A modeler's guide to extreme value software – Supplementary material

Léo R. Belzile, Christophe Dutang, Paul J. Northrop, Thomas Opitz

Compiled July 4, 2023

1 Likelihood inference for univariate extremes

1.1 Density and distribution function checks

We performed some sanity checks for various maximum likelihood estimation routines and parametric model implementations. Specifically, we verified that density functions are non-negative and evaluate to zero outside of the domain of the distribution, and that distribution functions are non-decreasing and map to the unit interval.

The generalized Pareto distribution has lower bound at the location parameter u and is bounded above at $u - \sigma/\xi$ whenever $\xi < 0$. Many software implementations forgo the location parameter, since for modelling large quantiles of a random variable Y above threshold u, it suffices to look at threshold exceedances $Y - u > 0$. No threshold exceedance should be exactly equal to zero so the value of the density at that point is immaterial, even if it should be set to zero in practice.

Certain packages, listed in Table 1.1 and Table 1.2, have incorrect implementations of density and distribution functions.

1.2 Optimization routines

We compared the maximum likelihood estimates returned by default estimation procedures for different packages for simulated data, checking that the value returned is a global optimum and the gradient is approximately zero whenever $\widehat{\xi} > -1$.

Léo R. Belzile
Département de sciences de la décision, HEC Montréal, Canada, E-mail: leo.belzile@hec.ca

Christophe Dutang
Université Grenoble Alpes, CNRS, Grenoble INP, LJK, Grenoble, France

Paul J. Northrop
Department of Statistical Science, University College London, United Kingdom

Thomas Opitz
Biostatistics and Spatial Processes, INRAE, Avignon, France

Table 1.1: Evaluation of generalized Pareto model density and distribution functions.

package	location	density	distribution function
eva	yes	correct	correct
evd	yes	incorrect for $x = u$	correct
evir	yes	incorrect for $x < u$	incorrect outside support
extraDistr	yes	incorrect for $x = u$	correct
extRemes	yes	incorrect for $x = u$	correct
fExtremes	yes	incorrect for $x = u$	correct
lmom	yes		incorrect outside support
lmomco	yes	correct	incorrect outside support
mev	yes	correct	correct
POT	yes	incorrect for $x = u$	correct
QRM	no	incorrect for $x = u$	correct
qrmtools	no	correct	correct
ReIns	yes	correct	correct
Renext	yes	incorrect for $x = u$	correct
revdbayes	yes	correct	correct
SpatialExtremes	yes	incorrect for $x = u$	correct
tea	yes	correct	correct
texmex	yes	correct	correct
TLMoments	yes	correct	correct

Table 1.2: Evaluation of generalized extreme value density and distribution functions.

package	density	distribution function
EnvStats	correct	correct
evd	correct	correct
evir		
extraDistr	correct	correct
ExtremalDep		
extRemes	correct	correct
fExtremes	correct	correct
lmomco	incorrect for $x < \mu$	incorrect for $x < \mu$
mev	correct	correct
QRM	correct	correct
qrmtools	correct	correct
revdbayes	correct	correct
SpatialExtremes	correct	correct
texmex	correct	correct
TLMoments	correct	correct

1.2.1 Generalized Pareto distribution

For threshold exceedances, we simulated 50 exceedances from a generalized Pareto distribution $\mathsf{GP}(\sigma = 1000, \xi = -0.5)$ and from an exponential distribution with $\sigma = 1000$. The large scale value is intended to check the robustness of gradient-based algorithms; from an optimization perspective, it is wise to ensure that the gradient of each component, scale and shape, are not magnitudes apart. The data can easily be scaled prior to the optimization in case this is problematic.

Figure 1.1 shows the distribution of the score vector, i.e., the gradient of the log likelihood. The latter should vanish when evaluated at the maximum likelihood estimator $(\widehat{\sigma}, \widehat{\xi})$ provided $\widehat{\xi} > -1$. Most instances of non-zero gradient are attributable

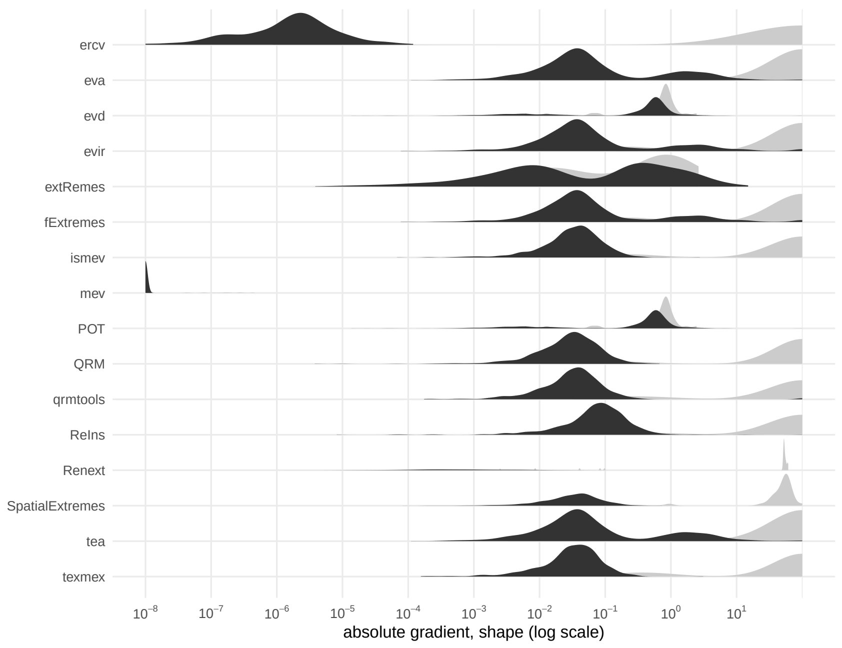

Fig. 1.1: Magnitude of the shape component of the score vector at the value returned by the optimization routine. The density plots are based on 1000 samples simulated from a generalized Pareto distribution with shape $\xi = -0.5$ and scale $\sigma = 1000$, split by simulations yielding a boundary case ($\widehat{\xi} = -1$, gray) and regular case ($\widehat{\xi} > -1$, black); the y-axis scale for each package is different to ease visualization. Results for samples for which the numerical routines failed to converge or the gradient is unevaluated are not shown.

to boundary cases with $\widehat{\xi} = -1$ not accounted for. Other discrepancies are due to numerical tolerance for convergence, but the differences in log likelihood relative to the maximum over all routines are negligible in most non-boundary cases investigated. Some routines, based on Nelder–Mead simplex algorithm, do not check the gradient but this is immaterial if the value of the function is nearly identical to that at the maximum likelihood estimate.

Figure 1.2 shows these differences through survival function plots, highlighting instances where the package fails to return correct values. Most packages do fine, except for a handful: `evd`, `extRemes` and `POT` (which uses routines from `evd`) stand out of the lot.

We can figure out the source of some of these oddities by plotting the distribution of the shape parameter estimates over all 1000 replications (see Figure 1.3). Both `POT` and `evd` return sampling distributions that are underdispersed relative to other implementations, while `ercv` and `extRemes` both have a large number of runs that return exactly zero for the shape parameter. The `QRM` package has unexpectedly small spread and a positive bias for estimation of ξ, different from other packages because it fails more often when ξ is negative. Both `ercv` and `extRemes` routines return zero shape estimates, leading to noticeable point masses. Only `SpatialExtremes` and `mev`

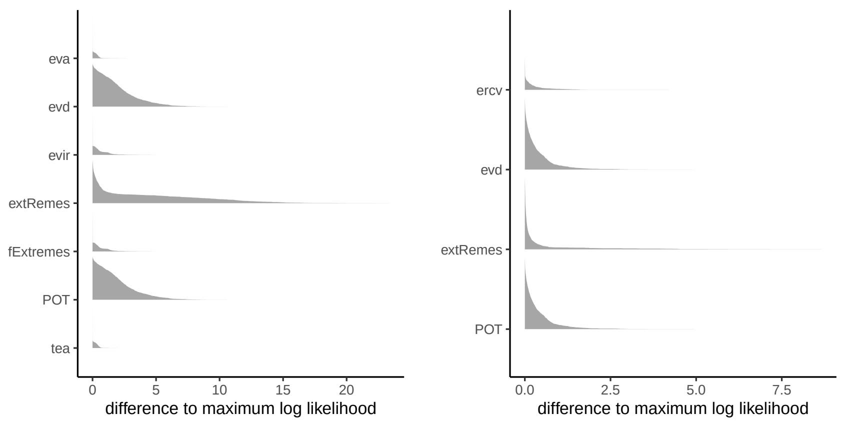

Fig. 1.2: Differences between the likelihood evaluated at the parameters returned by the routines and the maximum likelihood over all routines for generalized Pareto samples with negative shape ($\xi = -0.5$, left) and exponential samples (right), both with large scale parameter $\sigma = 1000$. Results for samples for which the numerical routines failed to converge are not shown. Only packages with 90% percentile giving a discrepancy larger than 10^{-4} are shown.

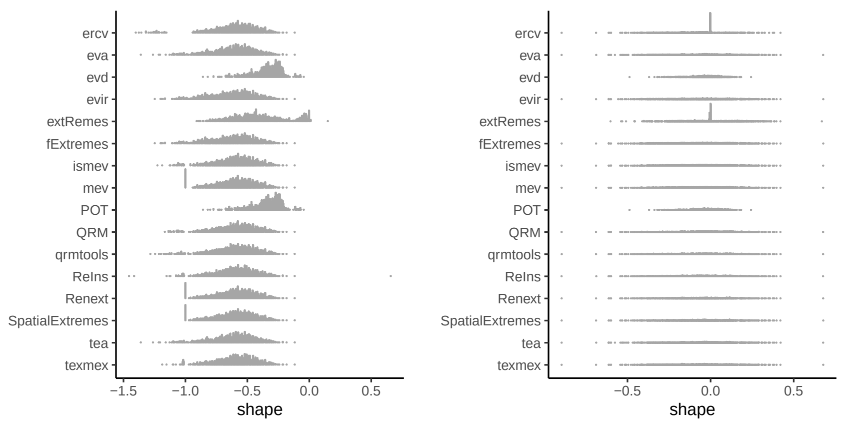

Fig. 1.3: Dot plots of shape parameter estimates returned by optimization routine for generalized Pareto samples with negative shape ($\xi = -0.5$, left) and exponential samples (right). Results for samples for which the numerical routines failed to converge are not shown.

correctly return $\xi = -1$, while `Renext` returns a hard-coded lower bound which can also be set to $\xi = -1$.

Some packages have routines that fail to converge often when the shape is negative; the most likely culprit for this is poor starting values. The routines in `ercv` and `fExtremes` (same as `evir`) fail often in small samples: for $n = 20$ exceedances, the function returned an error in 225 simulations. For the latter, the error is due to poor implementation of the log-likelihood that leads to infinite finite differences between

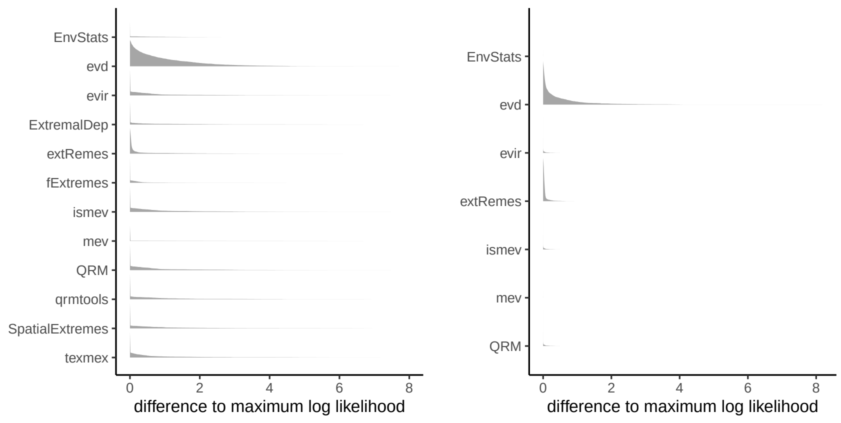

Fig. 1.4: Differences between the likelihood evaluated at the parameters returned by the routine and the maximum likelihood over all routines for generalized extreme value samples with negative shape (left) and exponential samples (right), both with large scale parameter $\sigma = 1000$. Results for samples for which the numerical routines failed to converge are not shown. Only packages with 90% percentile giving a discrepancy larger than 10^{-4} are shown.

estimates. For `QRM`, the choice of starting values, which cannot be modified by the user, is not adequate with strong negative shapes: it failed in more than 50 ($n = 20$), 122 ($n = 50$), 169 ($n = 100$) and 253 ($n = 1000$) times for negative shapes, indicating that the issue is not sample size. The `qrmtools` package, which supersedes `QRM`, has no such problems.

1.2.2 Generalized extreme value distribution

The optimization routines for the generalized extreme value distribution with scale $\sigma = 1000$ and shape parameters $\xi \in \{-0.5, 0, 0.5\}$ are better behaved and nearly all packages give identical results: only `evd` and `texmex` failed to converge and returned abnormally high shape values in a handful of instances out of 1000 simulations.

Unsurprisingly, the portrait (see Figures 1.4 and 1.5) is the same for the generalized extreme value distribution when it comes to boundary constraints: for example, `climextRemes` does not return shapes less than or equal to -1. `extRemes` has odd behaviour with a visible point mass at $\xi = 0$ in the simulations, even when this value has measure zero. Only `mev` and `SpatialExtremes` handle the boundary constraints. Figure 1.4 shows the difference in maximum likelihood returned by the packages, excluding cases with $\widehat{\xi} = -1$ for which the log likelihood becomes unbounded for combinations of σ and $\xi < -1$. Some packages, such as `evd`, also sometimes return a local optimum (perhaps due to use of the BFGS routine) and this in turn leads to erroneous comparisons of nested models.

Table 1.4 gives a breakdown of the number of instances for which the maximisation routine failed: two packages, `climextRemes` and `EnvStats`, stand out for negative shapes and the percentage of failures increases with the sample size.

Table 1.3: Number of failures for the optimization routine for maximum likelihood-based estimation of the generalized Pareto model (out of 1000 simulations).

(a) bounded tail ($\xi = -0.5$)

	20	50	100	1000
evir	225	15	0	0
fExtremes	225	15	0	0
QRM	50	122	169	253

(b) exponential ($\xi = 0$)

	20	50	100	1000
evir	37	0	0	0
fExtremes	37	0	0	0
QRM	4	12	7	0

(c) heavy tail ($\xi = 0.5$)

	20	50	100	1000
evir	7	0	0	0
fExtremes	7	0	0	0
QRM	1	0	0	0

Table 1.4: Number of failures for the optimization routine for maximum likelihood-based estimation of the generalized extreme value model (out of 1000 simulations).

(a) bounded tail ($\xi = -0.5$)

	100	1000	20	50
climextRemes	151	200	172	127
EnvStats	156	215	100	131
evir	0	0	27	0
fExtremes	23	0	92	28
ismev	0	0	4	0
mev	0	0	11	0
texmex	0	0	3	0

(b) light tail ($\xi = 0$)

	100	1000	20	50
climextRemes	0	0	4	0
EnvStats	0	0	4	0
fExtremes	0	0	1	0

(c) heavy tail ($\xi = 0.5$)

	100	1000	20	50
climextRemes	2	0	1	1
EnvStats	9	1	10	10
evir	0	0	4	0
fExtremes	2	0	5	3

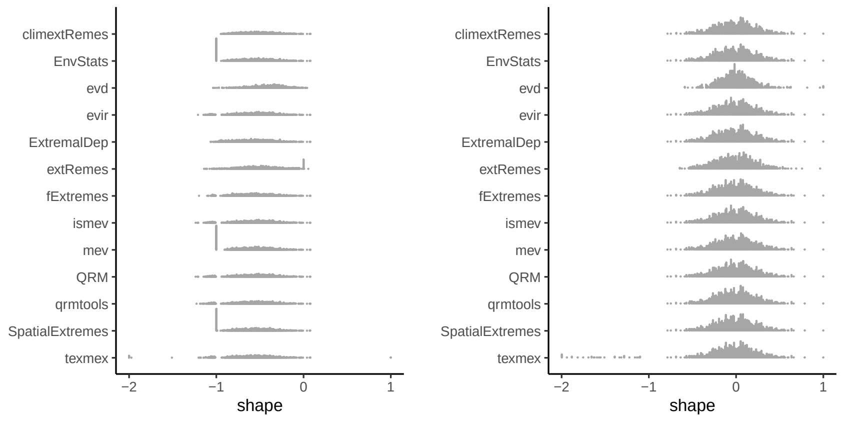

Fig. 1.5: Dot plots of shape parameter estimates returned by optimization routines for generalized extreme value samples with negative shape ($\xi = -0.5$, left) and Gumbel samples (right). Results for samples for which the numerical routines failed to converge are not shown.

2 Bayesian univariate inference for extremes

Creating a benchmark for Bayesian univariate analysis of extremes is complicated because approximate posterior samples returned by Markov chain Monte Carlo are autocorrelated so we cannot rely only on speed of execution or correctness. The effective sample size, which measures the equivalent number of independent draws from the posterior, is a better unit than the number of draws returned. We also factor in the amount of time it takes for the algorithm to proceed, but note that higher initialization costs may make such comparisons unfair if the cost of setup is larger than that of sampling. The different packages use different sampling algorithms: those that are implemented in low-level programming languages like `C` are inherently faster. Most packages that uses random walk Metropolis–Hastings steps discard initial draws during the so-called burn-in period (sometimes to tune proposal standard deviations, mostly to let the chain reach the posterior distribution and reduce impact of starting values). Other considerations include flexibility of methods, the choice of likelihood or the possibility to include covariates.

– In the `texmex` package, users can run multiple chains with burn-in and thinning, but iterations are preserved (which results in a heavier memory footprint). The choice of prior is restricted relative to most other packages. Rather than random walk Metropolis steps, proposals are drawn independently from a distribution which is centered at the maximum a posteriori, with a scale matrix matching the Hessian at the mode. This allows for good mixing, at the expense of a preliminary optimization (and tentatively terrible results should the latter fail to converge to the maximum a posteriori distribution).
– The `evdbayes` package has a comprehensive documentation, but some of its features are unconventional: the generalized Pareto model includes a location parameter that is modeled along as the threshold, but this is typically fixed. This leads to many proposals for the random walk Metropolis–Hastings ratio that lead to negative infinity, so we discard this altogether from the comparison. It can lead to adaption of the proposal.
– While very flexible, `extRemes` is noticeably slower than other packages and particularly inefficient with the default options (not setting `proposalParams` leads to effective sample sizes that are insufficient for any analysis to be reliable in our examples). It can be somewhat customized (and includes more flexible prior specification), but there is limited documentation on how to complete this in the package itself (but see the accompanying Journal of Statistical Software paper). The current options for evaluating the marginal likelihood in `BayesFactor` are unreliable and shouldn't be used (e.g., Neal, 2018).
– The `ExtremalDep` package also allows for estimation of the generalized extreme value distribution with potential covariates for the location parameter and censoring below a marginal threshold, using a random walk Metropolis-Hastings algorithm. However, the user needs to provide starting values and default values for the variances of the multivariate normal proposals, `sig0`, and the code returns an error if there is no censoring and covariates are provided. The output is less user-friendly than other packages, as there are no methods associated with the returned list.
– `revdbayes` provides independent draws from the posterior at a fraction of the costs of the other packages. Unless one has to include covariates in the parameters, it is the recommended approach.

Table 2.1: Effective sample size divided by the number of iterations (percentage).

(a) nonstationary generalized extreme value model

package	loc	loc (trend)	scale	shape
evdbayes	10.1	4.6	9.5	13.2
extRemes	6.1	6.9	6.9	4.9
STAN	90.0	86.3	100.0	69.7
texmex	6.6	7.0	7.0	7.2

(b) generalized Pareto model

package	scale	shape
extRemes	4.5	4.3
MCMC4extremes	6.6	7.1
STAN	48.3	43.8
texmex	12.4	10.3

2.1 Evaluation of effectiveness

We look at computation time (Figure 2.1) and effectiveness (Table 2.1) of algorithms as measured by the effective sample size, computed using the `coda` method (based on autocorrelation of the chains). Alternative better methods exist based on running multiple chains, but we forgo these.

The `revdbayes` implementation is exact and fastest, thus should be privileged in any problem not involving covariates. **Stan** simulates posterior samples using a Hamiltonian Monte Carlo algorithm. The latter is much more efficient than Metropolis-Hastings random walk proposals since it uses information about the geometry of the posterior distribution: the programming language requires bespoke definitions of the extreme value models and some care is necessary for shapes close to zero for the GEV distribution.

Of all the remaining packages, `texmex` gives the best performance because it uses proposals informed by the maximum a posteriori estimate. This wouldn't necessarily work with a multimodal objective function, but seems to do a good job in the simple scenarios we considered (and which are supported by the package). While we cannot know if we have converged to the target posterior distribution, the chains appear stationary.

The algorithm for `MCMC4Extremes` is fast, considering the number of observations it samples, but the implementation is crude and inefficient, including a burn-in period of 50K simulations, contrary to what the documentation states. The function is also not customizable.

The performance of `extRemes` is more dependent on tuning parameters than other implementations. Initial trials with the default parameter revealed problems: while the model starts at the MLE (so close to the stationary distribution), the default standard deviation of the normal random walk proposals are particularly ill-suited to the Venice sea level example. Trace plots (not shown) revealed lack of stationarity with default tuning parameters. With adapted proposals (and vague priors), the output seems satisfactory, but the effective sample size is subpar compared to other methods.

The `evdbayes` package includes a generalized Pareto model, but the latter also has a location parameter so is not directly comparable with other outputs.

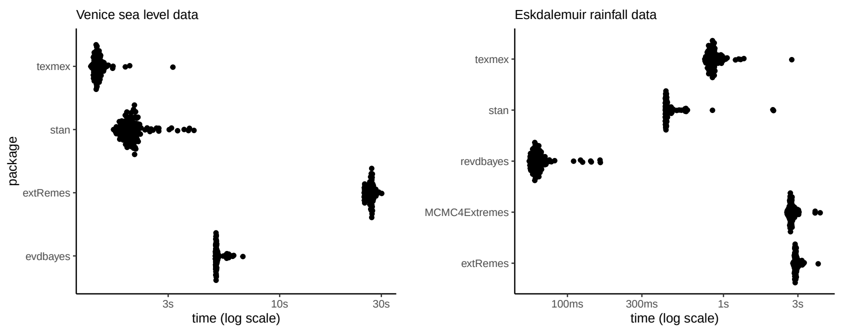

Fig. 2.1: Swarm plot of execution time (including preliminary optimization if necessary) of different numerical routines for a generalized extreme value model with linear trend in location fitted to the Venice sea level data (left) and the generalized Pareto distribution fitted to the Eskdalemuir rainfall data (right).

Data availability

The datasets analysed in Section 2 are available from the **mev** package.

3 Software version

Package	Version	License	Package	Version	License
BMAmevt	1.0.5	GPL (≥ 2)	jointPm	2.3.2	GPL (≥ 2)
climextRemes	0.3.0	BSD-3-clause, †	laeken	0.5.2	GPL (≥ 2)
coda	0.19-4	GPL (≥ 2)	lax	1.2.0	GPL (≥ 2)
CompRandFld ⋆	1.0.3-6	GPL (≥ 2)	lite	1.1.0	GPL (≥ 2)
copula	1.1-2	GPL (≥ 3), †	lmom	2.9	CPL-1.0
CTRE ⋆	0.1.0	GPL-3	lmomco	2.4.7	GPL
ercv	1.0.1	GPL (≥ 2)	Lmoments	1.3-1	GPL-2
erf ♯	0.0.1	GPL-3	lmomRFA	3.5	CPL-1.0
eva	0.2.6	GPL (≥ 2)	loo	2.6.0	GPL (≥ 3)
evd	2.3-6.1	GPL-3	longevity ♯	2023.03.22	GPL-3
evdbayes	1.1-3	GPL (≥ 2)	MCMC4Extremes	1.1	GPL-2
evgam	1.0.0	GPL-3	mev	1.15	GPL-3
evir	1.7-4	GPL (≥ 2)	mgcv	1.8-42	GPL (≥ 2)
evmix	2.12	GPL-3	mvPot	0.1.5	GPL-2
evtclass	1.0	GPL-3	POT	1.1-10	GPL (≥ 2)
exdex	1.2.1	GPL (≥ 2)	ptsuite	1.0.0	GPL-3
ExtremalDep	0.0.4-0	GPL (≥ 2)	QRM	0.4-31	GPL (≥ 2)
extremefit	1.0.2	GPL-2	qrmtools	0.0-16	GPL (≥ 3), †
ExtremeRisks	0.0.4	GPL (≥ 2)	RandomFields ⋆	3.3.14	GPL (≥ 3)
extRemes	2.1-3	GPL (≥ 2)	rbm ♯	1.0.0	MIT
extremeStat	1.5.5	GPL (≥ 2)	ReIns	1.0.12	GPL (≥ 2)
extremis	1.2.1	GPL (≥ 3)	Renext	3.1-3	GPL (≥ 2)
extremogram	1.0.2	GPL-3	revdbayes	1.5.1	GPL (≥ 2)
EVcopula ♯	0.1	GPL-3	RobExtremes	1.2.0	LGPL-3
evt0	1.1-4	GPL (≥ 2)	RTDE	0.2-1	GPL (≥ 2)
ev.trawl ⋆	0.1.0	MIT †	SimCop	0.7.0	GPL (≥ 2)
fCopulae	4022.85	GPL (≥ 2)	spatialADAI ♯	0.1.0	none
fExtremes	4021.83	GPL (≥ 2)	SpatialExtremes	2.1-0	GPL (≥ 2)
futureheatwaves	1.0.3	GPL-2	SpatialGEV	1.0.0	GPL-3
GEVcdn	1.1.6-2	GPL-3	tailDepFun	1.0.1	GPL-3
graphicalExtremes	0.2.0	GPL-3	tea	1.1	GPL-3
gremes ⋆	0.1.1	GPL-2	texmex	2.4.8	GPL (≥ 2)
hkevp	1.1.5	GPL	threshr	1.0.3	GPL (≥ 2)
IDF	2.1.2	GPL (≥ 2)	TLMoments	0.7.5.3	GPL (≥ 2)
INLA ♯	22.12.16	GPL-2	tsxtreme	0.3.3	GPL (≥ 2)
ismev	1.42	GPL (≥ 2)	VaRES	1.0.2	GPL (≥ 2)

Table 3.1: List of R packages, software licenses and version numbers at the time of the review. Additional file licenses are denoted with a † and packages archived from CRAN at the time of writing are denoted with a star ⋆. Packages available only from Github repository or personal websites are denoted with a ♯