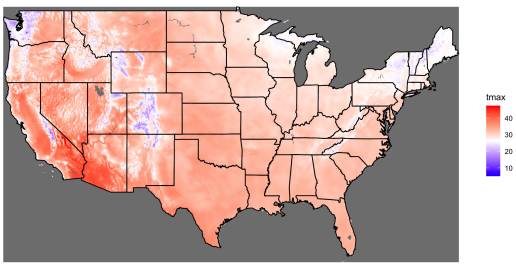
Scaling black-box inference to large spatial settings for extreme climate

Amanda Lenzi[a] and Emily Hector[b]

[a]The University of Edinburgh

[b]North Carolina State University

Extremal temperature in the U.S.A



Annual maxima temperature data on a 5 km by 5 km grid from NOAA's NClimGrid from 2023

- Modelling spatial datasets is **computationally challenging**, even with restrictive model assumptions and in moderate dimension

Simulation-based inference

$\mathrm{L}(\boldsymbol{\theta}_0; \mathbf{Y}) = f(\mathbf{Y}; \boldsymbol{\theta}_0)$ cannot be evaluated as a function of $\boldsymbol{\theta}_0$; however, one can *simulate* $\mathbf{y} \sim f(\mathbf{Y}; \boldsymbol{\theta}_0)$ for any fixed $\boldsymbol{\theta}_0$

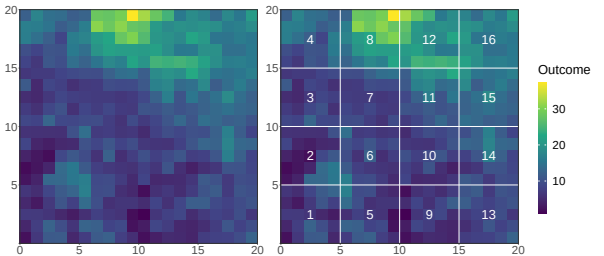
- Approximate Bayesian computation (ABC)
- Simulation-based inference (SBI) / likelihood-free inference (LFI) using neural networks

Key assumption: It is easy to simulate from $f(\mathbf{Y}; \boldsymbol{\theta}_0)$

This talk: When this assumption does not hold (e.g., models for spatial extremes)

Local models

Train a neural network on data simulated on **small blocks** of the domain



Computational efficiency:

- Scalable to very high dimensions even when fast simulation from the model is not possible (only simulate on the small spatial domain and can be done in parallel)
- Once the neural network has been trained, estimation is independent of the actual data size (fully amortised)

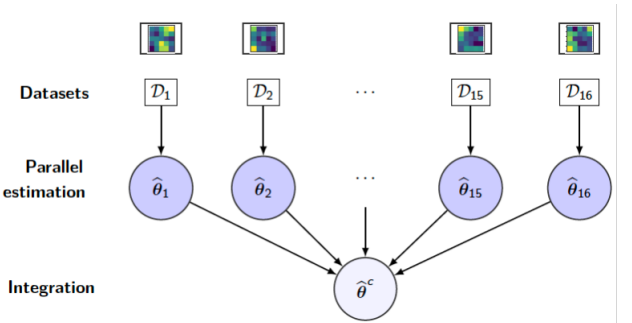
Local model fitting using neural networks

- partition the spatial domain $\mathcal{D}$ into K disjoint regions $\mathcal{D}_1, \ldots, \mathcal{D}_K$ such that $\cup_{k=1}^{K} \mathcal{D}_k = \mathcal{D}$ and denote by d_k the number of locations in $\mathcal{D}_k$
- Fix a set of candidate parameter values $\{\boldsymbol{\theta}_t\}_{t=1}^{T}$
- For each $\boldsymbol{\theta}_t$, simulate data $Y_t(\mathcal{D}_k)$ independently on each block to generate training data $\{\boldsymbol{\theta}_t, Y_t(\mathcal{D}_k)\}$
- Train a neural network a_k using $Y_t(\mathcal{D}_k)$ as the input and $\boldsymbol{\theta}_t$ as the output for each $k = 1, \ldots, K$ (in parallel across K computing nodes)
- Use the trained neural networks to estimate the value of $\boldsymbol{\theta}_0$ for observed data $Y(\mathcal{D}_k)$, $k = 1, \ldots, K$, yielding K estimates $\widehat{\boldsymbol{\theta}}_k = a_k\{\widehat{\boldsymbol{A}}_k; Y(\mathcal{D}_k)\}$

Computationally amortised estimates of $\boldsymbol{\theta}_0$ from each of the K blocks

Neural network integration

Need to combine estimates from the K small blocks



- Use the mean estimator $\widehat{\boldsymbol{\theta}}_c = K^{-1} \sum_{k=1}^{K} \widehat{\boldsymbol{\theta}}_k$?
- Issue: Its variance may be inflated by the block estimates $\widehat{\boldsymbol{\theta}}_k$

$$\mathrm{Var}(\widehat{\boldsymbol{\theta}}_m) = K^{-2}(\textstyle\sum_{k=1}^{K} \mathrm{Var}(\widehat{\boldsymbol{\theta}}_k) + \sum_{k,k'=1,\ k\neq k'}^{K} \mathrm{Cov}(\widehat{\boldsymbol{\theta}}_k, \widehat{\boldsymbol{\theta}}_{k'}))$$

Challenge in developing a divide-and-conquer approach

How to combine the dependent parameter estimates from each block into a global estimate

- Need calibrated uncertainty quantification (nominal quantiles match the estimated quintiles)
- Most (pseudo)likelihood-based divide-and-conquer methods focus on prediction due to this difficulty
- Account for the dependence $\text{Cov}(\widehat{\boldsymbol{\theta}}_k, \widehat{\boldsymbol{\theta}}_{k'})$ between $\widehat{\boldsymbol{\theta}}_k$ inherited from the dependence between $\mathcal{D}_k$

Recent work on divide-and-conquer approach

Hector and Reich (2023); Hector et al. (2024) designed a divide-and-conquer approach for max-stable and Gaussian processes

- However, it remains computationally burdensome when the number of spatial locations is large (rely on composite likelihood and full likelihood estimation, respectively, within each block)

- Requires repeated, independent observations of the spatial domain to estimate the dependence between estimators from each block

- We propose an improved weighted estimator that accounts for this cross-covariance and thereby minimizes the variance of the resulting estimator

Our proposed divide-and-conquer approach

- Replace the (pseudo)likelihood evaluation in each block with the black-box parameter estimation
- Use the covariance between the B bootstrap replicates $\widehat{\boldsymbol{\theta}}_{kb}$ to estimate the covariance between $\widehat{\boldsymbol{\theta}}_k$
- Propose selecting a neural network from multiple trained networks to minimize the influence of the *amortisation gap* on the downstream inference
- We illustrate this empirical strategy both without and with the divide-and-conquer framework

Neural network integration

The major difference between our proposal and previous work is that the outcome process is **only observed once**, and bootstrapping is used to quantify the dependence between $\widehat{\boldsymbol{\theta}}_k$

- We sample independent replicates $Y_b(\mathcal{D}_k)$, $b = 1, \ldots, B$ from $f\{y(\mathcal{D}_k); \widehat{\boldsymbol{\theta}}_m\}$ and generate bootstrap replicates in each block of the partitioned spatial domain using $\widehat{\boldsymbol{\theta}}_{kb} = a\{\widehat{\boldsymbol{A}}_k; Y_b(\mathcal{D}_k)\}$

- The bootstrap replicates are only *conditionally* independent across $k = 1, \ldots, K$ given $\widehat{\boldsymbol{A}}_k, Y_b(\mathcal{D}_k)$: the distribution from which $Y_b(\mathcal{D}_k)$ are sampled depends on $\widehat{\boldsymbol{\theta}}_m$, which is shared across the blocks and whose variance captures dependence between $\widehat{\boldsymbol{\theta}}_k$

- The bootstrap replicates $\widehat{\boldsymbol{\theta}}_{kb}$ are *marginally* dependent with a dependence structure that captures the dependence between block estimators $\widehat{\boldsymbol{\theta}}_k$

Neural network integration

We define a one-step meta-estimator that is asymptotically equivalent to the optimal estimator as $d_k \to \infty$

$$\widehat{\boldsymbol{\theta}}_c = \left\{ \sum_{k,k'=1}^{K} (\widehat{\boldsymbol{W}}_{opt})_{k,k'} \right\}^{-1} \sum_{k,k'=1}^{K} (\widehat{\boldsymbol{W}}_{opt})_{k,k'} \widehat{\boldsymbol{\theta}}_{k'},$$

where $\boldsymbol{W}_{opt}^{-1}$ is a bootstrap estimator of $\boldsymbol{v}(\boldsymbol{\theta}_0) = \mathsf{Var}(\widehat{\boldsymbol{\theta}}_1^\top, \ldots, \widehat{\boldsymbol{\theta}}_K^\top)$

- The estimator $\widehat{\boldsymbol{\theta}}_c$ can be computed in an accelerated distributed data network
- *Amortised estimation*: no new training needs to occur when new data $Y(\mathcal{D}')$ are collected so long as $\mathcal{D}'$ can be partitioned into blocks of sizes in $\{d_1, \ldots, d_K\}$
- B, the number of bootstrap samples can be made arbitrarily large, yielding an estimator $\boldsymbol{W}_{opt}(\boldsymbol{\theta})$ that can be made arbitrarily precise for $\boldsymbol{v}(\boldsymbol{\theta}_0)$

Can the uncertainty of $\widehat{\theta}_c$ be quantified computationally efficiently as well?

We assume that $(\widehat{\boldsymbol{\theta}}_c - \boldsymbol{\theta}_0)$ is approximately Gaussian, with mean $\mathbf{0}$ and variance $\boldsymbol{j}^{-1}(\boldsymbol{\theta}_0) = \{\sum_{k,k'=1}^{K} \boldsymbol{v}^{-1}(\boldsymbol{\theta}_0)\}^{-1}$ and estimate $\boldsymbol{j}(\boldsymbol{\theta}_0)$ with

$$\widehat{\boldsymbol{J}}_{opt} = \textstyle\sum_{k,k'=1}^{K} (\widehat{\boldsymbol{W}}_{opt})_{k,k'},$$

and construct large sample confidence intervals for

$$\widehat{\boldsymbol{\theta}}_c \pm z_{\alpha/2} \Big[\mathsf{diag} \Big\{ \textstyle\sum_{k,k'=1}^{K} (\widehat{\boldsymbol{W}})_{k,k'} \Big\}^{-1} \Big]^{1/2}$$

- Holds if $\widehat{\boldsymbol{\theta}}_k - \widehat{\boldsymbol{\theta}}_k^{\star} = o_p(1)$ is a Gaussian process covariance model, where $\widehat{\boldsymbol{\theta}}_k^{\star}$ is the maximum likelihood estimator based on $Y(\mathcal{D}_k)$
- Empirical evidence shows that the distribution of $\widehat{\boldsymbol{\theta}}_c$ is Gaussian and centered at $\boldsymbol{\theta}_0$ with variance estimated by $\widehat{\boldsymbol{J}}_{opt}^{-1}$ when $a_k(\boldsymbol{A}; \cdot)$ is sufficiently complex, and d_k, B are sufficiently large

Simulations with Gaussian processes

Aim: Inference on $\boldsymbol{\theta}_0 = \{\log(\tau_0^2), \log(\phi_0^2)\}$ when d is large for a mean-zero Gaussian process with covariance function
$C\{y(j), y(j')\} = \tau_0^2 \exp(-\|j - j'\|_2/\phi_0^2)$

Spatial domain: Square gridded spatial domain of dimension $d^{1/2} \times d^{1/2}$. Fix $\tau_0^2 = 1$ and $\phi_0^2 = 3$ and vary $d \in \{60^2, 90^2, 120^2\}$

Training and validation data: Sequences of size 70^2 from $\log(\phi_0^2) - 0.5$ to $\log(\phi_0^2) + 0.5$ and $\log(\tau_0^2) - 0.5$ to $\log(\tau_0^2) + 0.5$

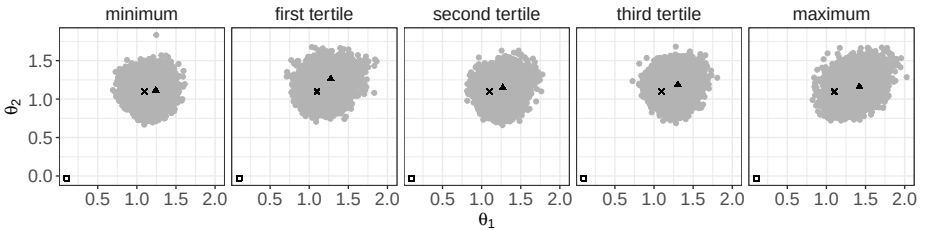
Local fit: Partition $\mathcal{D}$ into square blocks of $d_k = 30^2$ locations each ($d = 60^2, 90^2, 120^2$ give $K = 4, 9, 16$ blocks)

Local estimation: Plug in $Y(\mathcal{D}_k)$ and estimate $\widehat{\boldsymbol{\theta}}_k$ in each block k

Integration: $\widehat{\boldsymbol{\theta}}_c$ is computed using $B = 5{,}000$ bootstrap replicates

Local fit: What is the influence on inference of the stochastic nature of gradient descent?

A CNN is trained 500 times with different seeds (all with the same architecture and training and validation data)



Neural network estimates (triangles), maximum likelihood estimates (squares), true values (crosses) and bootstrap replicates (grey dots) of $\boldsymbol{\theta}_0$ based on the five trained neural networks with minimum, first, second, third tertile and maximum average validation loss (AVB).

We select the neural network with the smallest minimized validation loss

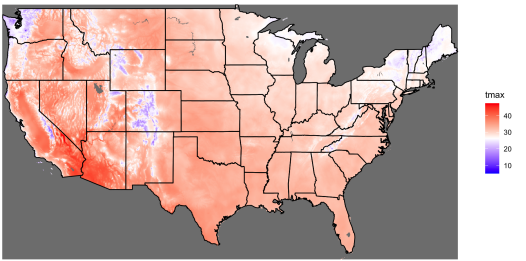
Simulations with Gaussian processes

d	parameter	RMSE$\times 100$	ESE$\times 100$	ASE$\times 100$	CP (%)
60^2	ϕ^2	7.81 (9.12,9.45)	7.80 (9.11,9.43)	7.76	93.4
	τ^2	7.55 (8.58,8.69)	7.54 (8.57,8.55)	7.09	92.4
90^2	ϕ^2	5.70	5.67	5.19	92.6
	τ^2	5.23	5.20	4.74	92.2
120^2	ϕ^2	4.12	4.05	3.89	94.2
	τ^2	3.77	3.71	3.56	94.2

Metrics for the distributed neural estimator with MLE and Vecchia in parentheses

- Similar RMSE and ESE (negligible bias)
- 95% confidence intervals reach their nominal levels (Gaussian approximation works well for inference)
- MLE and Vecchia have very similar RMSE and ASE
- The distributed approach is much faster than competitors (15.5 sec versus 960 and 1690 sec)

Simulations with Brown-Resnick processes



- Brown-Resnick processes are a type of max-stable processes, which are used for studying extreme events in space
- These processes are well-known for having full likelihoods that are computationally intractable
- **Challenge:** Classical or Bayesian inference effectively impossible, even in small dimension

Simulations with Brown-Resnick processes

Aim: Inference on $\boldsymbol{\theta}_0 = \{\log(\lambda_0), \text{logit}(\nu_0)\}$ from a zero-mean Gaussian process with semivariogram $\gamma(\mathbf{h}) = (\|\mathbf{h}\|/\lambda)^\nu$, $\lambda > 0$ and $\nu > 0$

Spatial domain: Square gridded spatial domain of dimension $d^{1/2} \times d^{1/2}$. Fix $\lambda_0 = 1$ and $\nu_0 = 1$ and vary $d \in \{20^2, 30^2, 40^2, 50^2\}$

Training and validation data: Sequences of size 70^2 from $\log(\lambda_0) - 0.5$ to $\log(\lambda_0) + 0.5$ and $\text{logit}(\nu_0) - 0.5$ to $\text{logit}(\nu_0) + 0.5$

Local fit: Partition $\mathcal{D}$ into square blocks of $d_k = 30^2$ locations each ($d = 60^2, 90^2, 120^2$ give $K = 4, 9, 16$ blocks)

Local estimation: Plug in $Y(\mathcal{D}_k)$ and estimate $\widehat{\boldsymbol{\theta}}_k$ in each block k

Integration: $\widehat{\boldsymbol{\theta}}_c$ is computed using $B = 5{,}000$ bootstrap replicates

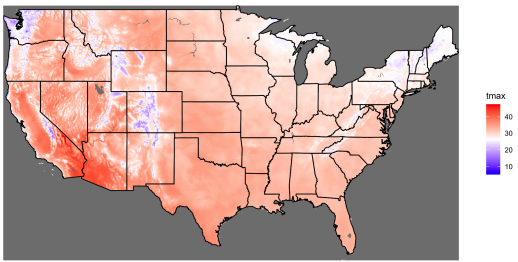
Simulations with Brown Resnick processes

d	Parameter	RMSE$\times 100$	ESE$\times 100$	ASE$\times 100$	CP (%)
20^2	θ_1	10.2 (83.2)	10.1 (80.8)	10.10	94.40
	θ_2	13.6 (109)	13.5 (106)	12.70	94.40
30^2	θ_1	6.5 (115)	6.43 (112)	6.75	95.20
	θ_2	8.64 (169)	8.65 (160)	8.46	94.60
40^2	θ_1	5.3 (91.1)	5.26 (90.1)	5.06	94.30
	θ_2	6.57 (166)	6.55 (156)	6.34	94.30
50^2	θ_1	4.23 (88.6)	4.17 (88.7)	4.05	95.00
	θ_2	5.19 (206)	5.19 (180)	5.07	93.80

Metrics for the distributed neural estimator with metrics of pairwise likelihood approximation in parentheses

- Similar RMSE and ESE
- RMSE, ESE and ASE values tend to decrease as d increases
- Low bias independent of the window size
- The CP is below the nominal coverage for larger values of range (estimated CI's might be too narrow or slightly biased when the spatial dependence is stronger)

United States temperature dataset

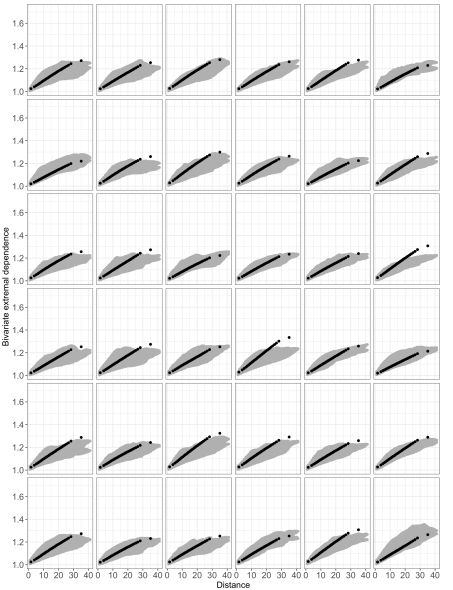


Annual maxima temperature data on a 5 km by 5 km grid from NOAA's NClimGrid from 2023

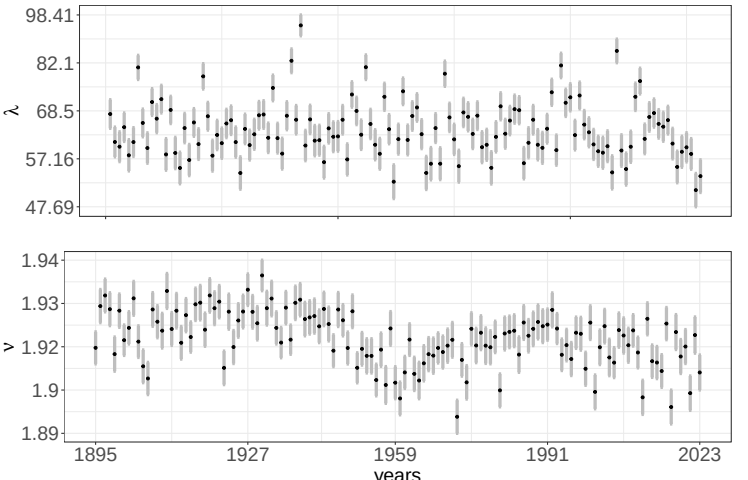
- Monthly values in a 5×5 latitude/longitude grid for the Continental U.S. from 01 January 1895 to the present
- We compute yearly temperature maxima for the 129 years for a spatial region of size $d = 180^2 = 32{,}400$
- We fit a GEV distribution for each spatial location separately, which we then use to transform annual maxima to a common unit Fréchet scale

Empirical bivariate extremal coefficients

- Empirical values from the NOAA data (grey dots) computed on blocks of size $d = 30^2$ (grey dots) and integrated neural network-based approach (black dots)
- Highly spatially dependent, with maxima that are very correlated even at larger distances
- The model estimates are close to the empirical estimates at all distances



Estimated 95% confidence intervals



95% CIs of the combined weighted estimator $\widehat{\boldsymbol{\theta}}_c$ of the range parameter (top) and smoothness (bottom). Black dots indicate the combined weighted estimator

Summary

Local models combined with neural network are quick and easy to fit even when simulation from the model is not straightforward

Utilizing bootstrapping samples from the neural network estimator in the divide-and-conquer enables inference in very large areas across varying number of spatial domains

Much of the potential of neural amortized inference is yet to be realized

... a lot of work (and simulations) still needed