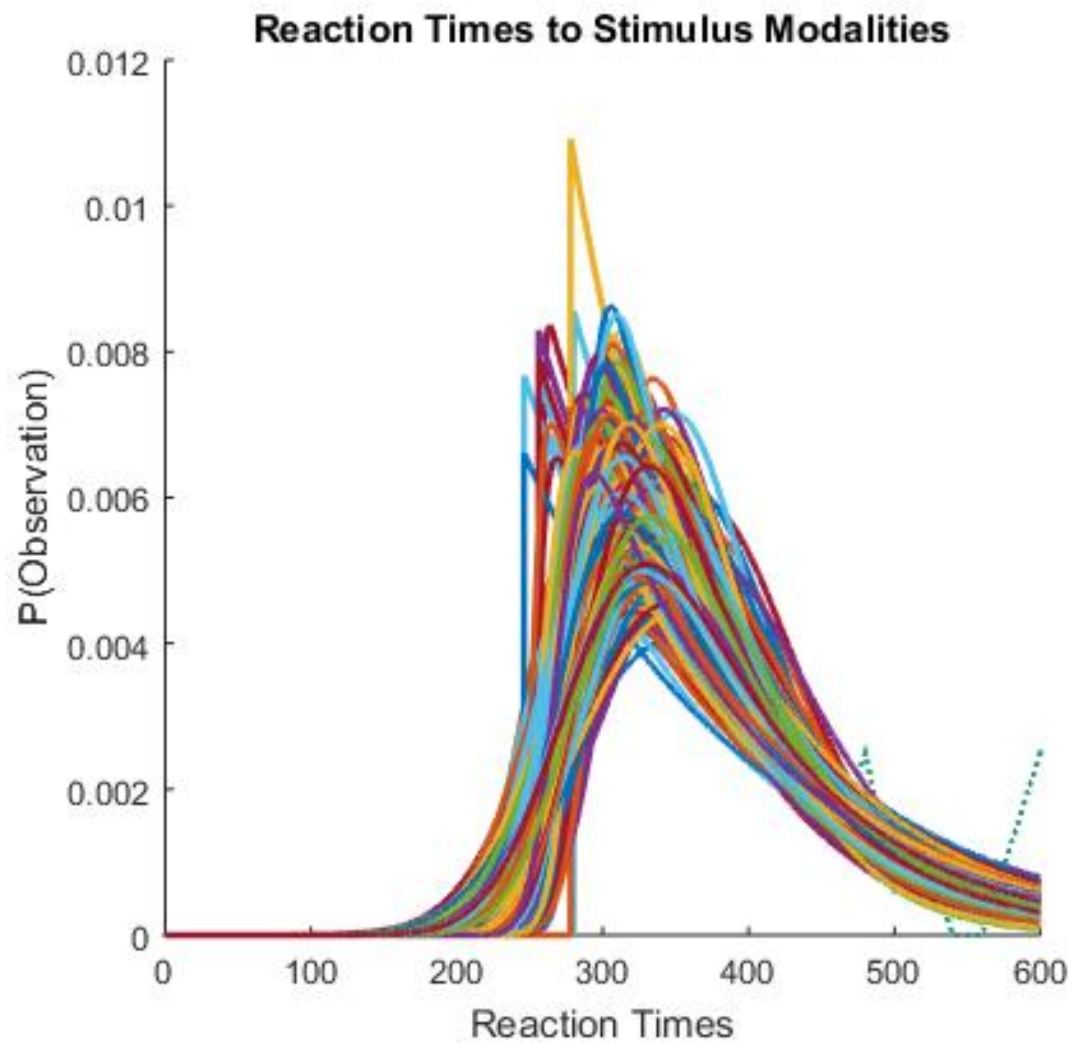



```
[~, bootSamp] = bootstrp(1000,@mean,visual);
```

Checking out our Confidence

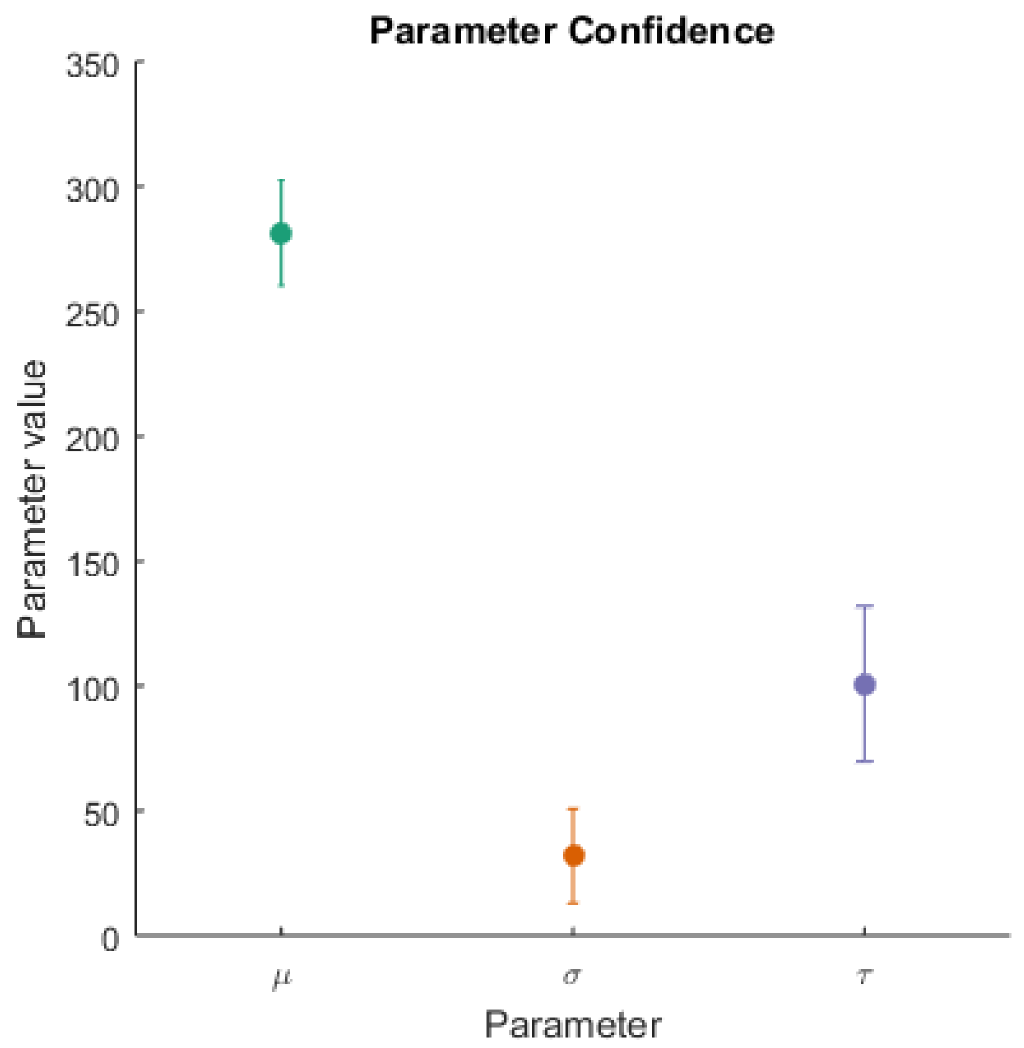
Ex-Gaussian fits from every parameter set

(n = 1000)



Confidence of Fitted Parameters

We can plot parameter values along with our 95% confidence bounds for a clearer picture




```
quanParam = quantile(visParams,[0.025 0.975],2);
meanParams = mean(visParams,2);
LowerBound = meanParams-quanParam(:,1);
UpperBound = quanParam(:,2)-meanParams;
```

Questions?