CSIRO

Using GeneRaVE for Extremes: An Initial Attempt

Aloke Phatak

CMIS 09/96

June 2009

Enquiries should be addressed to:

Aloke Phatak
CSIRO Mathematical and Information Sciences
Leeuwin Centre, 65 Brockway Rd, Floreat, WA, Australia 6014
Private Bag 5, Wembley, WA, Australia 6913
Telephone: +61 8 9333 6184
Fax: +61 8 9333 6121
Email : Aloke.Phatak@csiro.au

Distribution list

Publications Officer	(1)
Authors	(1)

CONTENTS

1 Purpose **2**

2 GeneRaVE **2**

3 RChip Library **3**

4 HGengine **4**

5 Generalized Extreme Value Distribution **4**

6 Sparse Variable Selection for the GEV distribution **6**

6.1 Modelling the Mean 6

6.2 Modelling the Scale 7

7 A Simple Example **8**

7.1 Data 8

7.2 Invoking `RChip` 8

7.3 Modelling the Mean 9

7.4 Modelling the Scale 13

8 Discussion **15**

A Derivatives of the GEV Log-likelihood **18**

A.1 Modelling the Mean 18

A.2 Modelling the Scale 19

B Auxiliary R Functions for Implementing HGengine **21**

B.1 Modelling the Mean 21

B.2 Modelling the Scale 25

1 PURPOSE

This brief note summarizes some initial attempts to use GeneRaVE[1]—a rapid variable elimination method that can be applied to a wide class of models—to select variables in a statistical model of the extremes of a non-stationary sequence. In this statistical model, variations in the observed process are modelled as functions of a linear predictor in either the location parameter or the scale parameter of a generalized extreme value (GEV) distribution. Accompanying this document is an `Sweave`[2] file (`InitExtremes.Rnw`) that will allow the interested reader to reproduce not only the analyses carried out below, but the entire document itself.

2 GENERAVE

GeneRaVE is actually colloquial term for a suite of functions in the R library `RChip`[3] for analyzing the relationship between response variables and a set of predictors when the number of predictors, p, far exceeds the number of observations, N. Such data are commonly observed in gene expression studies, where the expression levels of tens of thousands of genes are measured from samples taken from tens or a few hundred individuals. Although the functions in `RChip` were originally conceived for analyzing data from bioinformatics studies, they are not of course limited to such data; they can be used in many other contexts where the $N < p$ problem restricts the scope of conventional model-building and parameter estimation procedures.

Underlying many of the functions in `RChip` is an engine that provides the mechanism for eliminating redundant variables in a wide variety of existing statistical models such as generalized linear models, multiclass logistic regression, proportional hazards survival models, and many others. This mechanism is based on the notion of model *sparsity*, and hence allows for the sensible estimation of parameters when the number of observations is much less than the number of potential explanatory variables.

Let[1] $L(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\phi})$ represent the likelihood function for a model that we would like to fit to data consisting of an $N \times p$ matrix of values of explanatory variables $\mathbf{X}$ and an $N \times 1$ vector of observations $\mathbf{y}$ of the response y. The vector $\boldsymbol{\beta}$ is a $p \times 1$ vector of parameters of primary interest, and $\boldsymbol{\phi}$ is a $q \times 1$ vector of parameters of secondary interest that might include, for example, scale and shape parameters.

To estimate the coefficients $\boldsymbol{\beta}$, GeneRaVE adopts a hierarchical Bayesian approach. The prior distribution for the elements of $\boldsymbol{\beta}$ is specified in such a way as to capture the underlying assumption that most of the coefficients are likely to be zero, and that specification leads to a prior for each of the coefficients β_i, $i = 1, 2, \ldots, p$, of the form

$$p(\beta_i) = \left[\frac{2^{(0.5-k)}}{\sqrt{\pi}\Gamma(k)}\right] \frac{\delta K_{(0.5-k)}(\delta|\beta_i|)}{(\delta|\beta_i|)^{(0.5-k)}}, \quad \delta = \sqrt{\frac{2}{b}} \tag{2.1}$$

[1]The discussion of GeneRaVE here follows closely the exposition of Kiiveri[1].

where K denotes a modified Bessel function of the third kind, and Γ denotes the gamma function. An expectation-maximization algorithm is used to obtain maximum *a posteriori* estimates of the coefficients β_i. Algorithmic details may be found in Kiiveri[1]. GeneRaVE has two hyperparameters k and b (or δ) that have to be selected. Cross-validation is used to determine k and b.

GeneRaVE can be seen as a generalization of $L1$-norm methods such as LASSO[4]; indeed, it includes LASSO as a special case.

3 RCHIP LIBRARY

For many 'standard' models, including Gaussian linear regression and many generalized linear models, the RChip library contains functions that can be used directly and that are easy to use. For example, the function HGgaussian contains only two required arguments:

```
> args(HGgaussian)
```

```
function (x, f, weights = rep(1, nrow(x)), sparsity.prior = "NG",
    bbess = 1e+07, kbess = 0, b0sc = 5, scale = -1, initb = NULL,
    no.prior = 1, control = HGcontrol(eta.lim = 1e+30))
NULL
```

They are the matrix $\mathbf{X}$ (x) of values of the explanatory variables and the N-vector $\mathbf{y}$ (f) of observations of the response; the remainder of the arguments, including the hyperparameters k (kbess) and b (bbess) have been assigned default values. Here's a simple example, adapted from the help file for HGgaussian:

```
> X <- matrix(rnorm(100 * 200), ncol = 200)
> # Generate response using a coefficient of 2 for the first
> # explanatory variable and with an intercept of one.
> y <- 2 * X[, 1] + 1 + 0.1 * rnorm(100)
> res <- HGgaussian(X, y)
> names(res)
```

```
[1] "beta"   "S"      "fv"     "varids" "sigma2"
```

The help file contains a description of the results object, but we can see which coefficients have been selected, and their corresponding values, by the following statements:

```
> # Variables selected (0 is the intercept term)
> res$varids
```

```
[1] 0 1
```

```
> # Values of coefficients
> res$beta[res$S]
```

```
[1] 1.008674 2.009463
```

Because the noise added in generating the vector y is small, HGgaussian easily recovers the 'true' model.

In addition to HGgaussian, RChip contains functions for two- and multi-class logistic regression, survival analysis, and a broad range of generalized linear models. However, in order to fit a regression where the response has a GEV distribution, we need to use another function, one that encodes general-purpose engine that imposes sparsity on a much broader class of statistical models.

4 HGENGINE

In theory, HGengine, the general-purpose engine outlined in Kiiveri[1] can be used to impose the sparsity constraint in eq. (2.1) (and other sparsity priors) on *any* statistical model as long as we can write a likelihood of the form $L(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\phi})$. We need only be able to calculate the first and second derivatives of the likelihood with respect to the linear predictor, $\eta_i = \mathbf{x}_i'\boldsymbol{\beta}$, and be able to calculate estimates of the parameters of secondary interest, $\boldsymbol{\phi}$. The arguments of HGengine are as follows:

```
> args(HGengine)
```

```
function (X, y, event, weights = rep(1, nrow(X)), bbess = 1e+07,
    kbess = 0, scale, bhat, Le, DLde, D2Lde, HGsc, sparsity.prior = "NG",
    no.prior = NULL, Offset = NULL, Offset.par.update = NULL,
    a = NULL, control = HGcontrol())
NULL
```

Along with the matrix of explanatory variables X, response vector y, and initial estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\phi}$ (bhat and scale, respectively), HGengine requires the following user-defined functions:

Le the value of the likelihood;

DLde the first derivative of the likelihood function with respect to the linear predictor η_i;

D2Lde the second derivative of the likelihood function with respect to η_i; and

HGsc the values of $\phi_1, \phi_2, \ldots, \phi_q$, the parameters of secondary interest.

Before illustrating how we might use HGengine, we first consider some aspects of the GEV distribution.

5 GENERALIZED EXTREME VALUE DISTRIBUTION

Let $Y_1, Y_2, \ldots, Y_N$ be independent random variables having a GEV distribution with *common* mean μ, scale σ, and shape ξ, for which we use the notation $Y_t \sim \text{GEV}(\mu, \sigma, \xi)$. For example, the sequence might represent the annual maximum sea-levels measured at a given location. Then, when $\xi \neq 0$, the log-likelihood can be written as[5]

$$\begin{aligned} l(\mu,\sigma,\xi) &= -\sum_{i=1}^{N}\log\sigma - (1+1/\xi)\sum_{i=1}^{N}\log\left[1+\xi\left(\frac{y_i-\mu}{\sigma}\right)\right] \\ &\quad -\sum_{i=1}^{N}\left[1+\xi\left(\frac{y_i-\mu}{\sigma}\right)\right]^{-1/\xi} \end{aligned} \tag{5.1}$$

provided that

$$1+\xi\left(\frac{y_i-\mu}{\sigma}\right) > 0,\ \text{ for } i = 1,2,\ldots,N \tag{5.2}$$

When eq. (5.2) is violated, the likelihood is zero, and the log-likelihood is $-\infty$ (alternatively, the negative log-likelihood equals $+\infty$). Prescott and Walden[6] use an alternative formulation, which, for common μ, σ, and shape parameter k, can be written as

$$l(\mu,\sigma,k) = -\sum_{i=1}^{N}\log\sigma - (1-k)\sum_{i=1}^{N}x_i - \sum_{i=1}^{N}\exp(-x_i) \tag{5.3}$$

where

$$x_i = -\frac{1}{k}\log\left[1-k\left(\frac{y_i-\mu}{\sigma}\right)\right] \tag{5.4}$$

and it is easy to show that $k = -\xi$. In the remainder of this note, we will use the formulation of Prescott and Walden.

When $Y_1, Y_2, \ldots, Y_N$ are independent, we can obtain estimates of the parameters of the GEV distribution by maximizing either eq. (5.1) or eq. (5.3). No analytical solution exists, but numerical optimization methods can be used, as in, for example, the function `gev.fit` contained in the R package `ismev`[7].

It may be, however, that the sequence $Y_1, Y_2, \ldots, Y_N$, is non-stationary, that the data do not support a model which assumes a constant distribution throughout time. For example, if annual maximum sea-levels appears to change linearly over time, a suitable model for Y_t might then be

$$Y_t \sim \text{GEV}(\mu(t),\sigma,k)$$

where

$$\mu(t) = \beta_0 + \beta_1 t$$

More generally, we can write

$$\mu(t) = \mathbf{x}'\boldsymbol{\beta}$$

where the vector $\mathbf{x} = (x_1, x_2, \ldots, x_p)'$ contains p covariates. Non-stationarity may also be expressed in terms of the other GEV parameters; for any of them, we can write

$$\theta(t) = h(\mathbf{x}'\boldsymbol{\beta}) = h(\eta) \tag{5.5}$$

where θ denotes either μ, σ, or k, though shape parameters are difficult to estimate with precision[5, p. 106]. The function h is the inverse-link, and $\eta = \mathbf{x}'\boldsymbol{\beta}$ is the linear predictor. For example, it may be appropriate to model the scale parameter as a function of the linear predictor:

$$\sigma(t) = \exp(\eta) \ \ (\Longrightarrow \log \sigma(t) = \eta)$$

Here, the exponential inverse-link function ensures that $\sigma > 0$ for all values of the covariates.

The class of models implied by eq. (5.5) is similar to generalized linear models (GLMs). However, as Coles[5, p. 108] points out, none of the standard results or computational tools are directly transferable to the estimation for GEV-distributed variables because the GLM family is restricted to distributions in the exponential family. Numerical methods can be used to maximize the log-likelihood, which can be written as

$$l(\mu(t), \sigma(t), k(t)) = -\sum_{i=1}^{N} \log \sigma(t) - (1-k)\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} \exp(-x_i) \tag{5.6}$$

where

$$x_i = -\frac{1}{k(t)} \log\left[1 - k(t)\left(\frac{y_i - \mu(t)}{\sigma(t)}\right)\right] \tag{5.7}$$

It is unlikely that we would correctly be able to specify models for all three GEV parameters simultaneously (nor would the resulting model be necessarily parsimonious!), but the function `gev.fit` in the package `ismev` allows the user to do so. It cannot, however, be used when $N < p$, nor does it provide functionality for automatic variable selection. `HGengine` allows only one of the GEV parameters to be modelled, but the sparsity prior enforces automatic variable selection, even when $N < p$.

6 SPARSE VARIABLE SELECTION FOR THE GEV DISTRIBUTION

The function `HGengine` requires (see §4) first (`DLde`) and second (`D2Lde`) derivatives of the (log-) likelihood with respect to the linear predictor η, and they will depend on the parameter we choose to model, as will the function `HGsc` required to calculate the values of the parameters that are not modelled. We consider first a model for the mean μ and then for the scale parameter σ.

6.1 Modelling the Mean

To model the mean, we will use a simple identity link function:

$$\mu_i = \mathbf{x}_i'\boldsymbol{\beta} = \eta_i, \quad i = 1, 2, \ldots, N$$

Hence, the likelihood, which we denote by l_μ to indicate that the mean is being modelled, is given by

$$l_\mu = -N \log \sigma - (1-k)\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} \exp(-x_i) \tag{6.1}$$

where

$$x_i = -\frac{1}{k}\log\left[1 - k\left(\frac{y_i - \eta_i}{\sigma}\right)\right] \tag{6.2}$$

It is straightforward to show (see Appendix A.1) that the first derivative of l_μ with respect to η_i is

$$\frac{\partial l_\mu}{\partial \eta_i} = \frac{(1-k)}{\sigma}\exp(kx_i) - \frac{1}{\sigma}\exp\left[(k-1)x_i\right] \tag{6.3}$$

with x_i defined in as in eq. (6.2). The second derivative (see Appendix A.1) is

$$\frac{\partial^2 l_\mu}{\partial \eta_i^2} = -\frac{(1-k)}{\sigma^2}\left[k\exp(kx_i) + \exp\left\{(k-1)x_i\right\}\right]\exp(kx_i) \tag{6.4}$$

Functions to calculate the likelihood (`LE.mu`) and the first and second derivatives are also shown in Appendix B. Each of the latter two functions returns a vector whose ith element is either $\partial l_\mu/\partial \eta_i$ (`DLDE.mu`) or $\partial^2 l_\mu/\partial \eta_i^2$ (`D2LDE.mu`).

Since we are interested in modelling the mean as a function of covariates, the scale and shape parameter constitute the parameters of secondary interest $\boldsymbol{\phi}$. A function to calculate the value of these parameters given the data and the current estimate of the η_i is outlined in Appendix B.

6.2 *Modelling the Scale*

To model the scale parameter, we will use an exponential inverse-link function, i.e.,

$$\sigma_i = \exp(\mathbf{x}_i'\boldsymbol{\beta}) = \exp(\eta_i)$$

and the log-likelihood, which we denote by l_σ to denote that the scale is being modelled, is given by

$$l_\sigma = -\sum_{i=1}^{N}\log\sigma_i - (1-k)\sum_{i=1}^{N}x_i - \sum_{i=1}^{N}\exp(-x_i) \tag{6.5}$$

where

$$x_i = -\frac{1}{k}\log\left[1 - k\left(\frac{y_i - \mu}{\sigma_i}\right)\right] \quad \text{and } \sigma_i = \exp(\eta_i) \tag{6.6}$$

The first and second derivatives of l_σ with respect to η_i are are somewhat messier than the derivatives with respective to μ_i, and the are derived in Appendix A.2. The expressions are as follows:

$$\frac{\partial l_\sigma}{\partial \eta_i} = \frac{\partial l_\sigma}{\partial \sigma_i}\cdot\frac{\partial \sigma_i}{\partial \eta_i} = -\frac{1}{k\sigma_i}\left(P_i + Q_i\right)\cdot\frac{\partial \sigma_i}{\partial \eta_i} \tag{6.7}$$

where

$$\begin{aligned} P_i &= 1 - \exp(-x_i) && (6.8)\\ Q_i &= \exp\left[(k-1)x_i\right] - (1-k)\exp(kx_i) && (6.9)\end{aligned}$$

and because $\sigma_i = \exp(\eta_i)$, $\partial\sigma_i/\partial\eta_i = \exp(\eta_i)$.

The second derivative can be expressed in terms of quantities that have already been calculated, i.e.,

$$\frac{\partial^2 l_\sigma}{\partial\sigma_i^2} = -\frac{1}{\sigma_i}\frac{\partial l_\sigma}{\partial\sigma_i} - \left[\frac{1}{\sigma_i}Q_i + \exp(-x_i)\cdot\frac{\partial x_i}{\partial\sigma_i}\right]\frac{\partial x_i}{\partial\sigma_i} \tag{6.10}$$

where $\partial x_i/\partial\sigma_i$ can be obtained by differentiating eq. (6.2) and is shown in eq. (A.2.5).

Functions to calculate the likelihood (`LE.sigma`) and the first (`DLDE.sigma`) and second (`D2LDE`) derivatives are listed in Appendix B. Here, we model the scale parameter as a function of covariates, and so the mean and shape parameters constitute ϕ, the parameters of secondary interest. A function to calculate the value of these parameters given the data and the current estimate of the η_i is also outlined in Appendix B.

7 A SIMPLE EXAMPLE

7.1 Data

The dataset was provided by Mark Palmer. It consists of yearly maximum rainfall amounts between 1953 and 1991 at a single station and associated mean sea level pressures on a 14×11 grid. A plot of the yearly maxima appears in Fig. 7.1, and there appears to be an upward trend after the mid-1960s. The locations of the station and the grid points are shown in Fig. 7.2. For the year 1956, all pressure values are constant (an error!), so it has been removed from the dataset before any further analyses. The resulting response vector `y` (of yearly maxima) is therefore of length 38, and the matrix of explanatory variables `X` is of dimension 38×154.

Because the objective here is to illustrate how `HGengine` might be used to model GEV-distributed random variables, little in the way of exploratory analysis of the data was carried out. However, there is not a strong relationship between the yearly maximum rainfall and mean sea-level pressure: the highest squared correlation with rainfall is only about 0.35.

7.2 Invoking `RChip`

The version of `RChip` that is publicly available from CSIRO Bioinformatics[2] does not contain more advanced functions such as `HGengine`, and users will have to obtain more recent 'beta' versions. After installing one of these versions, `RChip` can be invoked by an R statement of the form

```
> require(RChip, lib.loc = "/home/pha041/src/RChip/RChip_3.0.6")
```

or something similar. In the analysis below, we will also need to invoke functions that are ordinarily hidden from the user; these can be accessed using the double (`::`) or triple(`:::`) colon operators.

[2] https://www.bioinformatics.csiro.au/GeneRave/index.shtml

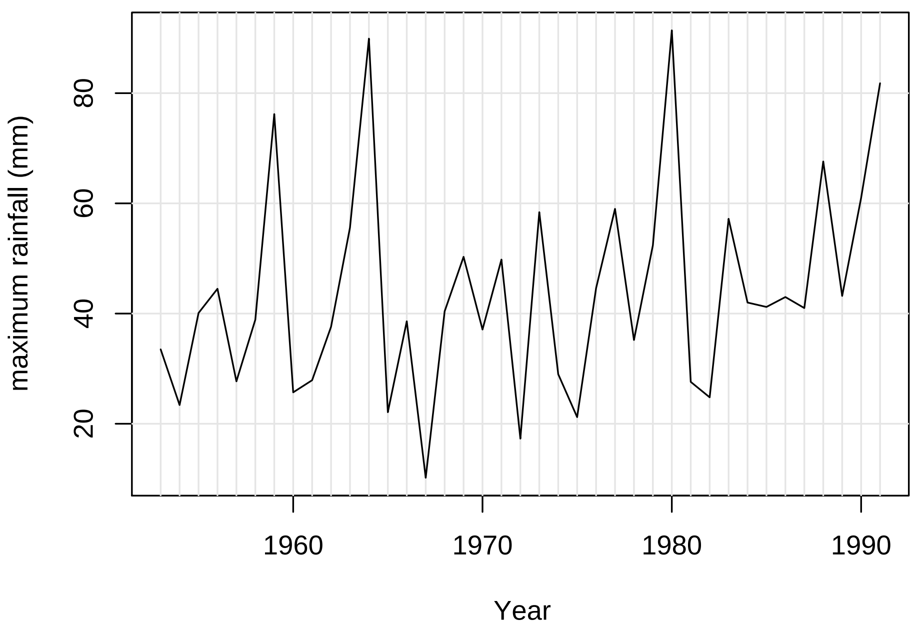

Figure 7.1. Maximum yearly rainfall in the period 1953–1991 at the station shown in Fig. 7.2.

7.3 *Modelling the Mean*

Recall from §4 that in addition to the data `y` and `X`, `HGengine` requires additional arguments, including initial estimates of the coefficients $\boldsymbol{\beta}$ and the parameters of secondary interest $\boldsymbol{\phi}$. We obtain the former by using an internal function that calculates a ridge-type estimator:

```
> initB <- RChip:::HGinit.g(cbind(1, scale(X)), y, lb = 1, weights = 1)
> initB[1:5]
```

```
[1] 42.6641026 -7.8255464 -6.2542437 -0.2515796  5.2602418
```

and the latter by using maximum-likelihood estimates without including any covariates:

```
> require(ismev)
> y.mle <- gev.fit(y)
> initParam <- y.mle$mle[c(2, 3)]
> initParam[2] <- -initParam[2]
```

Hence, the initial estimates of the scale and shape parameter k are

```
> initParam
```

```
[1] 15.53336209  0.03168927
```

The function `gev.fit` is part of the package `ismev`. Here, the elements of `initParam` are initial estimates of the scale and shape. Estimation in `ismev` uses a log-likelihood of the form given in eq. (5.1), but since we are using eq. (5.3), the estimate of the shape parameter (k) that we require is the *negative* of the shape parameter (ξ) in eq. (5.1).

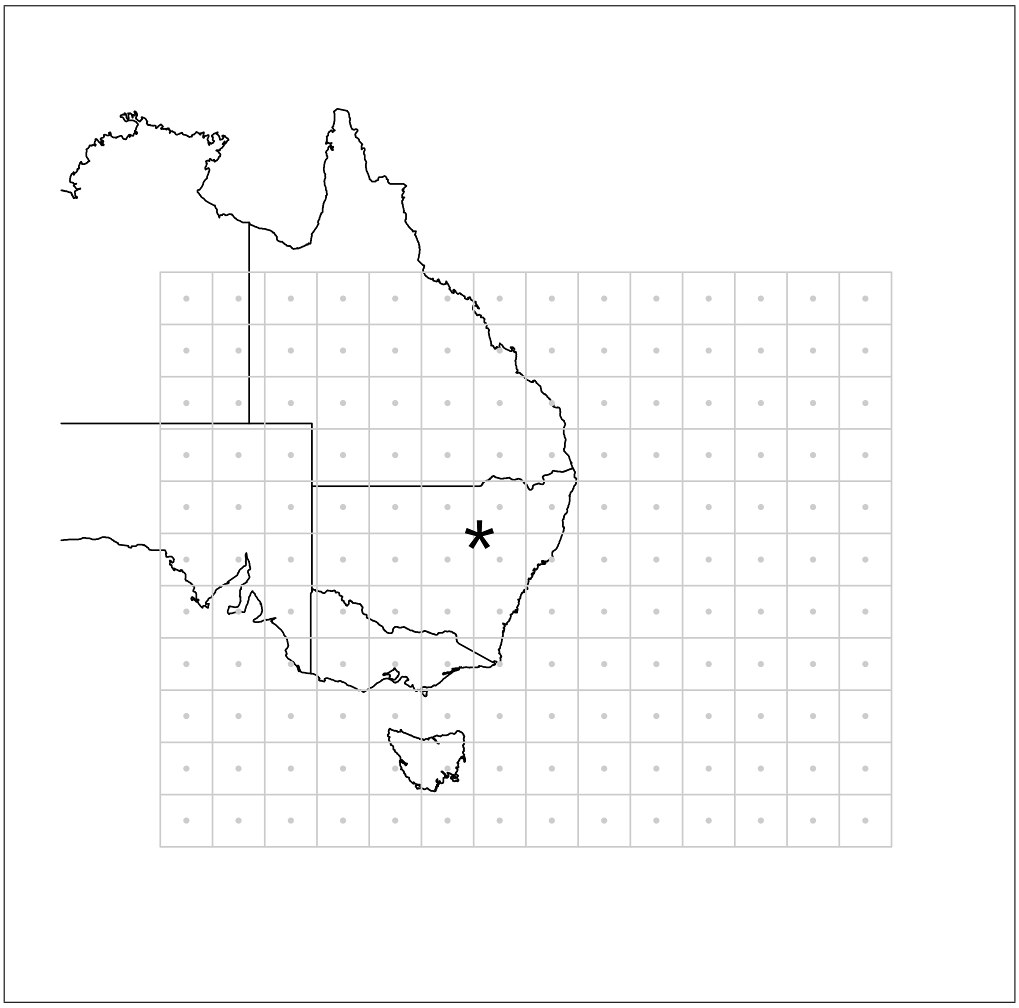

Figure 7.2. Map of Eastern Australia showing the location of the station (asterisk) at which yearly maximum rainfall was calculated, and the 11 × 14 grid points at which the corresponding mean sea-level pressure was available.

bbess	# vars	bbess	# vars
5×10^2	2	1×10^5	13
1×10^3	6	5×10^5	16
5×10^3	8	1×10^6	17
1×10^4	11	5×10^6	16
5×10^4	14	1×10^7	20

Table 7.1. Effect of increasing `bbess` on the number of variables selected when the mean is modelled using an identity link. The value of `kbess` is held constant at 0.6.

Additional arguments to `HGengine` include functions to calculate the log-likelihood and its first and second derivatives with respect to the linear predictor. These functions are outlined in Appendix B, and we now have all the pieces to run `HGengine` and carry out variable selection. The syntax is straightforward:

```
> mu.res <- HGengine(cbind(1, scale(X)), y,
+                     Le = LE.mu, DLde = DLDE.mu, D2Lde = D2LDE.mu,
+                     HGsc = HGSC.mu,
+                     scale = initParam, bhat = initB,
+                     bbess = 5e+03, kbess = 0.60)
```

Note that the matrix of explanatory variables is scaled so that its columns have unit sample variance and that a columns of ones has been explicitly added for the intercept term.

```
> nVars <- sum(mu.res$S[-1])
> cat("Number of variables selected:", nVars)

Number of variables selected: 8

> colsX <- (1:ncol(X))[mu.res$S[-1]]
> cat("Columns of X that are selected:", colsX)

Columns of X that are selected: 1 5 30 50 57 110 120 151
```

For this combination of `bbess` (= 5×10^3) and `kbess` (= 0.60), 8 variables, not including the intercept term, have been chosen, and the column (grid point) numbers are shown above. In Fig. 7.2, the grid points correspond to column numbers 1–154 arranged from left-to-right, top-to-bottom. The variables that are selected line roughly along a NW-SE line in two clusters, on either side of the station location. In general, the number of variables selected increases as `bbess` and `kbess` increase. Table 7.1 illustrates this pattern for fixed `kbess` and increasing `bbess`, though the trend is not monotonic. Cross-validation is used to select a `bbess/kbess` combination, but I have not carried it out for this illustrative exercise. The estimate of the mean of the ith observation is shown in Fig. 7.3: because we are using the identity link, $\hat{\mu}_i = \hat{\eta}_i = \mathbf{x}_i'\hat{\boldsymbol{\beta}}$, where $\hat{\boldsymbol{\beta}}$ contain the coefficients estimated by `HGengine`, of which 146 (= 154 − 8)are zero.

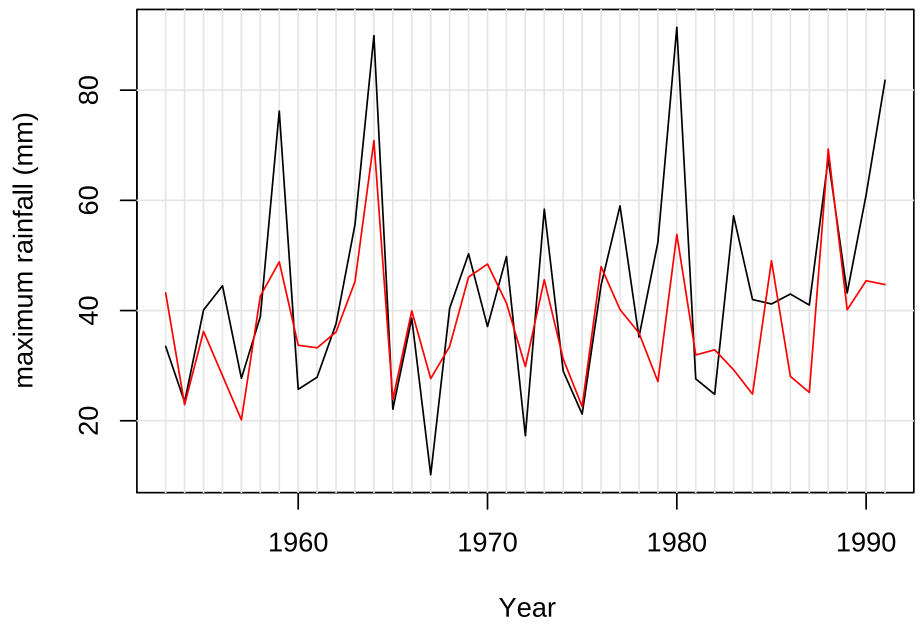

Figure 7.3. Maximum yearly rainfall (black) and the estimate of the mean $\hat{\mu}_i$ from a sparse regression containing 8 variables (red).

The object `mu.res` also contains the final estimates of the scale and shape parameter, and it is instructive to compare them with the estimates obtained earlier when we assumed a constant mean.

```
> mu.res$scale
[1] 11.8996201  0.1060878
> c(y.mle$mle[2], -y.mle$mle[3])
[1] 15.53336209  0.03168927
```

Thus, when we fit a conditional mean, the estimate of scale parameter decreases (as we might expect). The shape parameter k increases; alternatively, $\xi(= -k)$ *decreases*. We do not, however, have expressions for standard errors and so cannot determine whether these changes are significant. Although expressions for (approximate) standard errors of the parameters are available for many $L1$-norm methods, none exist for the GeneRaVE family and we have to resort to resampling methods to obtain standard errors.

The optimization routine (`optim`) used in the function `HGSC.mu` to estimate the scale and shape parameters for a given value of the linear predictor is very sensitive to the optimization method used internally and its associated parameters. By default, `optim` uses the simplex method of Nelder and Mead (see, for example, Avriel[9]), which has three adjustable coefficients, each of which is associated with one of its three constituent steps: reflection, expansion, and contraction. When default values are used, the function fails to converge. For this reason, I had to experiment with different values of the control parameters until the routine converged. As we will see below, setting the correct control parameters is even more critical when we try to estimate the mean and shape for a fixed value of the scale parameter.

7.4 *Modelling the Scale*

Initial estimates of the coefficients $\boldsymbol{\beta}$ and of the parameters of secondary interest $\boldsymbol{\phi}$ (here, the location and shape parameter) can be obtained in the same way as before:

```
> initB <- RChip:::HGinit.g(cbind(1, scale(X)), log(y.mle$mle[2] +
+     runif(y, -4, 4)), lb = 1, weights = 1)
> initParam <- y.mle$mle[c(1, 3)]
> initParam[2] <- -initParam[2]
> initParam
[1] 35.22676331  0.03168927
```

To obtain the initial estimates `initB`, I have simply used a ridge-estimator where the response variable is the log of the unconditional scale parameter plus some noise. Then, to run the function `HGengine` as before, we simply use the appropriate functions for the first and second derivatives with respect to the linear predictor, and for the function that calculates the parameters of secondary interest.

```
> sigma.res <-
+   HGengine(cbind(1, scale(X)), y,
+            Le = LE.sigma,
+            DLde = DLDE.sigma,
+            D2Lde = D2LDE.sigma,
+            HGsc = HGSC.sigma,
+            scale = initParam,
+            bhat = initB,
+            bbess = 5e+03,
+            kbess = 0.45)
> nVars <- sum(sigma.res$S[-1])
> cat("Number of variables selected:", nVars)
Number of variables selected: 2
> colsX <- (1:ncol(X))[sigma.res$S[-1]]
> cat("Columns of X that are selected:", colsX)
Columns of X that are selected: 126 141
```

Thus, for this combination of `bbess` $(= 5 \times 10^3)$ and `kbess` $(= 0.45)$, only 3 variables are selected, including the intercept term. In trying different combinations of `bbess` and `kbess`, however, I found that the function `HGSC.sigma` often failed. For example, as Table 7.2 shows (R code to generate it not shown) for a range of `bbess/kbess` values, this is the rule rather than the exception.

		kbess											
		0.4	**0.45**	**0.5**	**0.55**	**0.6**	**0.65**	**0.7**	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**
	10	NA	1	1	2	3	NA	NA	NA	NA	NA	NA	NA
	50	NA	1	2	3	NA	NA	NA	9	NA	NA	NA	NA
	100	NA	1	2	3	NA	NA	NA	NA	NA	NA	NA	NA
	500	NA	1	2	NA	NA	NA	NA	NA	NA	NA	NA	NA
	1000	NA	2	2	NA	NA	8	NA	NA	NA	NA	NA	NA
bbess	**5000**	1	2	3	NA	NA	NA	NA	NA	NA	NA	NA	103
	10000	1	2	NA	NA	NA	NA	NA	NA	NA	NA	NA	100
	50000	1	2	NA	NA	NA	NA	NA	NA	NA	NA	65	33
	1e+05	1	2	NA	NA	9	NA	NA	NA	NA	NA	88	NA
	5e+05	1	2	NA	NA	NA	NA	NA	NA	NA	69	NA	32
	1e+06	1	2	NA	NA	NA	NA	NA	NA	NA	88	NA	35
	5e+06	1	2	NA	NA	NA	NA	NA	NA	42	33	33	34

Table 7.2. Number of variables selected for different `bbess/kbess` combinations when the scale parameter is modelled with an exponential inverse-link. Where there is an '`NA`' entry, the function `HGSC.sigma` failed.

Moreover, running the same hyperparameter combination twice does not always result in the same variables being selected, nor in the same number of variables, and `HGSC.sigma` sometimes fails for `bbess/kbess` combinations that have previously yielded sensible results. Where different variables are sometimes selected, however, they tend to be at neighbouring grid points.

In the example above, the estimates of the coefficients are

```
> sigma.res$beta[sigma.res$S]
```

```
[1] 2.6439726 0.2156307 0.2345306
```

where the first element in the vector is the intercept. To assess whether the estimates of $\boldsymbol{\beta}$ obtained by `HGengine` are sensible, we can use the use the variables selected above in a straightforward regression using the function `gev.fit` from the library `ismev` as follows:

```
> sigma.gevfit <- gev.fit(y, scale(X[, colsX]), siglink = exp,
+                         sigl = 1:length(colsX))
```

```
> sigma.gevfit$mle
```

```
[1] 33.67719186  2.63960154  0.23578169  0.25419606 -0.03661750
```

The first element is the estimate of the mean, the last element is the estimate of the shape parameter $\xi(= -k)$, and in between are the estimates of the coefficients in the linear predictor. Comparing these with the estimates from `HGengine`, we can see that they are similar, though the latter are very slightly shrunken. The estimates of the location and shape parameter are also similar, and

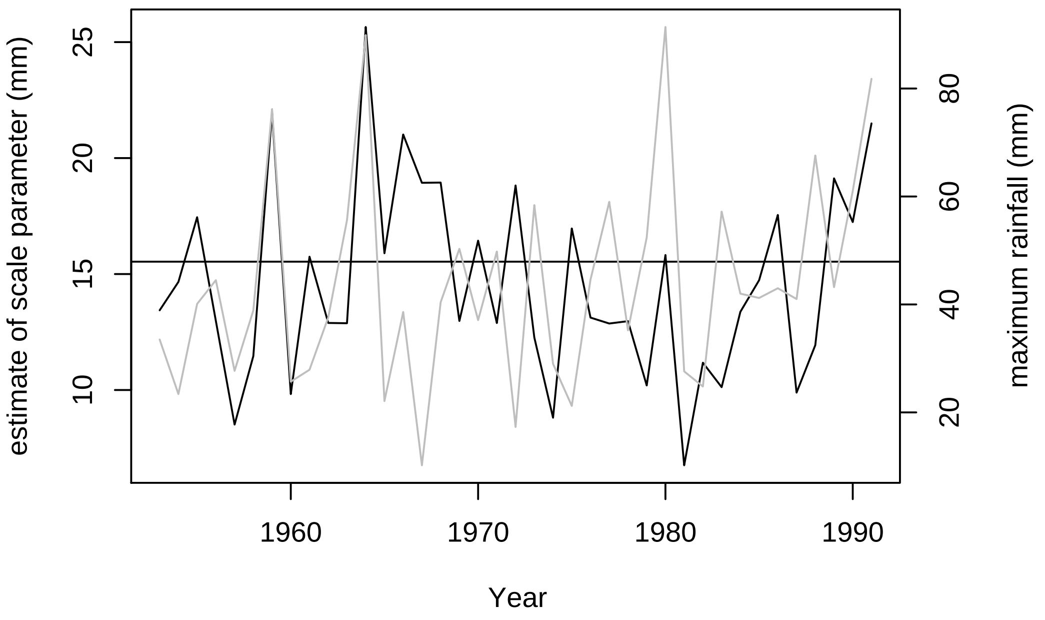

Figure 7.4. Estimates of scale parameter $\hat{\sigma}_i$ from a sparse regression (black). Also shown is the MLE estimate of the stationary scale parameter (horizontal black line) and the maximum yearly rainfall (grey).

they are given by

```
> sigma.res$scale
[1] 33.95157698  0.04389743
```

Figure 7.4 shows the estimates of the scale parameters $\hat{\sigma}_i$, and, for reference, the stationary estimate and the yearly maximum rainfall.

8 DISCUSSION

The objective here has been to demonstrate 'proof-in-principle' use of the function `HGengine` for rapid variable selection when the response variable has a GEV distribution. Using `HGengine`, we can model either the location or the scale parameter as a function of the linear predictor $\eta_i = \mathbf{x}_i'\boldsymbol{\beta}$ using appropriate link functions. Constructing parsimonious models in this way may be useful in, for example, downscaling applications, where potential explanatory variables include gridded outputs from general circulation models. These outputs can include mean sea-level pressure, geopotential heights, relative humidity, dew-point temperature depression, and many others. In even a modest-sized grid such as the one used here, the number of potential explanatory variables p can number in the many hundreds, and even thousands; furthermore, since the length of a series consisting of block maxima—for example, yearly maximum rainfall—is likely to be relatively short, $N \ll p$, and `HGengine` provides a rapid and efficient means of variable selection in circumstances where conventional methods such as stepwise regression will fail. Of course—keeping in mind all the caveats about empirical variable selection—we would hope that the variables selected by GeneRaVE *might* ultimately lead to a sensible and defensible interpretation of the drivers of extreme events and behaviour.

There are two aspects of using GeneRaVE that need to be explored further, one specific and immediate, relating to the specific use of `HGengine` for extremes, the other more general, about variable selection from gridded spatial data:

1. The estimation of parameters in `HGengine`, using the auxiliary functions described in Appendix B, must be stabilized. When `HGengine` fails to yield a solution, it does so because one of the auxiliary functions `HGSC.mu` or `HGSC.sigma`, which estimate the parameters ϕ of secondary interest, has failed. It appears that at some point during the optimization, the values of the second derivative grow without limit. The solution may be as simple as adding a statement to constrain the second derivative, or another optimization method in the function `optim` may be required.
2. In modelling the mean as a function of the linear predictor, eight locations on the grid were selected, and they were divided into two broad clusters. The grid points within a cluster are not in general adjacent to one another, though we might expect them to be so because of the smooth underlying spatial field. Like its other $L1$ counterparts, GeneRaVE does not explicitly enforce smoothness when the variables are ordered spatially, temporally, or in any other meaningful way, as in, for example, infrared spectra discretized at many wavelengths. There is, however, a generalization of LASSO, known as the fused LASSO[10], that may suggest a way forward. In addition to the LASSO constraint $\sum_{j=1}^{p} |\beta_j| \leq s$, the fused LASSO imposes an additional constraint that encourages sparsity in the differences between adjacent coefficients: $\sum_{j=2}^{p} |\beta_j - \beta_{j-1}| \leq t$. As Tibshirani *et al.*[10] point out, generalizations of the fused LASSO to gridded data are also possible, though computationally expensive because the number of constraints is of the order of p^2. It is not immediately clear how to modify the prior in eq. (2.1) to encourage sparsity in the differences of coefficients one unit apart in any direction, but that would certainly be an interesting research question!

ACKNOWLEDGMENTS

Thanks to Harri Kiiveri for many helpful discussions and much advice on using `HGengine` and to Mark Palmer for the extremes data.

REFERENCES

[1] Kiiveri, H. (2008). A general approach to simultaneous model fitting and variable elimination in response models for biological data with many more variables than observations. *BMC Bioinformatics*, **9** (1), 195.

[2] Leisch, F. (2002). Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002—Proceedings in Computational Statistics*, pp. 575–580. Heidelberg: Physica Verlag.

[3] CSIRO Bioinformatics (2008). *GeneRaVE*. `https://www.bioinformatics.csiro.au/GeneRave/index.shtml`.

[4] Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society*. Series B (Methodological). **58** (1), 267–288.

[5] Coles, S. (2001). *An Introduction to Statistical Modeling of Extreme Values*. London:Springer-Verlag.

[6] Prescott, P. and Walden, A.T. (1980). Maximum likelihood estimation of the parameters of the generalized extreme value distribution. *Biometrika*, **67** (3), 723–724.

[7] Stephenson, A. (2006). `ismev`: An Introduction to Statistical Modeling of extreme values. Original S functions by Stuart Coles. R port and R documentation by Alec Stephenson. R package version 1.3. `http://www.maths.lancs.ac.uk/~stephena/`.

[8] Natural Environment Research Council (Great Britain). (1975). *Flood Studies Report, Vol. I, Hydrological Studies*. London: Natural Environment Research Council.

[9] Avriel, M. (1976). *Nonlinear Programming: Analysis and Methods*. Englewood Cliffs, N.J.: Prentice-Hall.

[10] Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society*, Ser. B, **67** (1), 91–108.

A DERIVATIVES OF THE GEV LOG-LIKELIHOOD

A.1 Modelling the Mean

When we use an identity link to model the mean (i.e., $\mu_i = \eta_i$), the log-likelihood is given in eqs. (6.1) and (6.2) as

$$l_\mu = -N \log \sigma - (1-k) \sum_{i=1}^{N} x_i - \sum_{i=1}^{N} \exp(-x_i) \tag{A.1.1}$$

with

$$x_i = -\frac{1}{k} \log\left[1 - k\left(\frac{y_i - \eta_i}{\sigma}\right)\right] \quad \left(\Longrightarrow 1 - k\left(\frac{y_i - \eta_i}{\sigma}\right) = \exp(-kx_i)\right) \tag{A.1.2}$$

Then,

$$\frac{\partial l_\mu}{\partial x_i} = -(1-k) + \exp(-x_i) \tag{A.1.3}$$

Furthermore, we have that

$$\begin{aligned} \frac{\partial x_i}{\partial \eta_i} &= -\frac{1}{k}\frac{k}{\sigma}\left[1 - k\left(\frac{y_i - \eta_i}{\sigma}\right)\right]^{-1} \\ &= -\frac{1}{\sigma}\left[\exp(-kx_i)\right]^{-1} \\ &= -\frac{1}{\sigma}\exp(kx_i) \end{aligned} \tag{A.1.4}$$

where the simplification in going from the first to the second step occurs upon rearranging the expression for x_i and inserting the result. Finally, using the chain rule, we can write that

$$\begin{aligned} \frac{\partial l_\mu}{\partial \eta_i} &= \frac{\partial l_\mu}{\partial x_i} \cdot \frac{\partial x_i}{\partial \eta_i} \\ &= \left[-(1-k) + \exp(-x_i)\right]\left[-\frac{1}{\sigma}\exp(kx_i)\right] \\ &= \frac{(1-k)}{\sigma}\exp(kx_i) - \frac{1}{\sigma}\exp\left[(k-1)x_i\right] \end{aligned} \tag{A.1.5}$$

with x_i as above.

To calculate the second derivative, we use the chain rule again, and can write that

$$\frac{\partial^2 l_\mu}{\partial \eta_i^2} = \frac{\partial}{\partial x_i}\left(\frac{\partial l_\mu}{\partial \eta_i}\right) \cdot \frac{\partial x_i}{\partial \eta_i} \tag{A.1.6}$$

Hence,

$$\begin{aligned} \frac{\partial^2 l_\mu}{\partial \eta_i^2} &= \left\{\frac{k(1-k)}{\sigma}\exp(kx_i) + \frac{(1-k)}{\sigma}\exp\left[(k-1)x_i\right]\right\} \cdot \left[-\frac{1}{\sigma}\exp(kx_i)\right] \\ &= -\frac{(1-k)}{\sigma^2}\exp(kx_i)\left\{k\exp(kx_i) + \exp\left[(k-1)x_i\right]\right\} \end{aligned} \tag{A.1.7}$$

A.2 *Modelling the Scale*

We use an exponential inverse-link to model the scale parameter (i.e., $\sigma_i = \exp \eta_i$). Then the log-likelihood, which is given in eqs. (6.5) and (6.6), is

$$l_\sigma = -\sum_{i=1}^{N} \log \sigma_i - (1-k) \sum_{i=1}^{N} x_i - \sum_{i=1}^{N} \exp(-x_i) \tag{A.2.1}$$

where

$$x_i = -\frac{1}{k} \log\left[1 - k\left(\frac{y_i - \mu}{\sigma_i}\right)\right] \quad \text{and } \sigma_i = \exp(\eta_i) \tag{A.2.2}$$

Furthermore, we can rearrange the equation for x_i above to yield an expression that will be useful below in carrying out further simplifications:

$$k\left(\frac{y_i - \mu}{\sigma_i}\right) = 1 - \exp(-kx_i) \tag{A.2.3}$$

Using the chain rule, we can write that

$$\frac{\partial l_\sigma}{\partial \sigma_i} = -\frac{1}{\sigma_i} - \left[(1-k) - \exp(-x_i)\right] \frac{\partial x_i}{\partial \sigma_i} \tag{A.2.4}$$

Starting with the expression for x_i above, we can write that

$$\begin{aligned}
\frac{\partial x_i}{\partial \sigma_i} &= -\frac{1}{k}\left[1 - k\left(\frac{y_i - \mu}{\sigma_i}\right)\right]^{-1} \cdot \frac{\partial}{\partial \sigma_i}\left[1 - k\left(\frac{y_i - \mu}{\sigma_i}\right)\right] \\
&= -\frac{1}{k}\left[1 - k\left(\frac{y_i - \mu}{\sigma_i}\right)\right]^{-1} \frac{1}{\sigma} k\left(\frac{y_i - \mu}{\sigma}\right) \\
&= -\frac{1}{k\sigma_i} \exp(kx_i)\left[1 - \exp(-kx_i)\right] \\
&= -\frac{1}{k\sigma_i}\left[\exp(kx_i) - 1\right]
\end{aligned} \tag{A.2.5}$$

where eq. (A.2.3) has been used to arrive at the last expression. Then,

$$\begin{aligned}
\frac{\partial l_\sigma}{\partial \sigma_i} &= -\frac{1}{\sigma_i} + \frac{1}{k\sigma_i}\left[\exp(kx_i) - 1\right]\left[(1-k) - \exp(-x_i)\right] \\
&\vdots \\
&= -\frac{1}{k\sigma_i}(P_i + Q_i)
\end{aligned} \tag{A.2.6}$$

where

$$P_i = 1 - \exp(-x_i) \quad \text{and} \tag{A.2.7}$$

$$Q_i = \exp\left[(k-1)x_i\right] - (1-k)\exp(kx_i) \tag{A.2.8}$$

which is in the same form as that given in the NERC *Flood Studies Report*[8, p. 93]. Because we are using an exponential link, $\partial\sigma_i/\partial\eta_i = \exp(\eta_i)$, the derivative of the log-likelihood with respect to η_i is given by

$$\frac{\partial l_\sigma}{\partial \eta_i} = \frac{\partial l_\sigma}{\partial \sigma_i} \cdot \frac{\partial \sigma_i}{\partial \eta_i} = -\frac{1}{k\exp(\eta_i)}\left(P_i + Q_i\right)\exp(\eta_i) \tag{A.2.9}$$

where σ_i is replaced by $\exp(\eta_i)$ where it appears in P_i and Q_i.

The second derivative is somewhat messier, so let's proceed as follows to facilitate writing an R function. Using the chain rule, we can write that

$$\frac{\partial^2 l_\sigma}{\partial \eta_i^2} = \frac{\partial l_\sigma}{\partial \sigma_i} \cdot \frac{\partial \sigma_i^2}{\partial \eta_i^2} + \frac{\partial^2 l_\sigma}{\partial \sigma_i^2} \cdot \left(\frac{\partial \sigma_i}{\partial \eta_i}\right)^2 \tag{A.2.10}$$

The exponential inverse-link means that its derivatives are simple; furthermore, we have already derived an expression for $\partial l_\sigma/\partial\sigma_i$, so we need only calculate $\partial^2 l_\sigma/\partial\sigma_i^2$. Using the notation of eqs. (A.2.6), (A.2.7), and (A.2.8), and the chain rule,

$$\begin{aligned}
\frac{\partial^2 l_\sigma}{\partial \sigma_i^2} &= (P_i + Q_i) \cdot \frac{\partial}{\partial \sigma_i}\left(-\frac{1}{k\sigma_i}\right) - \frac{1}{k\sigma_i} \cdot \frac{\partial}{\partial \sigma_i}(P_i + Q_i) \\
&= \frac{1}{\sigma_i} \cdot \frac{1}{k\sigma_i}(P_i + Q_i) - \\
&\quad \frac{1}{k\sigma_i}\frac{\partial}{\partial x_i}\left[1 - \exp(-x_i) + \exp\left[(k-1)x_i\right] - (1-k)\exp(kx_i)\right] \cdot \frac{\partial x_i}{\partial \sigma_i} \\
&= -\frac{1}{\sigma_i}\frac{\partial l_\sigma}{\partial \sigma_i} - \\
&\quad \frac{1}{k\sigma_i}\left[\exp(-x_i) + (k-1)\exp\left[(k-1)x_i\right] - k(1-k)\exp(kx_i)\right] \cdot \frac{\partial x_i}{\partial \sigma_i} \\
&= \vdots \\
&= -\frac{1}{\sigma_i}\frac{\partial l_\sigma}{\partial \sigma_i} - \left[\frac{1}{\sigma_i}Q_i + \exp(-x_i) \cdot \frac{\partial x_i}{\partial \sigma_i}\right]\frac{\partial x_i}{\partial \sigma_i}
\end{aligned} \tag{A.2.11}$$

Eq. (A.2.11) expresses the second derivative of the log-likelihood with respect to the scale in terms of quantities that will have already been calculated to obtain the first derivative, and so it is straightforward to implement. Once we have calculated $\partial^2 l_\mu/\partial\sigma_i^2$ we simply substitute it into eq. (A.2.10) along with the other quantities to obtain $\partial^2 l_\mu/\partial\eta_i^2$.

B AUXILIARY R FUNCTIONS FOR IMPLEMENTING HGENGINE

The functions outlined here have been written quickly to illustrate the use of `HGengine`. They are prototypes only, and so are neither completely general nor particularly elegant! I have written two separate sets of functions, one for use when modelling the mean, the other when modelling the scale parameter.

B.1 Modelling the Mean

B.1.1 `LE.mu`

```
> # This is the function that is assigned to the argument 'Le' in the
> # function 'HGengine'. It is basically a wrapper for the function
> # 'GEV.lik.mu_eta'.
> `LE.mu` <- function(eta, data, events, weights, nuisparam)
+   {
+     Sigma <- nuisparam[1]
+     K <- nuisparam[2]
+     z <- GEV.lik.mu_eta(data, eta, Sigma, K)
+     z
+   }
```

B.1.2 `GEV.lik.mu_eta`

```
> # Calculates the log likelihood as a function of the linear predictor
> # eta when the mean is modelled as a function of eta. Note that
> # eta can be either a vector or a scalar here.
> `GEV.lik.mu_eta` <-
+   function(x, eta, sigma, k, muFUN = "I"){
+     muFUN <- match.fun(muFUN)
+     mu <- muFUN(eta)
+     N <- length(x)
+     y <- 1 - k * (x - mu)/sigma
+     if(any(y <= 0) || any(sigma <= 0))
+       return(-1e+06)
+     y <- -(1/k) * log(y)
+     # It's unlikely that BOTH mu and sigma will be vectors so we could
+     # get rid of the condition below:
+     if(length(sigma) == 1)
+       z <- -(N * log(sigma) + (1 - k) * sum(y) + sum(exp(-y)))
+     else z <- -(sum(log(sigma)) + (1 - k) * sum(y) + sum(exp(-y)))
+     attributes(z) <- NULL
+     z
+   }
```

B.1.3 HGSC.mu

```
> HGSC.mu <- function(eta, data, event,
+     weights, scale, ...) {
+     gev.lik.eta <- function(z, dat,
+         Eta) {
+         Sigma <- z[1]
+         K <- z[2]
+         eta <- Eta
+         x <- dat
+         z <- GEV.lik.mu_eta(x, eta,
+             Sigma, k = K)
+         z
+     }
+     init <- scale
+     res <- optim(init, gev.lik.eta,
+         dat = data, Eta = eta, control = list(fnscale = -1,
+             alpha = 0.25, beta = 0.5,
+             gamma = 1.1))
+     res$par
+ }
```

B.1.4 DLDE.mu

```
> # This is the function that is assigned to the argument 'DLde' in the
> # function 'HGengine'. It is a wrapper for 'dGEV.lik.dmu_eta'.
> `DLDE.mu` <- function(eta, data, events, weights, nuisparam)
+   {
+     Sigma <- nuisparam[1]
+     K <- nuisparam[2]
+     z <- dGEV.lik.dmu_eta(data, eta, Sigma, K)
+     z
+   }
```

B.1.5 dGEV.lik.dmu_eta

```
> # This function calculates dl/deta; for the identity link, dl/deta and
> # dl/dmu are identical, but not in general. If the link is not the
> # identity link, the user will have to specify dmu.deta.FUN here.
> `dGEV.lik.dmu_eta` <- function(x, eta, sigma, k,
+                                muFUN = "I",
+                                dmu.deta.FUN = function(x){1}){
+   muFUN <- match.fun(muFUN)
+   dmu.deta.FUN = match.fun(dmu.deta.FUN)
+   mu <- muFUN(eta)
+   z <- dGEV.lik.dmu(x, mu, sigma, k) * dmu.deta.FUN(eta)
+   z
+ }
```

B.1.6 dGEV.lik.dmu

```
> # Function to calculate dl/dmu
> `dGEV.lik.dmu` <- function(x, mu, sigma, k){
+   y <- 1 - k * (x - mu)/sigma
+   #
+   if(any(y <= 0) || any(sigma <= 0))
+     return(-1e+06)
+   #
+   y <- -(1/k) * log(y)
+   Q <- exp((k - 1) * y) - (1 - k) * exp(k * y)
+   z <- -Q/sigma
+   attributes(z) <- NULL
+   z
+ }
```

B.1.7 D2LDE.mu

```
> # This is the function that is assigned to the argument 'D2Lde' in the
> # function 'HGengine'. It is a wrapper for 'd2GEV.lik.dmu_eta2'.
> `D2LDE.mu` <- function(eta, data, events, weights, nuisparam)
+   {
+     Sigma <- nuisparam[1]
+     K <- nuisparam[2]
+     z <- d2GEV.lik.dmu_eta2(data, eta, Sigma, K)
+     z
+   }
```

B.1.8 d2GEV.lik.dmu_eta2

```
> # Calculates the second derivative of the log-likelihood with respect
> # to eta.  The derivatives of the inverse-link function have to be
> # entered here manually. This also calls functions for the first and
> # second derivatives of the log-likelihood with respect to mu.
> `d2GEV.lik.dmu_eta2` <-
+   function(x, eta, sigma, k,
+            dl.dmu.FUN = "dGEV.lik.dmu",
+            d2l.dmu2.FUN = "d2GEV.lik.dmu2",
+            muFUN = "I",
+            dmu.deta.FUN = function(x){1},
+            d2mu.deta2.FUN = function(x){0})
+ {
+   dl.dmu.FUN <- match.fun(dl.dmu.FUN)
+   d2l.dmu2.FUN <- match.fun(d2l.dmu2.FUN)
+   muFUN = match.fun(muFUN)
+   dmu.deta.FUN <- match.fun(dmu.deta.FUN)
+   d2mu.deta2.FUN <- match.fun(d2mu.deta2.FUN)
+
+   mu <- muFUN(eta)
+   z <- dl.dmu.FUN(x, mu, sigma, k) * d2mu.deta2.FUN(eta) +
+     d2l.dmu2.FUN(x, mu, sigma, k) * (dmu.deta.FUN(eta))^2
+   attributes(z) <- NULL
+   z
+ }
```

B.1.9 d2GEV.lik.dmu2

```
> # This calculates the second derivative of the GEV log likelihood wrt
> # mu. It requires a vector (mu) of mean values.
> `d2GEV.lik.dmu2` <-
+   function(x, mu, sigma, k){
+     y <- 1 - k * (x - mu)/sigma
+     #
+     if(any(y <= 0) || any(sigma <= 0))
+       return(-1e+06)
+     #
+     y <- -(1/k) * log(y)
+     z <- k * exp(k * y) + exp((k - 1) * y)
+     z <- -z * (1 - k) * exp(k * y)/(sigma^2)
+     attributes(z) <- NULL
+     z
+   }
```

B.2 *Modelling the Scale*

The functions listed here follow the same pattern as those written when modelling the mean, so this subsection provides only listings with few comments.

B.2.1 `LE.sigma`

```
> `LE.sigma` <- function(eta, data, events, weights, nuisparam)
+   {
+     Mu <- nuisparam[1]
+     K <- nuisparam[2]
+     z <- GEV.lik.sigma_eta(data, Mu, eta, K)
+     z
+   }
```

B.2.2 `GEV.lik.sigma_eta`

```
> `GEV.lik.sigma_eta` <-
+   function(x, mu, eta, k, sigmaFUN = "exp"){
+     sigmaFUN <- match.fun(sigmaFUN)
+     sigma <- sigmaFUN(eta)
+     N <- length(x)
+     y <- 1 - k * (x - mu)/sigma
+     if(any(y <= 0) || any(sigma <= 0))
+       return(-1e+06)
+     y <- -(1/k) * log(y)
+     if(length(sigma) == 1)
+       z <- -(N * log(sigma) + (1 - k) * sum(y) + sum(exp(-y)))
+     else z <- -(sum(log(sigma)) + (1 - k) * sum(y) + sum(exp(-y)))
+     attributes(z) <- NULL
+     z
+   }
```

B.2.3 `HGSC.sigma`

```
> `HGSC.sigma` <- function(eta, data, event, weights, scale, ...)
+   {
+     gev.lik.eta <- function(z, dat, Eta)
+       {
+         Mu <- z[1]
+         K <- z[2]
+         eta <- Eta
+         x <- dat
+         z <- GEV.lik.sigma_eta(x, Mu, eta, k = K)
+         z
+       }
+
+     #in2 <- sqrt(6 * var(data))/pi
+     #in1 <- mean(data) - 0.57722 * in2
+     #init <- c(in1, 0.1)
+     init <- scale
+     #cat("initial values are", init, "\n")
+
+     res <- optim(init, gev.lik.eta, dat = data, Eta = eta,
+                  control = list(fnscale = -1))
+                       #alpha = 0.25, beta = 0.5, gamma = 1.1))
+     res$par
+   }
```

B.2.4 DLDE.sigma

```
> `DLDE.sigma` <- function(eta, data, events, weights, nuisparam)
+   {
+     Mu <- nuisparam[1]
+     K <- nuisparam[2]
+     z <- dGEV.lik.dsigma_eta(data, Mu, eta, K)
+     z
+   }
```

B.2.5 dGEV.lik.dsigma_eta

```
> `dGEV.lik.dsigma_eta` <- function(x, mu, eta, k,
+                            sigmaFUN = "exp",
+                            dsig.deta.FUN = "exp"){
+   sigmaFUN <- match.fun(sigmaFUN)
+   dsig.deta.FUN = match.fun(dsig.deta.FUN)
+   sigma <- sigmaFUN(eta)
+   z <- dGEV.lik.dsigma(x, mu, sigma, k) * dsig.deta.FUN(eta)
+   z
+ }
```

B.2.6 dGEV.lik.dsigma

```
> `dGEV.lik.dsigma` <- function(x, mu, sigma, k){
+   ## This and calculates the derivative of the GEV log likelihood wrt
+   ## sigma.  NB This requires a *vector* (sigma) of scale values for
+   ## each observation, and it returns a vector of individual log
+   ## likelihood values.  For example, if all the values in sigma are
+   ## the same, summing the result gives the value of the derivative
+   ## calculated as if sigma were a constant scalar.
+   y <- 1 - k * (x - mu)/sigma
+   #
+   if(any(y <= 0) || any(sigma <= 0))
+     return(-1e+06)
+   #
+   y <- -(1/k) * log(y)
+   P <- 1 - exp(-y)
+   Q <- exp((k - 1) * y) - (1 - k) * exp(k * y)
+   z <- -(P + Q)/(sigma * k)
+   attributes(z) <- NULL
+   z
+ }
```

B.2.7 D2LDE.sigma

```
> `D2LDE.sigma` <- function(eta, data, events, weights, nuisparam)
+   {
+     Mu <- nuisparam[1]
+     K <- nuisparam[2]
+     z <- d2GEV.lik.dsigma_eta2(data, Mu, eta, K)
+     z
+   }
```

B.2.8 d2GEV.lik.dsigma_eta2

```
> `d2GEV.lik.dsigma_eta2` <-
+   function(x, mu, eta, k,
+            dl.dsigma.FUN = "dGEV.lik.dsigma",
+            d2l.dsigma2.FUN = "d2GEV.lik.dsigma2",
+            sigmaFUN = "exp",
+            dsig.deta.FUN = "exp",
+            d2sig.deta2.FUN = "exp"){
+     dl.dsigma.FUN <- match.fun(dl.dsigma.FUN)
+     d2l.dsigma2.FUN <- match.fun(d2l.dsigma2.FUN)
+     sigmaFUN = match.fun(sigmaFUN)
+     dsig.deta.FUN <- match.fun(dsig.deta.FUN)
+     d2sig.deta2.FUN <- match.fun(d2sig.deta2.FUN)
+
+     sigma <- sigmaFUN(eta)
+     z <- dl.dsigma.FUN(x, mu, sigma, k) * d2sig.deta2.FUN(eta) +
+       d2l.dsigma2.FUN(x, mu, sigma, k) * (dsig.deta.FUN(eta))^2
+     attributes(z) <- NULL
+     z
+   }
```

B.2.9 d2GEV.lik.dsigma2

```
> `d2GEV.lik.dsigma2` <-
+   function(x, mu, sigma, k){
+     ## This and calculates the second derivative of the GEV log
+     ## likelihood wrt sigma. As above, this requires a vector (sigma)
+     ## of scale values.
+     y <- 1 - k * (x - mu)/sigma
+     #
+     if(any(y <= 0) || any(sigma <= 0))
+       return(-1e+06)
+     #
+     y <- -(1/k) * log(y)
+     z <- exp(-y) + (k - 1) * exp((k - 1) * y) -
+       k * (1 - k) * exp(k * y)
+     z <- z * (exp(k * y) - 1)/(sigma * k)^2 -
+       dGEV.lik.dsigma(x, mu, sigma, k)/sigma
+     attributes(z) <- NULL
+     z
+   }
```