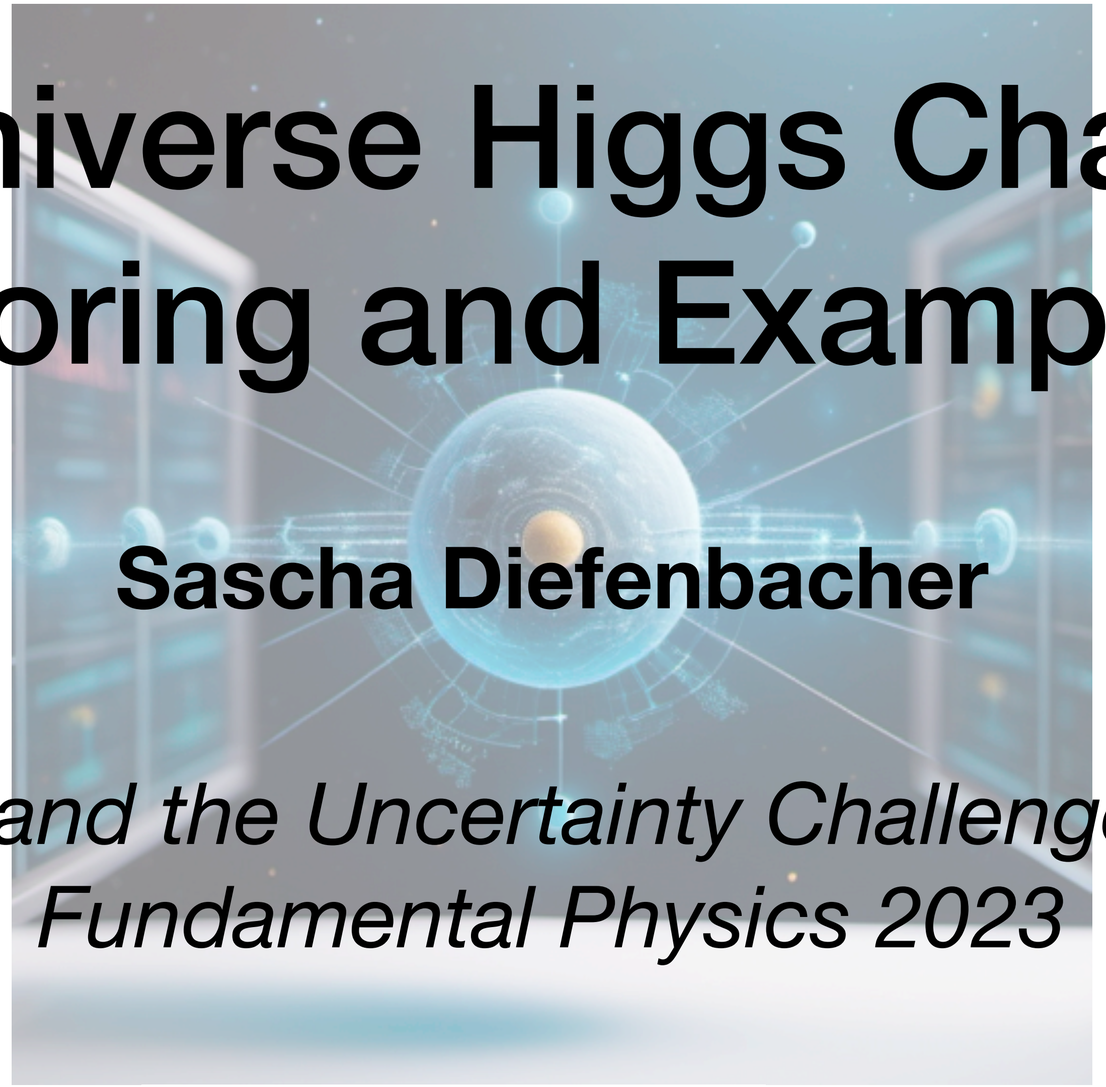
# Fair Universe Higgs Challenge Scoring and Examples



**Sascha Diefenbacher**

*AI and the Uncertainty Challenge in Fundamental Physics 2023*

BERKELEY LAB

FAIR Universe ML PLATFORM

NERSC

ATLAS EXPERIMENT

# Uncertainty Quantification Metrics

- Alternative Idea: Quantile score:
  - Method should return interval $[\mu_{16}, \mu_{84}]$
  - Corresponds to central 68% quantile of likelihood function
  - Also corresponds to interval defined by 1 standard deviation (under Gaussian uncertainty assumption)
  - Interval can also be defined with Bayesian methods that output a posterior

# Uncertainty Quantification Metrics

- For N test sets and predicted $[\mu_{16}, \mu_{84}]_i, i \in [0,N]$
  - Calculate fraction of times interval contains $\mu_{\text{true}}$ to get coverage c:

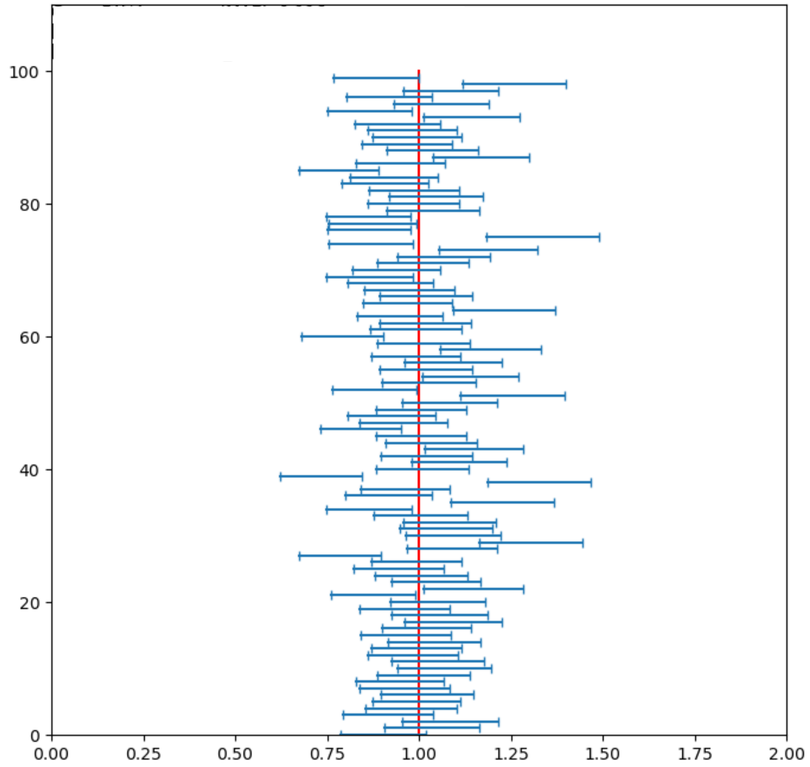
$$c = \frac{1}{N} \sum_{i=0}^{N} 1 \text{ if}(\mu_{\text{true},i} \in [\mu_{84} - \mu_{16}]_i)$$

  - Calculate average interval width w:

$$w = \frac{1}{N} \sum_{i=0}^{N} \mu_{84,i} - \mu_{16,i}$$

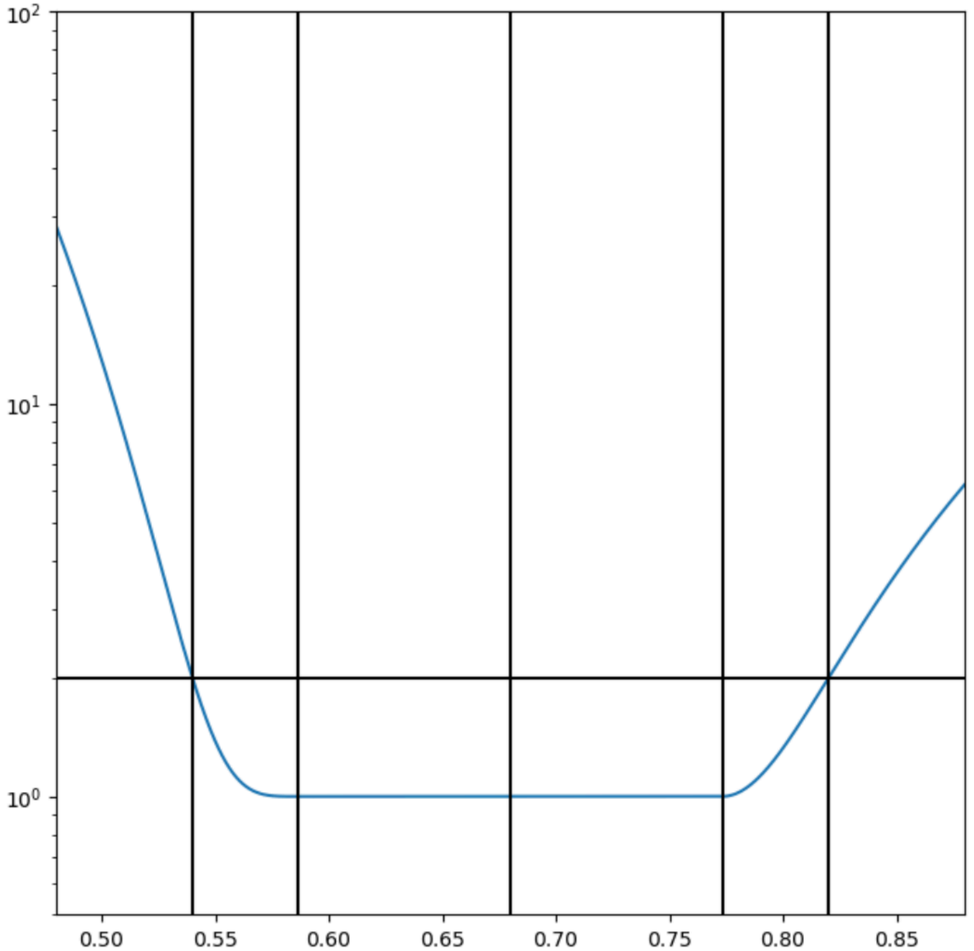
  - Combine both values for score s:

$$s = w\, f(c)$$



# Uncertainty Quantification Metrics

- Combine both values for score $s = w\ f(c)$
- Scaling function $f$:
  - Ideal coverage: 0.68 (68% interval)
  - $f = 1$ around $c = 0.68$
  - Power scaling outside of $c = 0.68$
  - Stricter penalty for undershooting



# Uncertainty Quantification Metrics

- Combine both values for score $s = w\ f(c)$
- Scaling function $f$:

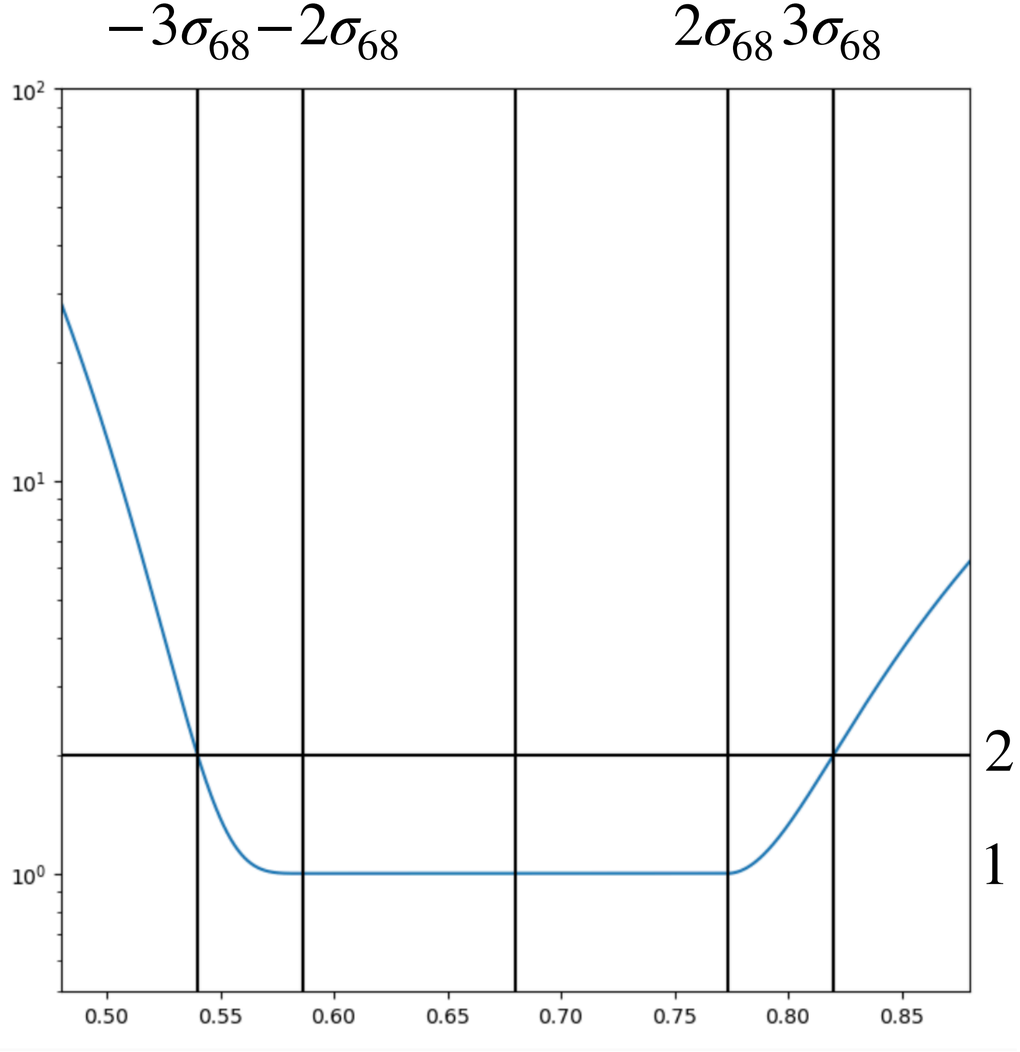
$$x \geq 0.68 - 2\sigma_{68} \text{ and } x \leq 0.68 + 2\sigma_{68} : 1.$$

$$x < 0.68 - 2\sigma_{68} : 1 + \left|\frac{x-(0.68-2\sigma_{68})}{\sigma_{68}}\right|^4$$

$$x > 0.68 + 2\sigma_{68} : 1 + \left|\frac{x-(0.68+2\sigma_{68})}{\sigma_{68}}\right|^3$$

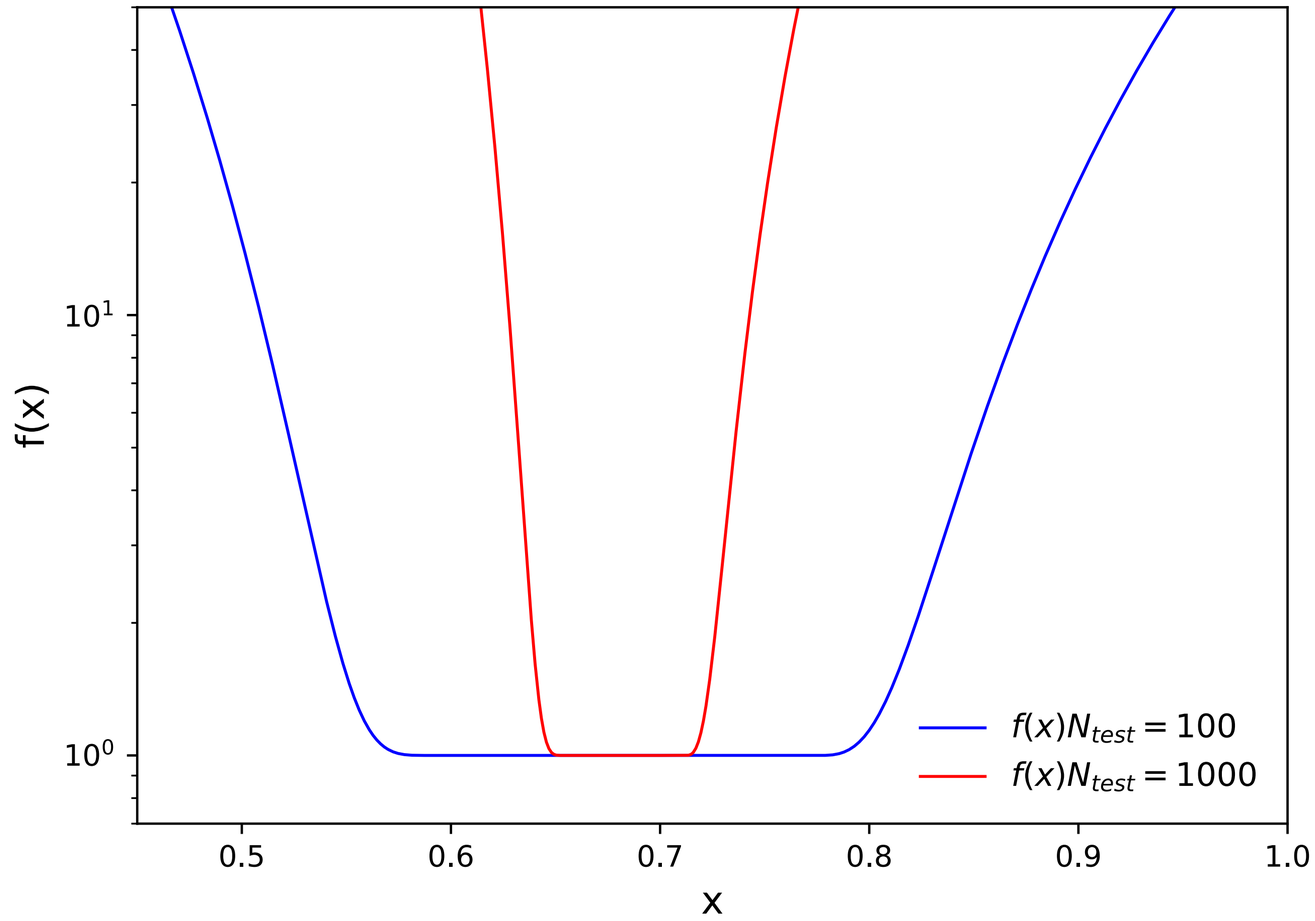
$$\text{with } \sigma_{68} = \frac{\sqrt{(1-0.68)0.68N)}}{N}$$

- Uncertainty for the coverage assumes binomial error on coverage value distr.
- For perfect coverage: perfect score 95% of the time, good score 99% of the time



# Uncertainty Quantification Metrics

- Larger N_test: smaller ‘good’ range



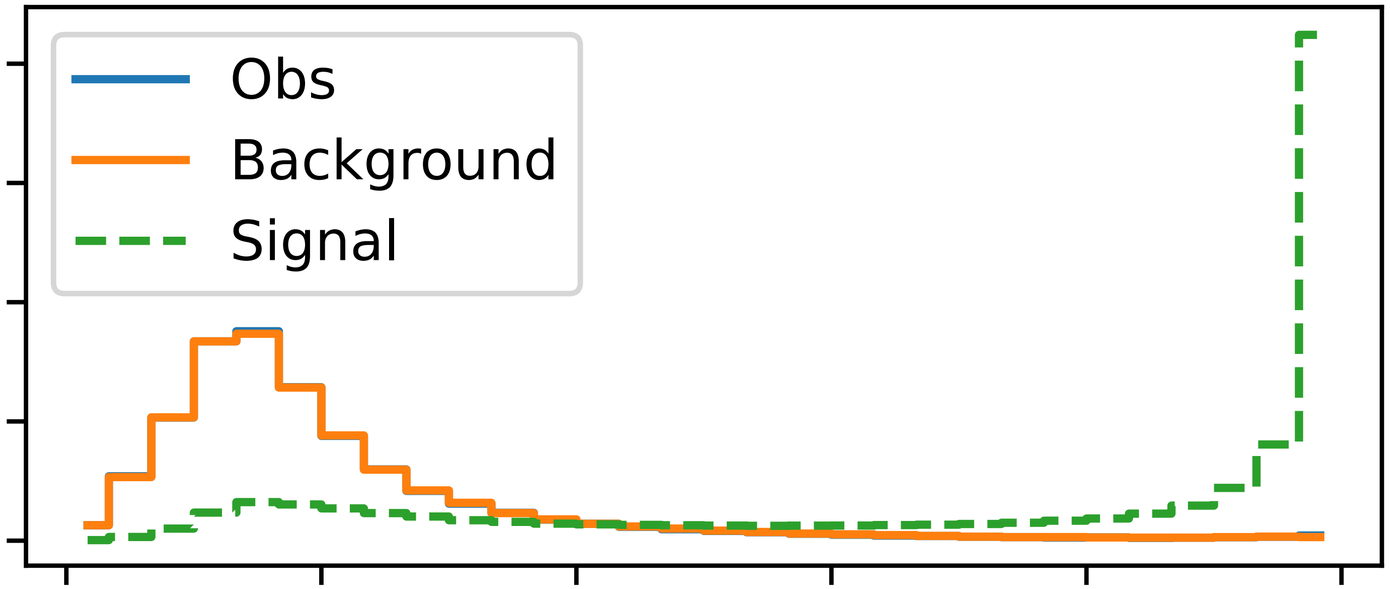
# Uncertainty Quantification Metrics

- 3 remaining problems with $s = w\, f(c)$:
  1. Scores can become very large
  2. Lower scores winning is unintuitive
     ➡ $s = -\ln[w\, f(c)]$
  3. Methods that return $\mu_{16} = \mu_{84}$ always win,
     since $w = 0$ ➡ $s = \inf$
     ➡ $s = -\ln[(w + \epsilon)\, f(c)]$
     choose $\epsilon$ significantly smaller than
     minimal width $\epsilon = 10^{-3}$ in the challenge

| | | | | |
|---|---|---|---|---|
| 1 | Ihsan Ullah | 2 | 2023-11-02 | [redacted] |
| 2 | ragansu | 5 | 2023-11-03 | 515061194.28 |
| 3 | ragansu | 5 | 2023-11-03 | [redacted] |

# Example: Sysaware Templates

- Train classifier on signal vs. background
- Define template histograms for signal and for background
- Apply same network to test set
- Perform Bin-wise NLL calculation to determine $\mu$



# Example: Sysaware Templates

- Train classifier on signal vs. background

```
class ClassifierNet(torch.nn.Module):
    def __init__(self, input_dim = 33, layer_widths = [n_nodes_awe,]*n_hidden_awe + [1,]):
        super(ClassifierNet, self, ).__init__()

        self.input_dim = input_dim
        self.layer_widths = layer_widths

        self.activation_fn = F.leaky_relu

        self.activate_final = F.sigmoid

        self.layers = []
        prev_width = self.input_dim
        for layer_width in self.layer_widths:
            self.layers.append(torch.nn.Linear(prev_width, layer_width))
            prev_width = layer_width

        self.layers = torch.nn.ModuleList(self.layers)


    def forward(self, x):
        for i, layer in enumerate(self.layers[:-1]):
            x = self.activation_fn(layer(x))
        x = self.layers[-1](x)


        if not (self.activate_final is None):
            #print(x)

            x = self.activate_final(x)
        return x
```

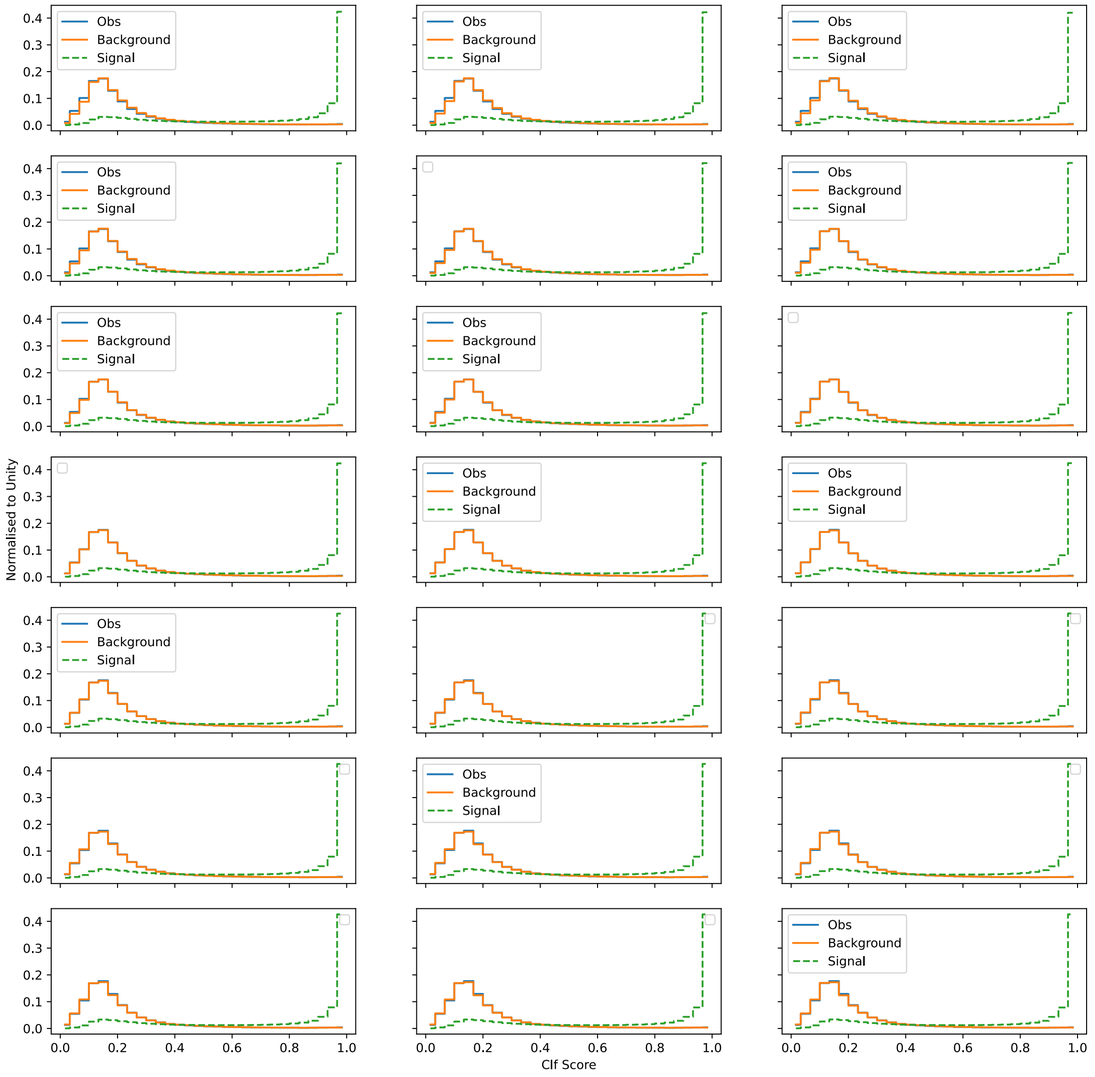
# Example: Sysaware Templates

- Perform Bin-wise NLL calculation to determine $\mu$

```
def nll_stat(N,S,B,mu):
    from scipy.special import loggamma
    nexp=mu*S+B
    nll_stat = - (N* np.log(nexp) - nexp - loggamma(N) ) # Poisson
    if np.isinf(nll_stat):
        pass
        #print (N, nexp)
    return nll_stat
```

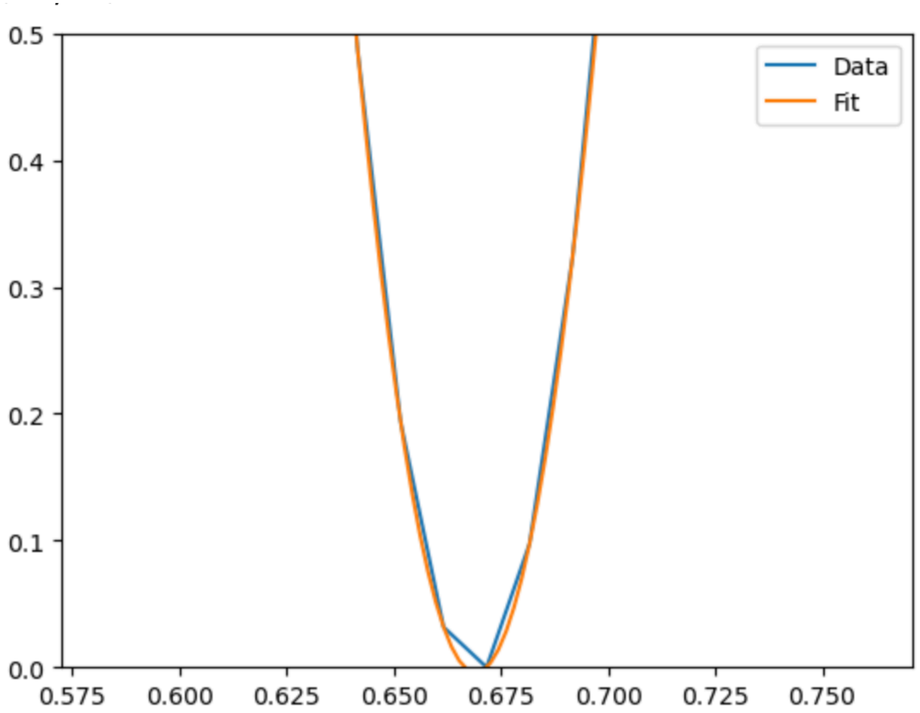
# Example: Sysaware Templates

- Sysaware modification: train network conditional on TES
- Generate template for each TES value
- Perform fit to both TES and $\mu$



# Example: Sysaware Templates

- Interval from binned NLL values:
  - Polynomial fit to NLL
  - Analytically calculate crossover point with NLL-min(NLL)=0.5



# Example: Sysaware Templates

```python
def ConfidenceIntervalls(mu_values, nll1D, central=1.0, epoch=100, fitrange = 0.1, do_plot=False):
    mu_values_plt = mu_values[np.abs(mu_values-central)<=fitrange]

    z = np.polyfit(mu_values[np.abs(mu_values-central)<=fitrange], nll1D[np.abs(mu_values-central)<=fitrange], 4)
    poly = np.poly1d(z)
    #1 Sigma = 0.5, 2 Sigma = 2, 3 Sigma = 4.5
    threshs = [0.5, 2.0, 4.5]
    #threshs = [0.25, 1, 2.25]
    sigmas = [1.0, 2.0, 3.0]
    ret_intersects = []
    std = 0.0

    deriv1 = poly.deriv()
    deriv2 = poly.deriv(2)

    extr = (deriv1).roots
    minimum = extr[np.argsort(np.abs(extr-central))][0]

    #assert deriv2(minimum) > 0


    for t,s  in zip(threshs, sigmas):
        intersects = (poly - t).roots
        # Select 2 roots closest to central value (only needed for polynomials with order > 2
        ret_intersects.append(intersects[np.argsort(np.abs(intersects-central))][:2])
        tmp_std = (abs(ret_intersects[-1][0] - ret_intersects[-1][1])/(s*2))
        #print(tmp_std)
        std = std + tmp_std

    std = std/len(sigmas)
    mu_values_plt = mu_values[np.abs(mu_values-central)<=fitrange]
```