SpatialGEV: Fast Bayesian inference for spatial extreme value models in R

December 6, 2021

Summary

Extreme weather phenomena such as floods and hurricanes are of great concern due to their potential to cause extensive damage. To develop more reliable damage prevention protocols, statistical models are often used to infer the chance of observing an extreme weather event at a given location (Coles and Casson 1998; Cooley, Nychka, and Naveau 2007; Sang and Gelfand 2010). Here we present `SpatialGEV`, an R package providing a fast and convenient toolset for analyzing spatial extreme values using a hierarchical Bayesian modeling framework. In this framework, the marginal behavior of the extremes is given by a generalized extreme value (GEV) distribution, whereas the spatial dependence between locations is captured by modeling the GEV parameters as spatially varying random effects following a Gaussian process (GP). Users are provided with a streamlined way to build and fit various GEV-GP models in R, which are compiled in C++ under the hood. For downstream analyses, the package offers methods for Bayesian parameter estimation and forecasting of extreme events.

Statement of need

The GEV-GP model has important applications in meteorological studies. For example, let $y = y(\boldsymbol{x})$ denote the amount of rainfall at a spatial location $\boldsymbol{x}$. To forecast extreme rainfalls, it is often of interest for meteorologists to estimate the $p\%$ rainfall return value $z_p(\boldsymbol{x})$, which is the value above which precipitation levels at location $\boldsymbol{x}$ occur with probability p, i.e.,

$$\Pr\big(y(\boldsymbol{x}) > z_p(\boldsymbol{x})\big) = 1 - F_{y|\boldsymbol{x}}\big(z_p(\boldsymbol{x})\big) = p, \tag{1}$$

where the CDF is that of the GEV distribution specific to location $\boldsymbol{x}$. When p is chosen to be a small value, $z_p(\boldsymbol{x})$ indicates how extreme the precipitation level might be at location $\boldsymbol{x}$.

In a Bayesian context, the posterior distribution $p(z_p(\boldsymbol{x}) \mid \boldsymbol{y})$, where $\boldsymbol{y} = (\boldsymbol{y}(\boldsymbol{x}_1), \ldots, \boldsymbol{y}(\boldsymbol{x}_n))$ represents rainfall measurements at n different locations, is very useful for forecasting extreme weather events. Traditionally, Markov Chain Monte Carlo (MCMC) methods are used to sample from the posterior distribution of the GEV model (e.g., Cooley, Nychka, and Naveau 2007; Schliep et al. 2010; Dyrrdal et al. 2015). However, this can be extremely computationally intensive when the number of locations is large. The `SpatialGEV` package implements Bayesian inference based on the Laplace approximation as an alternative to MCMC, making large-scale spatial analyses orders of magnitude faster while achieving roughly the same accuracy as MCMC. The Laplace approximation is carried out using the R/C++ package `TMB` (Kristensen et al. 2016). Details of the inference method can be found in Chen, Ramezan, and Lysy (2021).

Statement of field

The R pacakge `SpatialExtremes` (Ribatet, Singleton, and R Core team 2020) is one of the most popular software for fitting spatial extreme value models, which employs an efficient Gibbs sampler. The Stan programming language and its R interface `RStan` (Stan Development Team 2020) provides off-the-shelf implementations for Hamiltonian Monte Carlo and its variants (Neal 2011; Hoffman and Gelman 2014), which are considered state-of-the-art MCMC algorithms and often used for fitting hierarchical spatial models. A well-known alternative to MCMC is the `R-INLA` package (Lindgren and Rue 2015) which implements the integrated nested Laplace approximation (INLA) approach. As an extension of the Laplace approximation, INLA is often considerably more accurate. However, the INLA methodology is inapplicable to GEV-GP models in which both location and scale parameters are modeled as random effects. Chen, Ramezan, and Lysy (2021) compares the speed and accuracy of the Laplace method implemented in `SpatialGEV` to `RStan` and `R-INLA`. It is found that both `SpatialGEV` and `R-INLA` are two orders of magnitude faster than `RStan`. Moreover, `R-INLA` with just one spatially varying parameter is found to forego a substantial amount of modeling flexibility, which can lead to considerable bias in estimating the return levels in (1).

Example

Model fitting

The main functions of the `SpatialGEV` package are `spatialGEV_fit()`, `spatialGEV_sample()`, and `spatialGEV_predict()`. This example shows how to apply these functions to analyze a simulated dataset using the GEV-GP model. The spatial domain is a 20×20 regular lattice on $[0, 10] \times [0, 10] \subset \mathbb{R}^2$, such that there are $n = 400$ locations in total. The GEV location parameter $a(\boldsymbol{x})$ and the log-transformed scale parameter $\log(b(\boldsymbol{x}))$ are generated from the unimodal and bimodal functions depicted in Figure 1. The GEV shape parameter s is set to be $\exp(-2)$, constant across locations. One observation per location is simulated from the GEV distribution conditional on the GEV parameters $(a(\boldsymbol{x}), b(\boldsymbol{x}), s)$. The simulated data is provided by the package as a data frame called `simulatedData`.

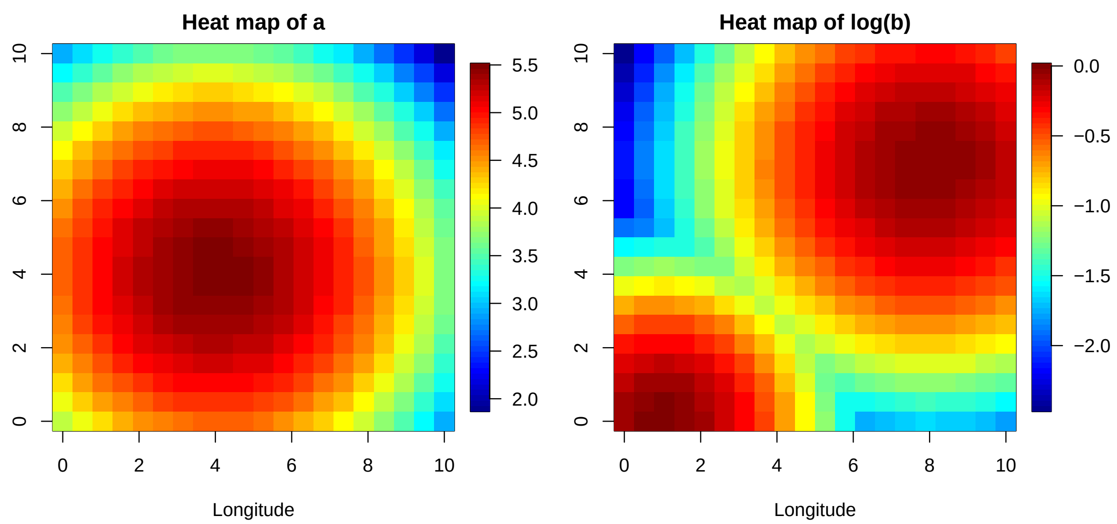

Figure 1: The simulated GEV location parameters $a(\boldsymbol{x}_i)$ and log-transformed scale parameters $\log(b(\boldsymbol{x}_i))$ plotted on regular lattices.

The GEV-GP model is fitted by calling `spatialGEV_fit()`. By specifying `random="ab"`, both the location parameter a and scale parameter b are considered spatial random effects. Initial parameter values are passed to `init_param`, where `log_sigma_{a/b}` and `log_ell_{a/b}` are hyperparameters in the GP kernel functions for the GEV parameter spatial processes. The argument `reparam_s="positive"` means we constrain the shape parameter to be positive, i.e., its estimation is done on the log scale. The posterior mean estimates of the spatial random effects can be accessed from `mod_fit$report$par.random`, whereas the fixed effects can be obtained from `mod_fit$report$par.fixed`.

```
set.seed(123)                       # set seed for reproducible results
require(SpatialGEV)                 # load package
locs <- cbind(simulatedData$lon, simulatedData$lat)  # location coordinates
a <- simulatedData$a                # true GEV location parameters
```

```
logb <- simulatedData$logb         # true GEV (log) scale parameters
logs <- -2                         # true GEV (log) shape parameter
y <- simulatedData$y               # simulated observations
mod_fit <- spatialGEV_fit(y = y, X = locs, random = "ab",
                          init_param = list(a = a, log_b = logb, s = logs,
                                            log_sigma_a = 1, log_ell_a = 5,
                                            log_sigma_b = 1, log_ell_b = 5),
                          reparam_s = "positive", silent = TRUE)
```

Sampling from the joint posterior

Now, we show how to sample 5000 times from the joint posterior distribution of the GEV parameters using the function `spatialGEV_sample()`. Only three arguments need to be passed to this function: `model` takes in the list output by `spatialGEV_fit()`, `n_draw` is the number of samples to draw from the posterior distribution, and `observation` indicates whether to draw from the posterior predictive distribution of the data at the observed locations. The samples are used to calculate the posterior mean estimates of the 10% return level $z_{10}(\boldsymbol{x})$ at each location, which are plotted against their true values in Figure 2.

```
require(evd)                    # evd provides the GEV distribution
p_val <- 0.1                    # 10% return level

# Sample from the joint posterior distribution of all model parameters
n_draw <- 5000                  # Number of samples from the posterior
all_draws <- spatialGEV_sample(model = mod_fit, n_draw = n_draw)
all_draws <- all_draws$parameter_draws

# Calculate the posterior mean of the return levels from the samples
n_loc <- length(y)              # number of locations
q_means <- rep(NA, n_loc)       # posterior means of the return levels
s_vec <- exp(all_draws[, "s"])  # posterior samples of s
for (i in 1:n_loc){
  a_vec <- all_draws[ , paste0("a", i)]          # posterior samples of a(x_i)
  b_vec <- exp(all_draws[ , paste0("log_b", i)]) # posterior samples of b(x_i)
  q_means[i] <- mean(apply(cbind(a_vec, b_vec, s_vec), 1,
                         function(x) evd::qgev(1-p_val, x[1], x[2], x[3])))
}

# Calculate the true return values at different locations
q_true <- apply(cbind(a, exp(logb)), 1,
                function(x) evd::qgev(1-p_val, x[1], x[2], shape=exp(logs)))
# Plot q_means against q_true
```

Prediction at new locations

Next, we show how to predict the values of the extreme event at test locations. First, we divide the simulated dataset into training and test sets, and fit the model to the training dataset. We can simulate from the posterior predictive distribution of observations at the test locations using the `spatialGEV_predict()` function, which requires the fitted model to the training data passed to `model`, a matrix of the coordinates of the test locations passed to `X_new`, a matrix of the coordinates of the observed locations passed to `X_obs`, and the number of simulation draws passed to `n_draw`. Figure 3 plots the 95% posterior predictive intervals at the 20 test locations along with the true observed values as superimposed circles.

```
n_test <- 20                       # number of test locations
test_ind <- sample(1:400, n_test)  # indices of the test locations
locs_test <- locs[test_ind,]       # coordinates of the test locations
```

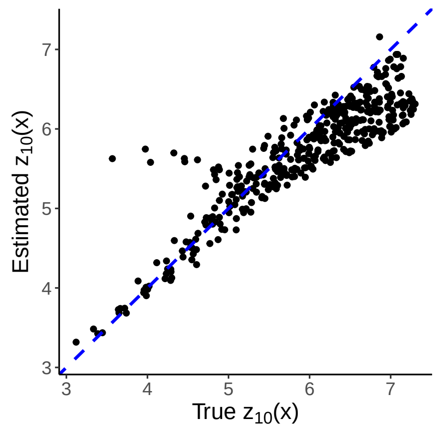

Figure 2: Posterior mean estimates of the 10% return level $z_{10}(\boldsymbol{x})$ plotted against the true values at different locations.

```
y_test <- y[test_ind]                  # observations at the test locations
locs_train <- locs[-test_ind,]         # coordinates of the training locations
y_train <- y[-test_ind]                # observations at the training locations

# Fit the GEV-GP model to the training set
train_fit <- spatialGEV_fit(y = y_train, X = locs_train, random = "ab",
                           init_param = list(a = a[-test_ind],
                                             log_b = logb[-test_ind],
                                             s = logs,
                                             log_sigma_a = 1, log_ell_a = 5,
                                             log_sigma_b = 1, log_ell_b = 5),
                           reparam_s = "positive", silent = TRUE)

# Make predictions at the test locations
pred <- spatialGEV_predict(model = train_fit, X_new = locs_test,
                            X_obs = locs_train, n_draw = 5000)
# Plot 95% posterior PI and the true observations
```

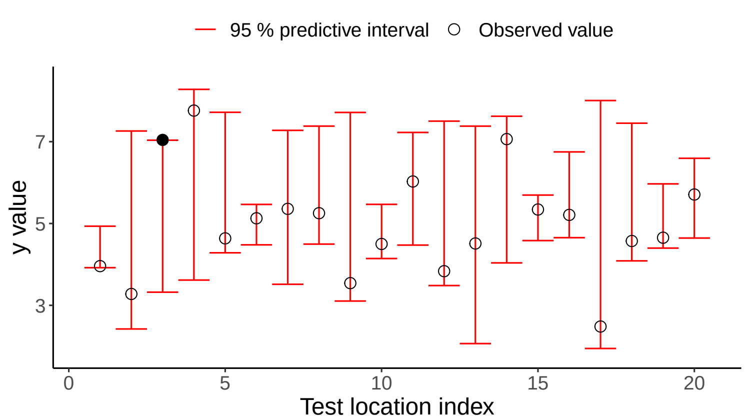

Figure 3: 95% posterior predictive intervals (PI) at test locations. Each circle corresponds to the true observation at that test location, with hollow ones indicating that they are inside the 95% PI, and solid ones indicating that they are outside of the 95% PI.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada, grant numbers RGPIN-2018-04376 (Ramezan), DGECR-2018-00349 (Ramezan) and RGPIN-2020-04364 (Lysy).

References

Chen, M., R. Ramezan, and M. Lysy. 2021. "Fast Approximate Inference for Spatial Extreme Value Models." http://arxiv.org/abs/2110.07051.

Coles, S. G., and E. Casson. 1998. "Extreme Value Modelling of Hurricane Wind Speeds." *Structural Safety* 20: 283–96.

Cooley, D., D. Nychka, and P. Naveau. 2007. "Bayesian Spatial Modeling of Extreme Precipitation Return Levels." *Journal of the American Statistical Association* 102: 824–40.

Dyrrdal, A. V., A. Lenkoski, T. L. Thorarinsdottir, and F. Stordal. 2015. "Bayesian Hierarchical Modeling of Extreme Hourly Precipitation in Norway." *Environmetrics* 26: 89–106.

Hoffman, M. D., and A. Gelman. 2014. "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo." *Journal of Machine Learning Research* 15: 1593–623.

Kristensen, K., A. Nielsen, C. W. Berg, H. Skaug, and B. M Bell. 2016. "TMB: Automatic Differentiation and Laplace Approximation." *Journal of Statistical Software* 70 (5): 1–21.

Lindgren, F. K., and H. Rue. 2015. "Bayesian Spatial Modelling with R-INLA." *Journal of Statistical Software* 63: 1–25.

Neal, R. M. 2011. "MCMC Using Hamiltonian Dynamics." In *The Handbook of Markov Chain Monte Carlo*. Chapman & Hall / CRC Press.

Ribatet, M., R. Singleton, and R Core team. 2020. "SpatialExtremes: Modelling Spatial Extremes." https://CRAN.R-project.org/package=SpatialExtremes.

Sang, H., and A. E. Gelfand. 2010. "Continuous Spatial Process Models for Spatial Extreme Values." *Journal of Agricultural, Biological, and Environmental Statistics* 15: 49–56.

Schliep, E. M., D. Cooley, S. R. Sain, and J. A. Hoeting. 2010. "A Comparison Study of Extreme Precipitation from Six Different Regional Climate Models via Spatial Hierarchical Modeling." *Extremes* 13: 219–39.

Stan Development Team. 2020. "RStan: The R Interface to Stan." http://mc-stan.org/.