Modeling Extreme Events with PyMC

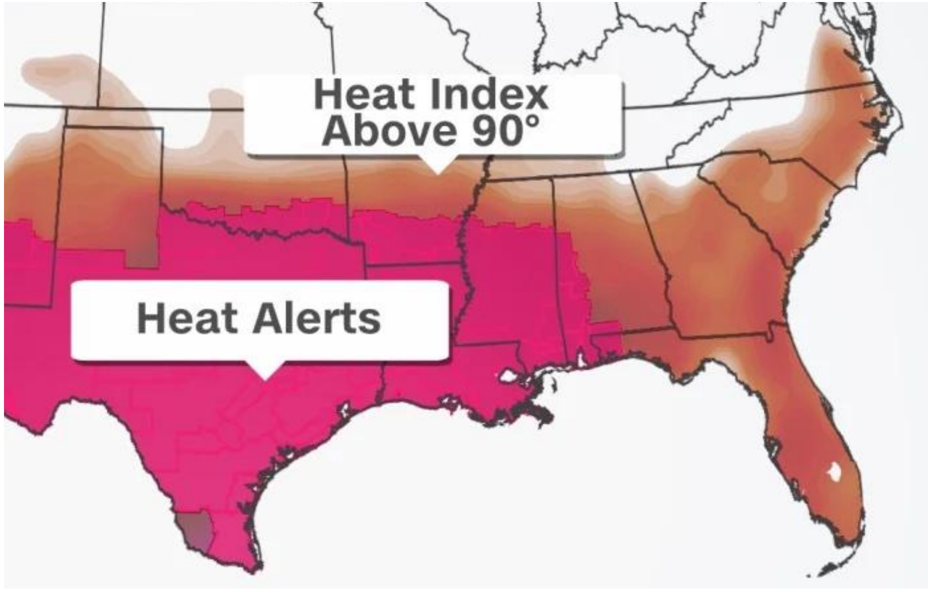
A Bayesian approach to modeling extreme rainfall

Jorn Mossel, Ph.D.
PyData Global 2023

Recent Extreme Weather Events

- **Extreme Temperatures**
- Droughts
- Flash Floods

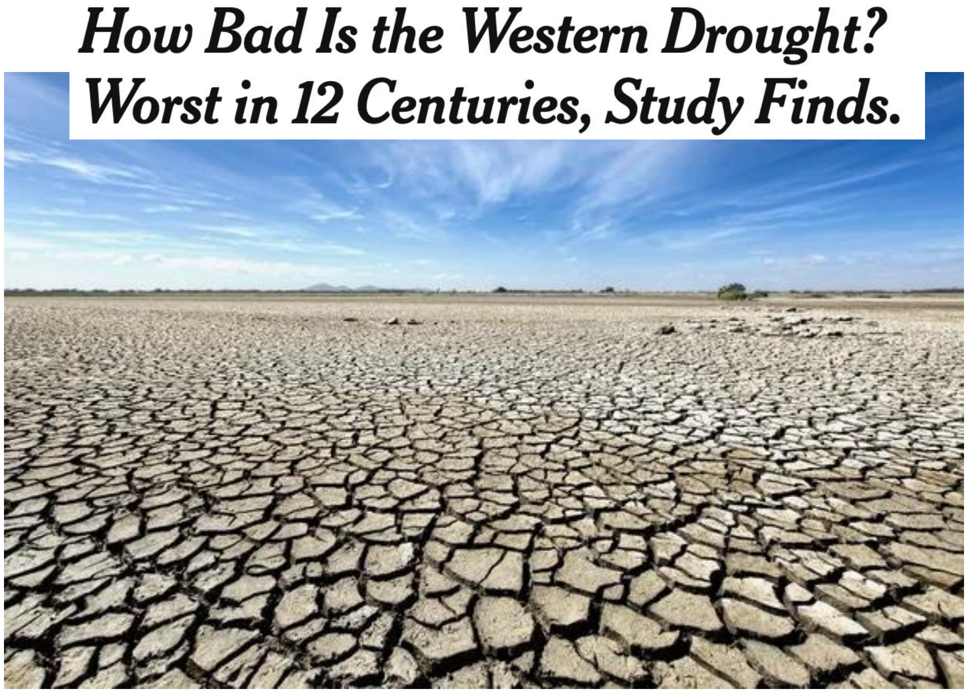
‘Ridiculous’ heat keeps tormenting Texas, with no end in sight



August 2023

Recent Extreme Weather Events

- Extreme Temperatures
- **Droughts**
- Flash Floods



February 2022

Recent Extreme Weather Events

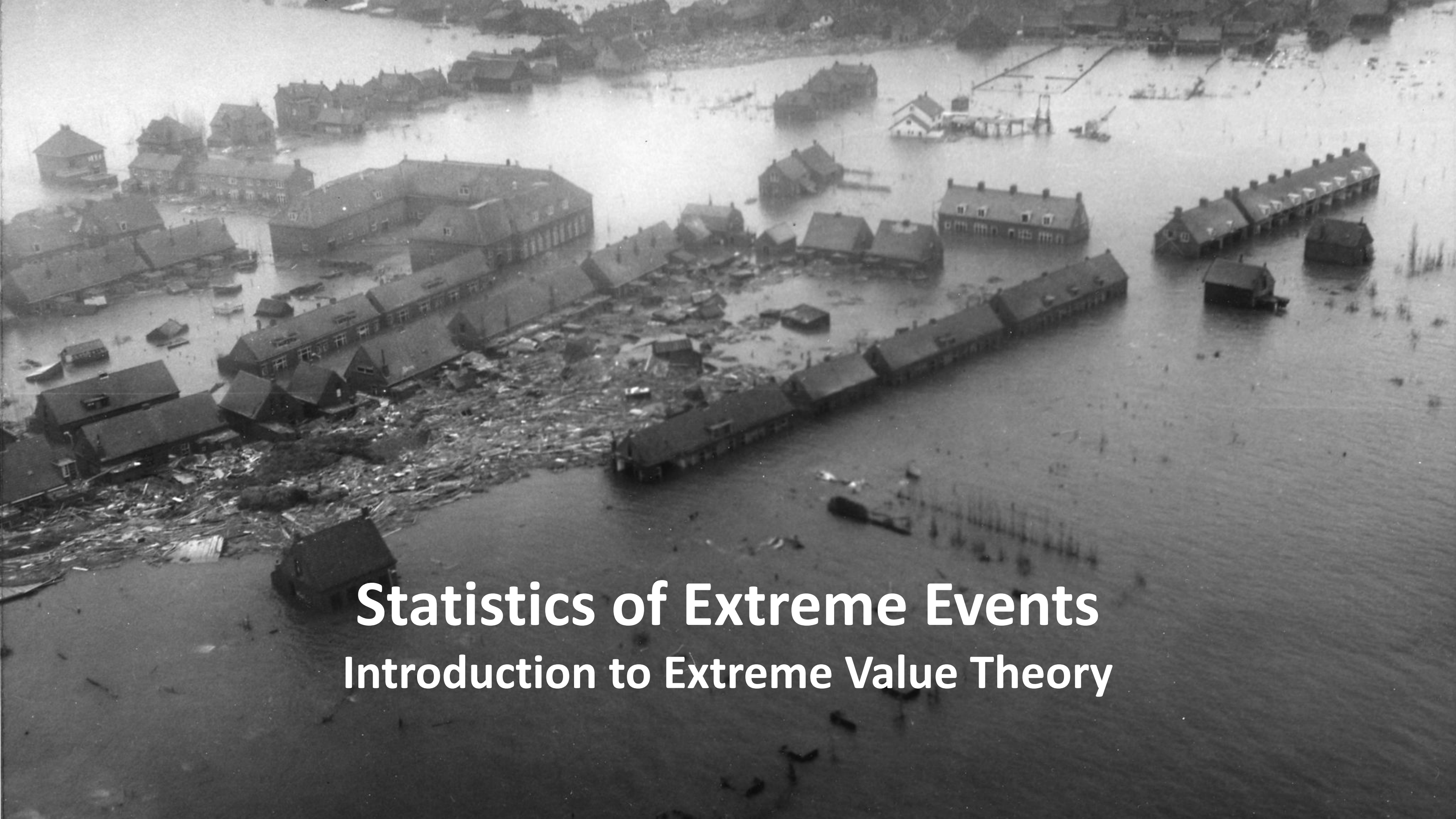
- Extreme Temperatures
- Droughts
- **Flash Floods**

At least 43 are dead after Ida causes flooding in four states.

September 2021

Overview

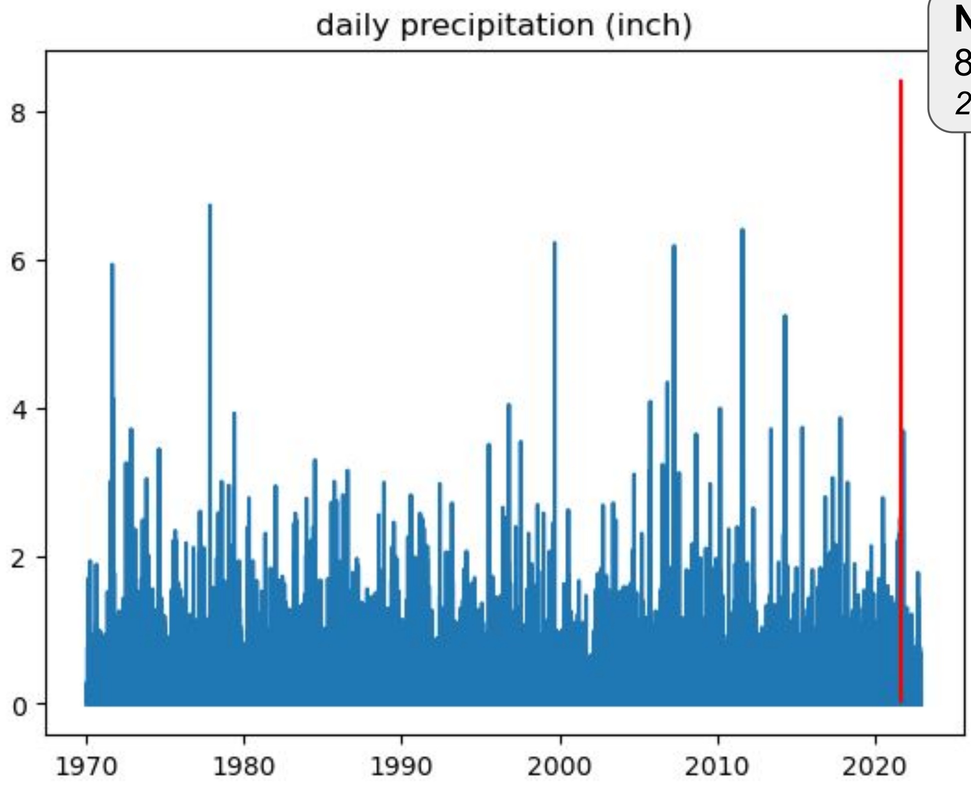
1. Statistics of Extreme Events
2. Intro to Bayesian Modelling with PyMC
3. Example: Extreme Rainfall in NYC
4. Bonus example: Gaussian Processes



Statistics of Extreme Events

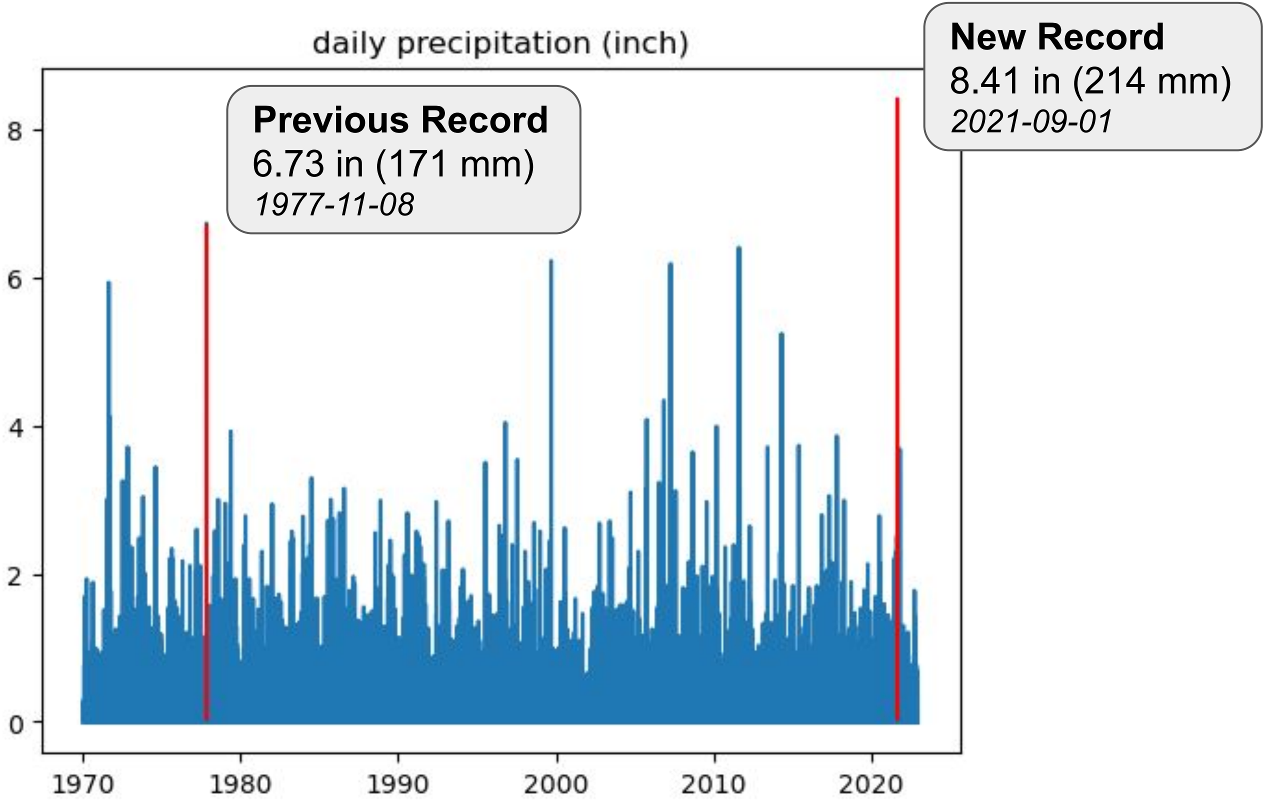
Introduction to Extreme Value Theory

Record rainfall at nearby Newark airport after Hurricane Ida

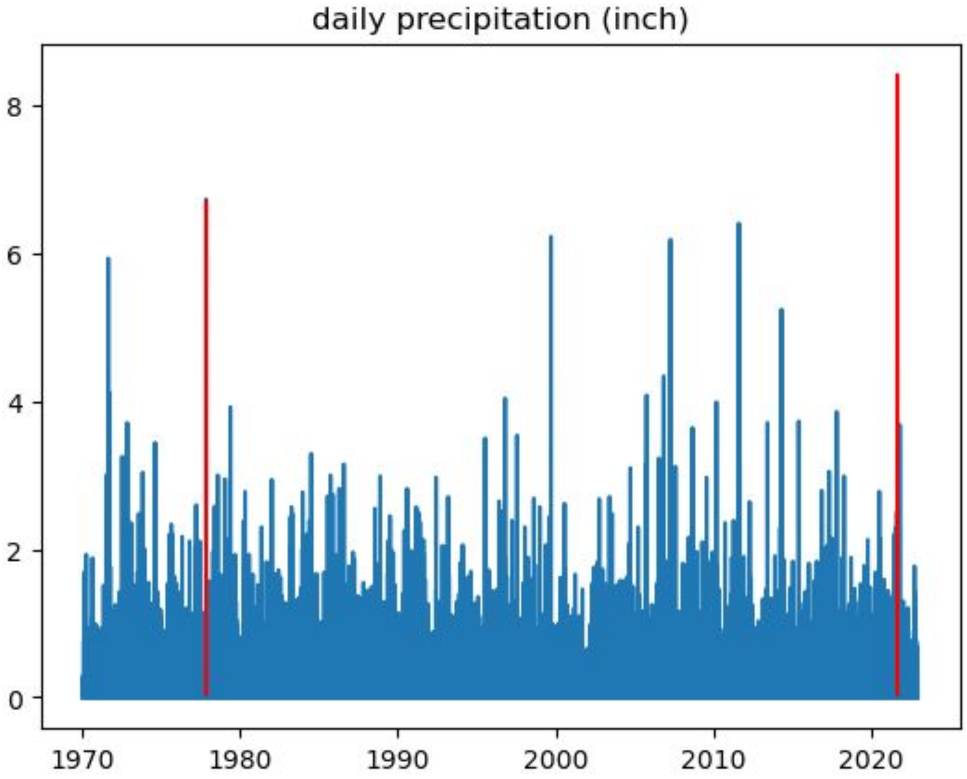


New Record
8.41 in (214 mm)
2021-09-01

Record rainfall at nearby Newark airport after Hurricane Ida



How can we define what is extreme?



Q: How extreme is this peak?

A: A better question is to ask how often we expect to observe a level of this or higher?

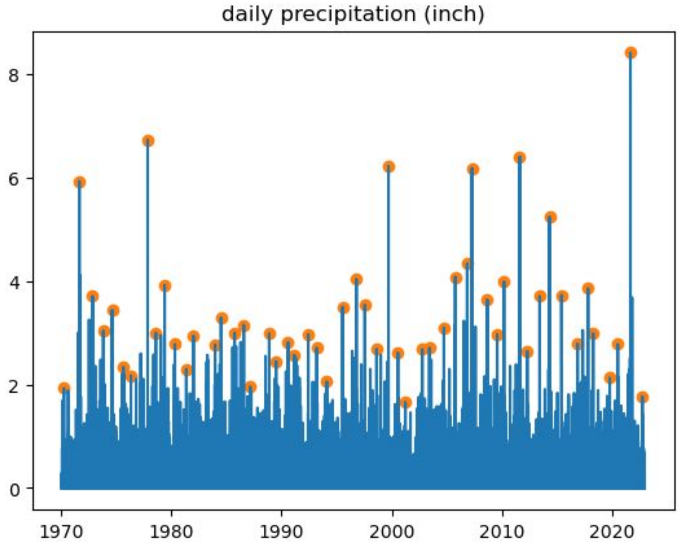
Q: Isn't that only possible if we have really long history?

A: Not if you know the probability distribution

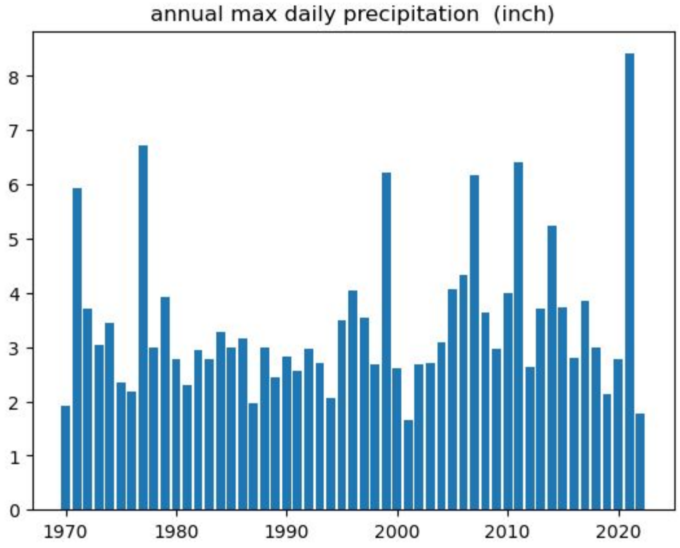
Q: How do you know which distribution to fit?

A: Extreme Value Theory will tells us

Finding a distribution for the extremes



Keep only annual maxima



Complicated (unknown) distribution

Extreme Value distribution (see next slide)

Extreme Value Theory states that

*The probability distribution of **maxima** (e.g. annual maxima of daily observations) can be approximated (under general assumptions) with a **Extreme Value distribution** which has a CDF of the form:*

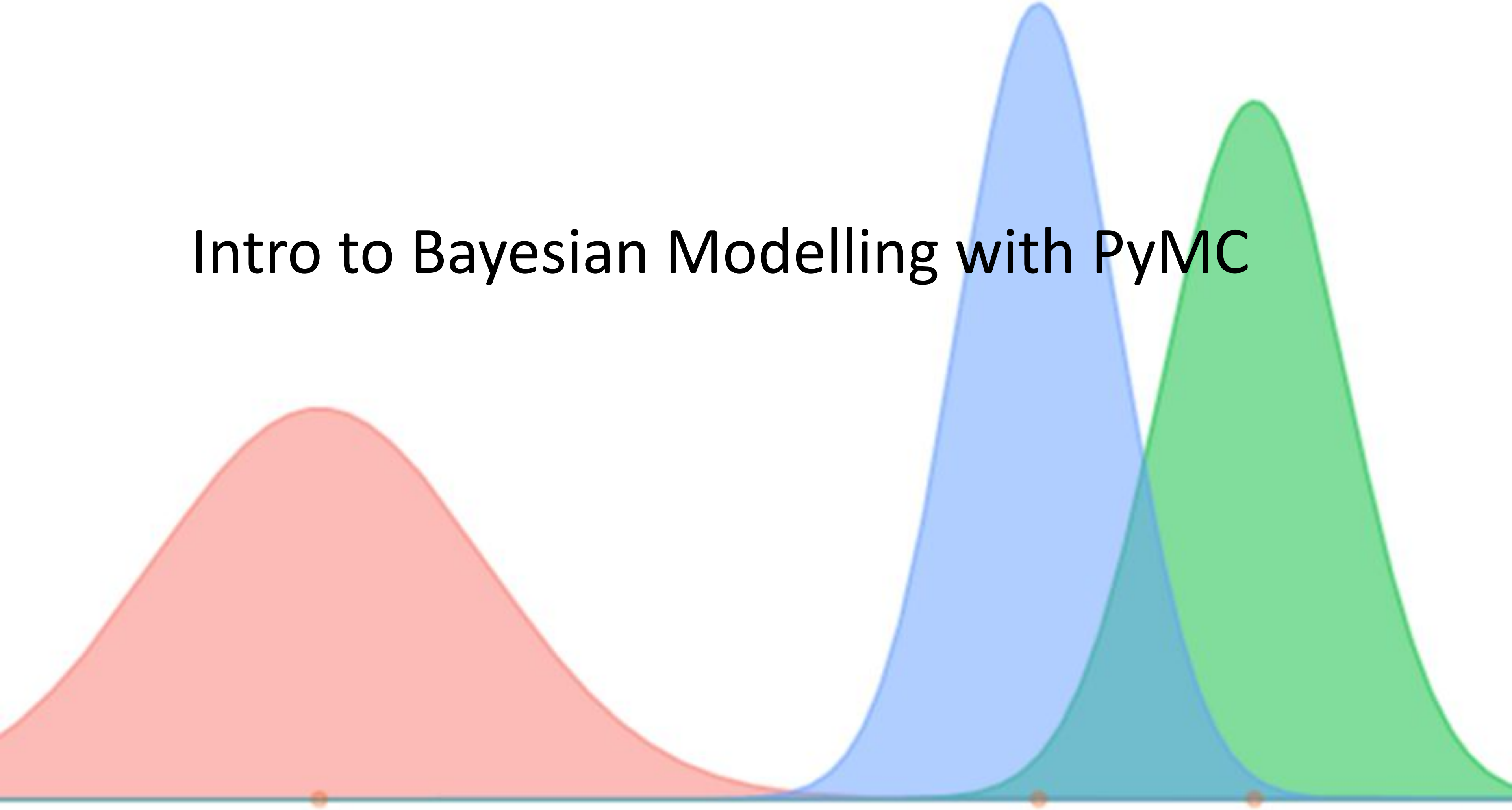
$$G(z) = \exp\left\{ -\left[1 + \xi \left(\frac{z-\mu}{\sigma} \right) \right]^{-1/\xi} \right\}$$

Regardless of what the underlying process is. Which can be max precipitation, temperature extremes, stock market crashes, etc.

Our task is to fit the parameters of this distribution to the observed data

- μ “location”
- σ “scale”
- ξ “shape (tailness)”

Intro to Bayesian Modelling with PyMC



Intro to Bayesian Modelling

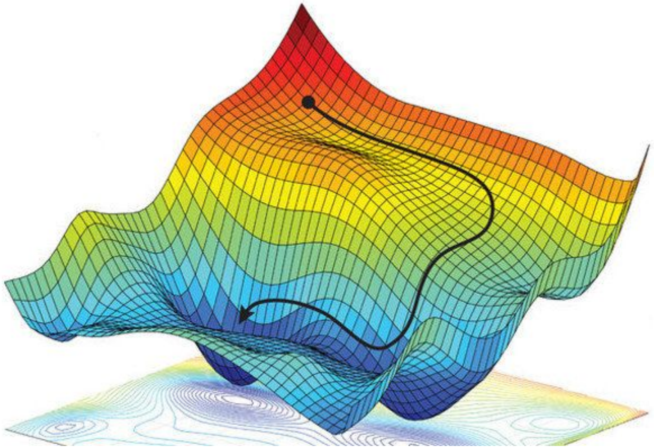
Non-bayesian methods

Most ML methods are solving for point estimates of model parameters by

Minimizing a loss function with gradient descent

With this approach the following is hard (in general)

- Uncertainty estimation (eg confidence intervals)
- Taking prior information into account



Intro to Bayesian Modelling

Bayesian methods

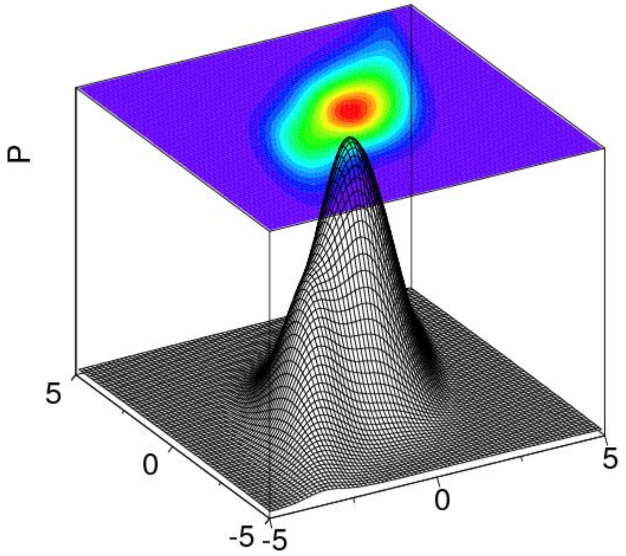
Find a *posterior distribution* for the model parameters θ

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{\int P(X|\theta)P(\theta)d\theta}$$

Given data *X*

- *Uncertainty* is captured in $P(\theta|X)$
- *Prior information* can be put into $P(\theta)$

"Fitting" is done via sampling from the distribution



What is PyMC?

PyMC is an open source library for probabilistic programming in Python.

- Intuitive way to express Bayesian models in code
- Fit the model using Monte Carlo Markov Chain (MCMC) sampling
- Includes tools for visualization and diagnostics

Example: Extreme Rainfall in NYC

Fit the Extreme Value Distribution in PyMC

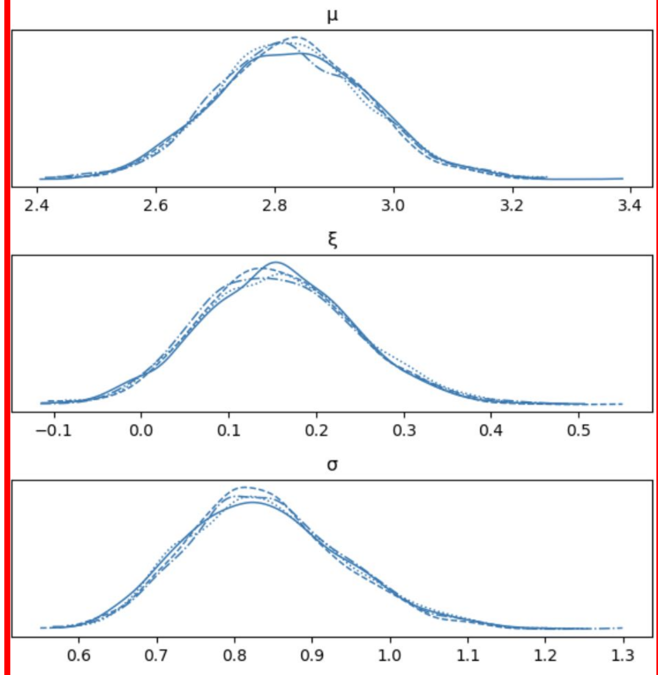
$$\boxed{P(\mu, \sigma, \xi | z)} \propto \boxed{P(z | \mu, \sigma, \xi)}\boxed{P(\mu, \sigma, \xi)}$$

```python
with pm.Model() as model:
    # Priors
    μ = pm.Normal("μ", mu=3, sigma=3.0)
    σ = pm.HalfNormal("σ", sigma=1.0)
    ξ = pm.Normal('ξ', mu=0.0, sigma=0.2)
    # Exteme Value likelihood
    gev = pmx.GenExtreme("gev", mu=μ, sigma=σ, xi=ξ, observed=z)
    # MCMC sampling
    trace = pm.sample(2000, target_accept = 0.98)
```

Weakly informative priors are used, with the aim to speed up the sampling without biasing the results.

The likelihood is the Extreme Value Distribution

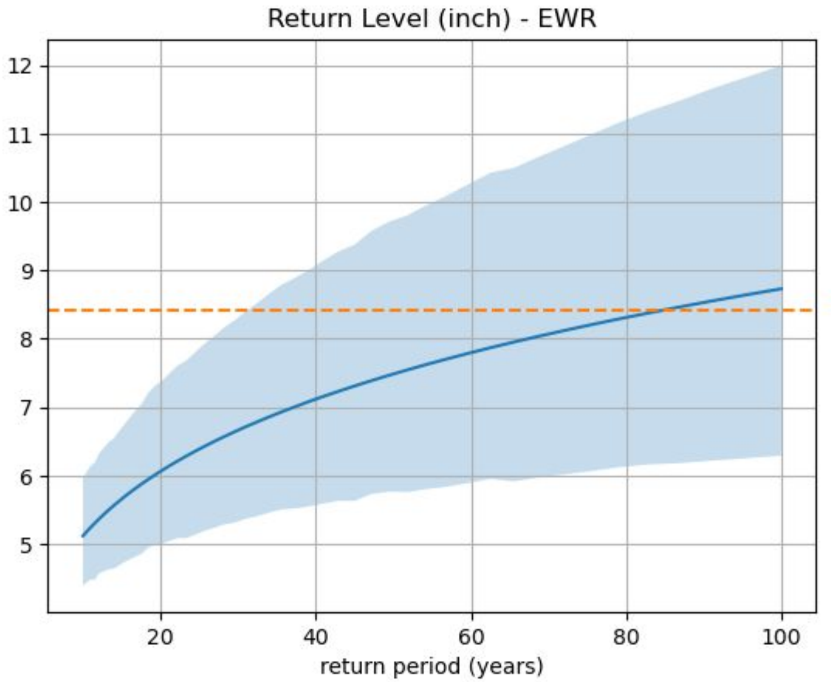
After sampling we obtain posterior distributions for the parameters.




```python
az.plot_trace(trace);
```

Return Levels

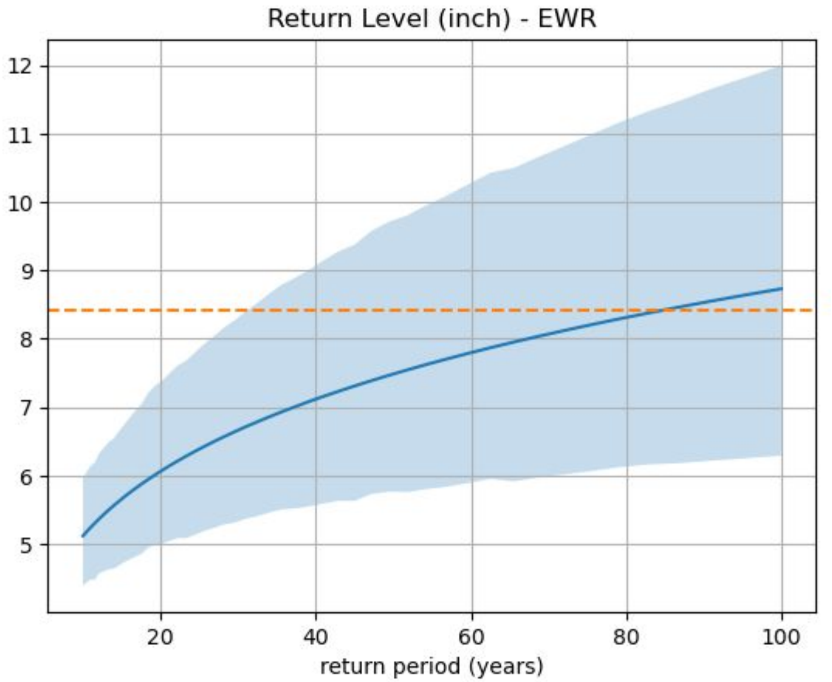
- For a **return period** (say every 100 years) we expect to exceed the **return level**



$$P(Z > \text{return level}) = \frac{1}{\text{return period}}$$

Return Levels

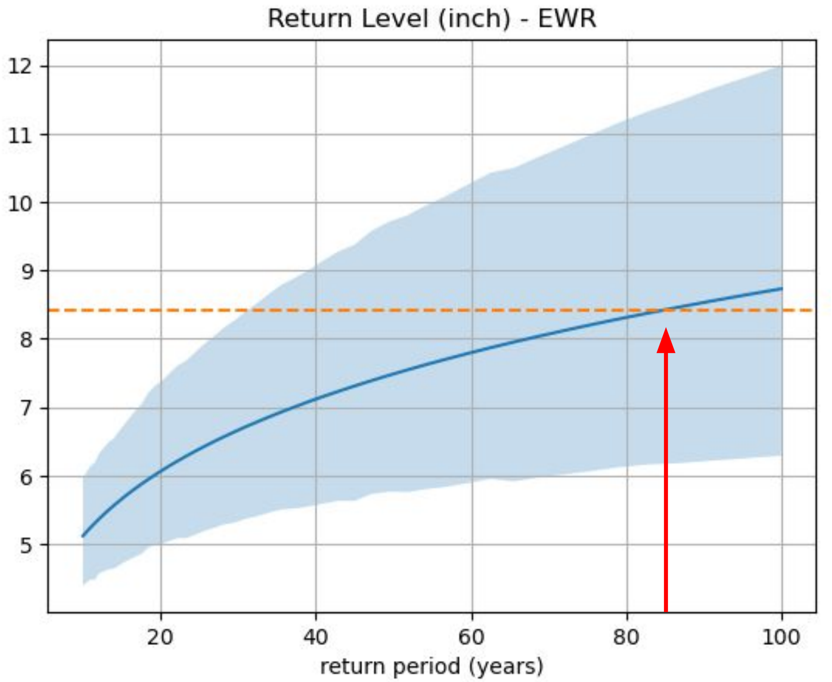
- For a **return period** (say every 100 years) we expect to exceed the **return level**
- After fitting the distribution in PyMC it's straightforward to compute statistics like return levels.




```
with model:
    rp = pm.ConstantData("rp", return_periods)
    rl = pm.Deterministic("rl", μ - σ/ξ * (1 - (-np.log(1 - 1/rp)) ** (-ξ)))
    posterior_pred = pm.sample_posterior_predictive(trace,var_names=['rl'])
```

Return Levels

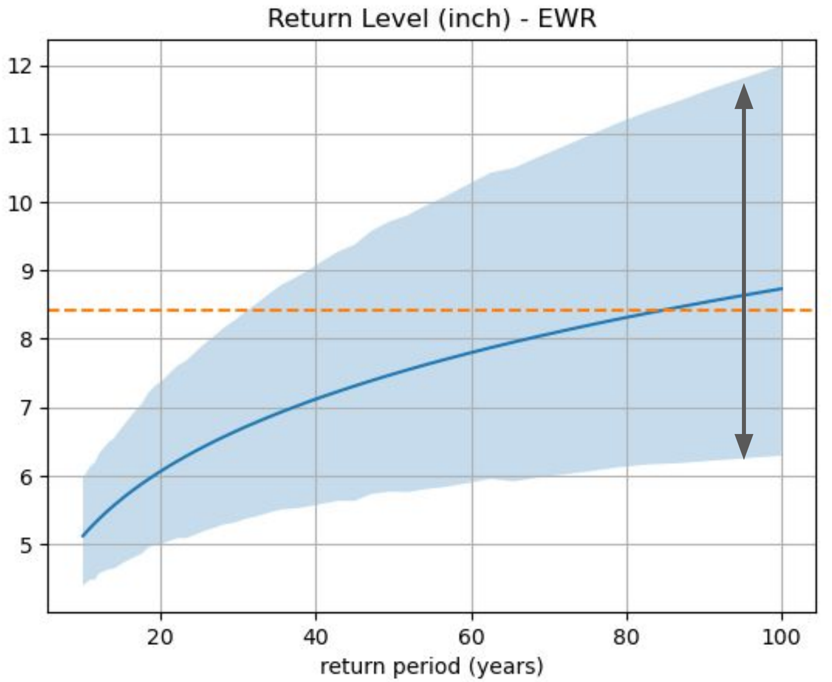
- For a **return period** (say every 100 years) we expect to exceed the **return level**
- After fitting the distribution in PyMC it's straightforward to compute statistics like return levels.
- The Newark record (8.4 inch) corresponds to a return period of 83 years.




```python
with model:
    rp = pm.ConstantData("rp", return_periods)
    rl = pm.Deterministic("rl", μ - σ/ξ * (1 - (-np.log(1 - 1/rp)) ** (-ξ)))
    posterior_pred = pm.sample_posterior_predictive(trace,var_names=['rl'])
```

Return Levels

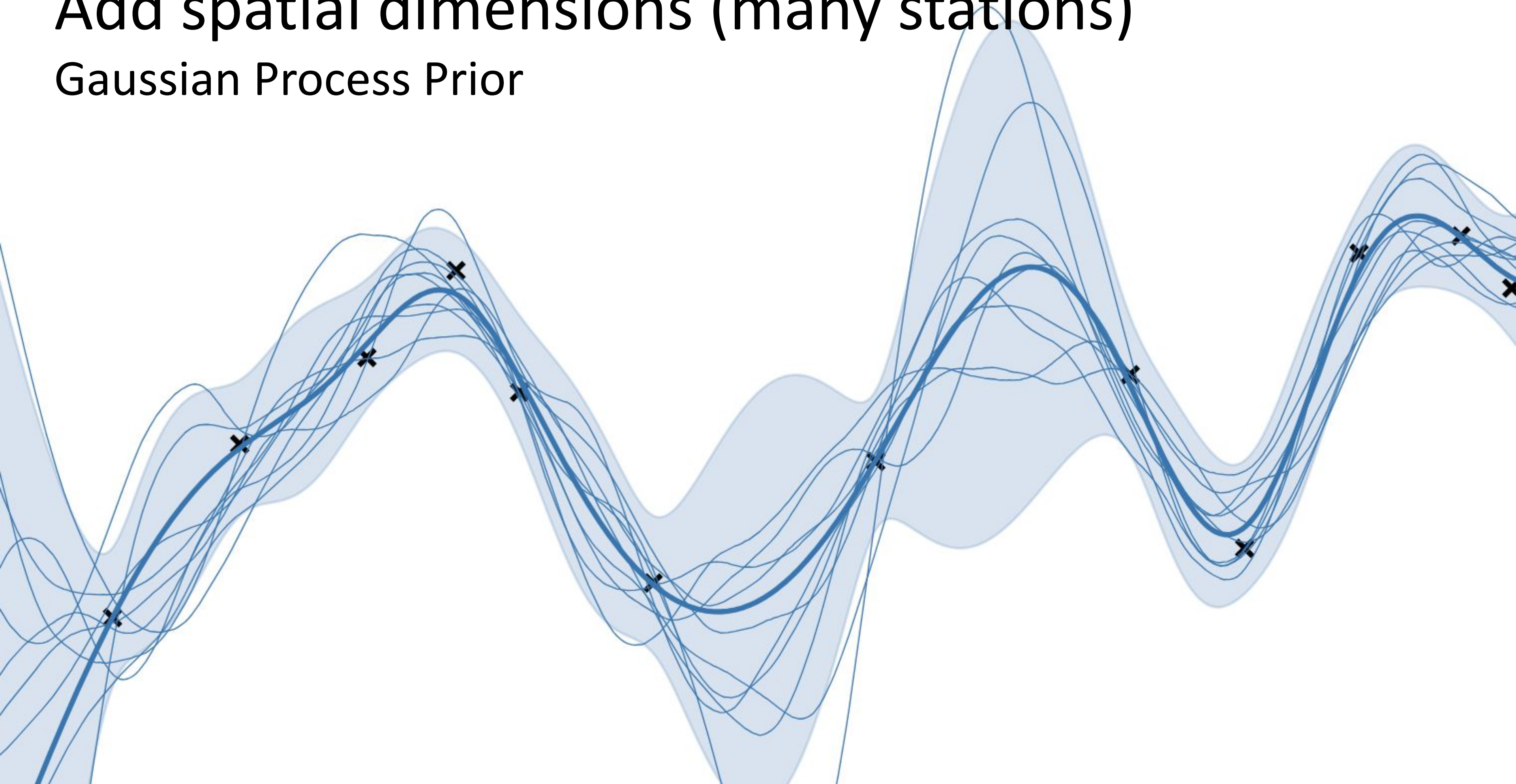
- For a **return period** (say every 100 years) we expect to exceed the **return level**
- After fitting the distribution in PyMC it's straightforward to compute statistics like return levels.
- The Newark record (8.4 inch) corresponds to a return period of 83 years.
- The advantage of using a Bayesian approach is that we can easily estimate the uncertainty as well




```python
with model:
    rp = pm.ConstantData("rp", return_periods)
    rl = pm.Deterministic("rl", μ - σ/ξ * (1 - (-np.log(1 - 1/rp)) ** (-ξ)))
    posterior_pred = pm.sample_posterior_predictive(trace,var_names=['rl'])
```

Add spatial dimensions (many stations)

Gaussian Process Prior



How can we improve the model? Using a Gaussian Process!

Add longer history?

- Often not available
- Might not be appropriate if the history is very different

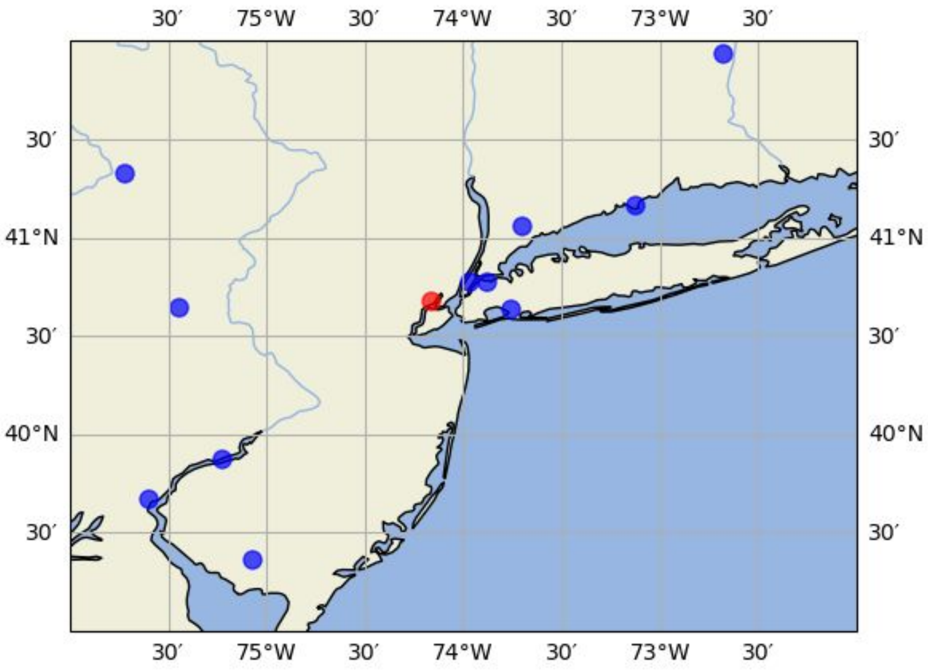
How can we improve the model? Using a Gaussian Process!

Add longer history?

- Often not available
- Might not be appropriate if the history is very different

Add data from other weather stations instead!

- We expect that measurements from nearby stations will be correlated (as function of distance)



How can we improve the model? Using a Gaussian Process!

Add longer history?

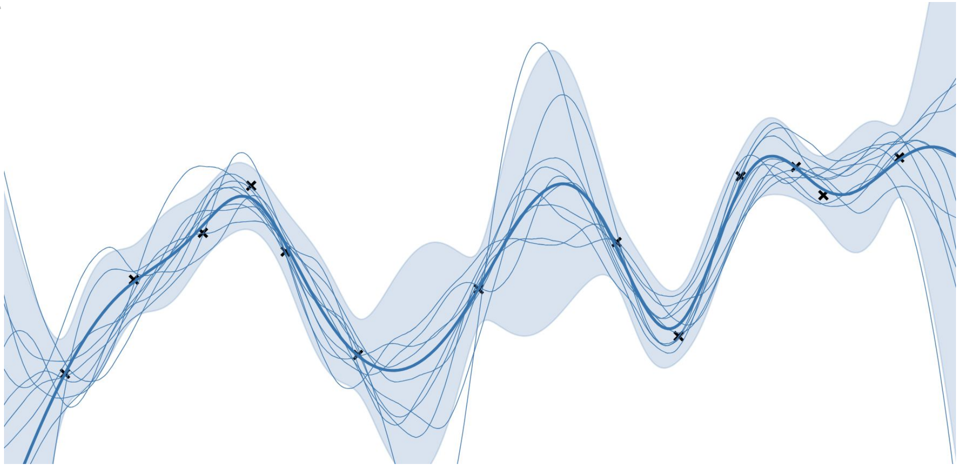
- Often not available
- Might not be appropriate if the history is very different

Add data from other weather stations instead!

- We expect that measurements from nearby stations will be correlated (as function of distance)

How to model this in a Bayesian way?

- “All” we need to do is to change the prior distribution
- The prior is a so-called Gaussian Process



Introduction to (Latent) Gaussian Processes

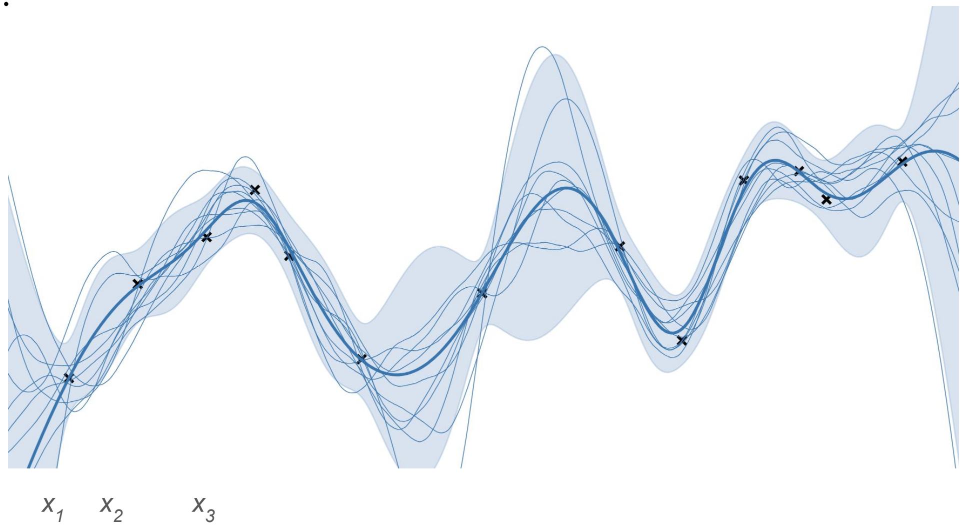
Instead of for each station finding independent prior distributions for the parameters $\mu_1, \mu_2, \mu_3, \ldots$

Find a prior distribution for a function $\mu(x)$ which can be evaluated at different locations

We want

- The mean of $\mu(x)$ to be smooth
- The variance of $\mu(x)$ to reduce when we add observations

A Gaussian Process achieves exactly this!



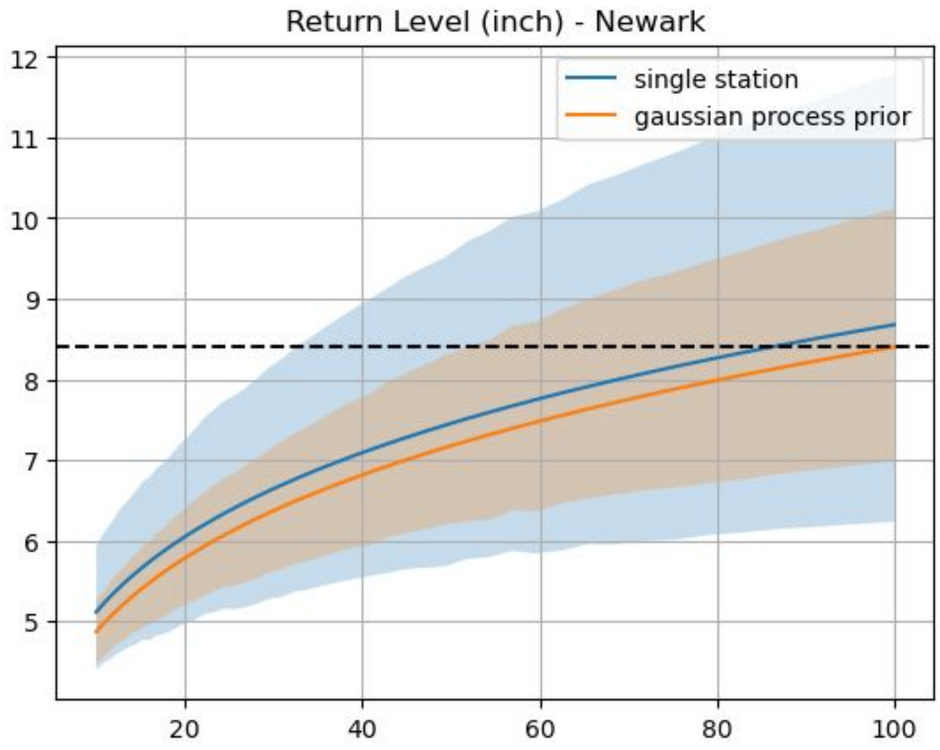
Bayesian Model with Gaussian Process Prior

```
with pm.Model(coords=coords) as gp_model:
    pt_idx = pm.ConstantData("station_idx", station_idx, dims="obs")
    pt_X_lonlat = pm.ConstantData("station_loc", X_lonlat, dims=("station","lonlat"))
    # gaussian process hyper parameters
    ℓ = pm.InverseGamma("ℓ", mu = 50.0, sigma = 50.0)
    η = pm.Gamma("η", mu=0.15, sigma=0.10, dims = "cov_params")
    # gaussian process prior for mu
    gp_μ = pm.gp.Latent(cov_func=η[0]**2 * Matern32Chordal(2, ℓ))
    μ_group = pm.Normal("μ_group", mu=3.0, sigma=1.0)
    μ = pm.Deterministic("μ", μ_group + gp_μ.prior("μ_gp", X=pt_X_lonlat), dims="station")
    # gaussian process prior for sigma
    gp_σ_log = pm.gp.Latent(cov_func=η[1]**2 * Matern32Chordal(2, ℓ))
    σ_log = gp_σ_log.prior("σ_log", X=pt_X_lonlat, dims="station")
    σ_log_group = pm.Normal("σ_log_group", mu=-1.0, sigma=2.0)
    σ = pm.Deterministic("σ", pm.math.exp(σ_log_group + σ_log),dims="station")
    # gaussian process prior for xi
    gp_ξ = pm.gp.Latent(cov_func=η[2]**2 * Matern32Chordal(2, ℓ))
    ξ_group =  pm.TruncatedNormal('ξ_group', mu=0.0, sigma=0.25, lower=-0.99, upper=0.99)
    ξ = pm.Deterministic("ξ", pm.math.tanh(ξ_group + gp_ξ.prior("ξ_gp", X=pt_X_lonlat)), dims="station")
    # likelihood for all observations
    gev = pmx.GenExtreme("gev", mu=μ[pt_idx], sigma=σ[pt_idx], xi=ξ[pt_idx], observed=y_, dims="obs")
```

The model is “slightly” more complicated but still “fits” on the screen.

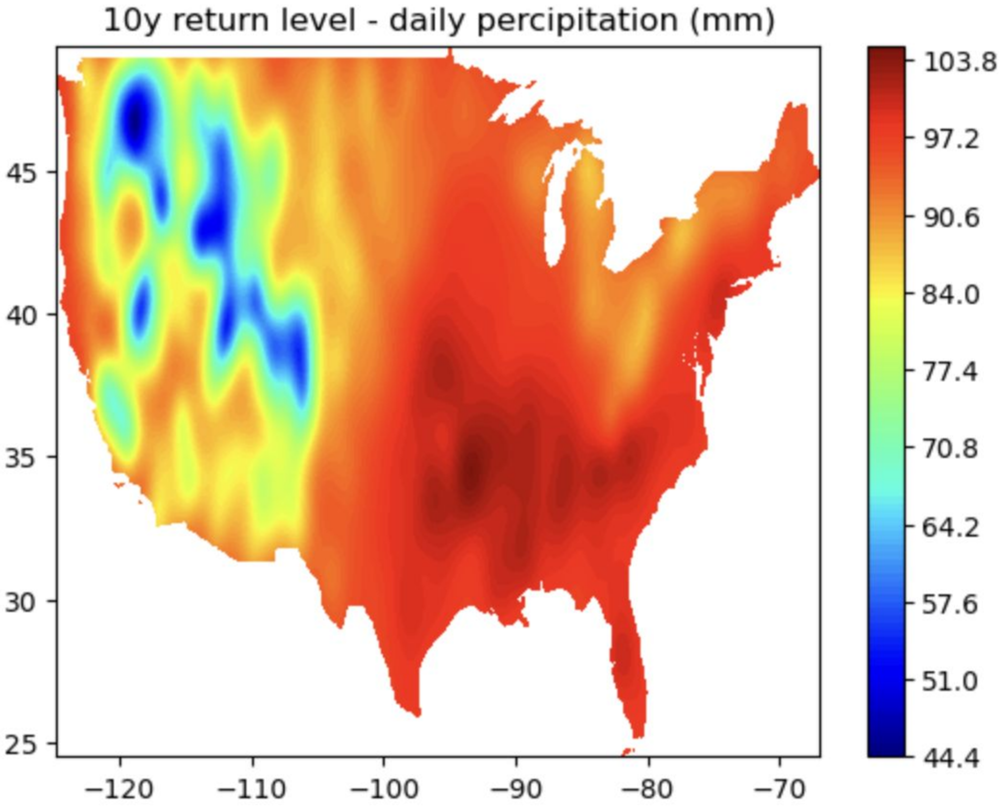
Results for Gaussian Process Prior

- Using 10 nearby stations to get a more accurate prediction for Newark
- Uncertainty almost twice as small!
- New estimate: Hurricane Ida was a once in a **100-year event**



Even more stations

3500 stations



Takeaways

- Modelling Extreme Events -> Extreme Value Theory
- Bayesian Modelling in python -> PyMC
- Modelling spatial problems -> Gaussian Processes

Thanks for attending!

For more, visit: https://github.com/jjmossel/extremeweather

- Full working code
- References on Extreme Weather modeling