Univariate Threshold Excess Modelling of Clinical Laboratory Safety Data using R

Harry Southworth and Janet E. Heffernan

August 7, 2012

1 Introduction

This document illustrates the use of the `texmex` package, [7] for performing extreme value analysis of some clinical safety laboratory data in `R`, [5]. The full analysis is described in [6]. This package vignette focusses on the fitting of the generalized Pareto distribution (GPD) modelling of the marginal variables. A separate vignette will examine the conditional multvariate extreme value modelling, which appears in [8].

To cite this vignette, refer to Vignette name: `texmex1d` and use the package citation:

```
To cite package 'texmex' in publications use:

  Harry Southworth and Janet E. Heffernan (2012). texmex: Threshold
  exceedences and multivariate extremes. R package version 1.3.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {texmex: Threshold exceedences and multivariate extremes},
    author = {Harry Southworth and Janet E. Heffernan},
    year = {2012},
    note = {R package version 1.3},
  }

ATTENTION: This citation information has been auto-generated from the
package DESCRIPTION file and may need manual editing, see
'help("citation")' .
```

Please note that this is not intended to be an R tutorial. For that, you will have to look elsewhere.

1.1 Preliminaries

Before doing anything else, you will need to install the `texmex` package. Depending on your installation of R, this can be done using the `install.packages` command in R, or by downloading the package from CRAN and installing it from a local archive.

Once you've got `texmex` installed, use the `library` command to make the package available to the current session. This vignette also makes use of the `rlm` function from the `MASS` package (see [9]) and trellis plots from the `lattice` package.

```
> library(texmex)
> library(MASS)
> library(lattice)
> palette(c("black","purple","cyan","orange"))
> set.seed(20111011)
```

The final command sets the random seed so that the results in this vignette may be reproduced exactly.

1.2 `texmex`

The `texmex` package for R was written by Harry Southworth and Janet E. Heffernan. The work was funded by AstraZeneca.

Some considerable effort has been made to ensure that the package does what it ought to, and to this end over 400 tests (at the current count) are built into the package. The tests compare the output of functions in `texmex` with published results and, where no published results were available, with output from independently written code. There are also logical tests (e.g. of use of informative prior distributions) and tests of the structures of objects.

To test the package functionality at any time, use the `RUnit` package [1]. With `RUnit` and `texmex` attached, you can perform all the tests, and view a report describing the results, as follows.

```
> library(RUnit)
> pdf("texmexValidation.pdf")
> res <- validate.texmex()
> dev.off()
> printHTMLProtocol(res, "texmexValidation.html")
```

Many of the plots produced in the `texmexValidation.pdf` file reproduce figures appearing in published material, and are intended for comparison against these figures. The validation versions of such plots produced by `texmex` are labelled with the target figure references.

Due to the large number of tests performed, including tests of bootstrap and MCMC procedures, `validate.texmex` takes quite a while to run.

1.3 Data

The dataset used in this example analysis is contained in the `texmex` package. The dataset is called `liver` and we can look at the first few lines and a summary as follows:

```
> head(liver)
```

```
  ALP.B ALT.B AST.B  TBL.B ALP.M ALT.M AST.M  TBL.M dose
1    80    13    14 12.654    87    22    22 23.085    A
2    37    15    16  6.498    37    25    23  8.037    A
3    52    10    13  4.788    55    10    13  6.498    A
4    36    13    13  6.840    35    11    12  8.037    A
5    39    18    12 14.364    37    21    15 16.758    A
6    48     8    13  6.156    50     8    13  5.985    A
```

```
> summary(liver)
```

```
     ALP.B            ALT.B            AST.B            TBL.B
 Min.   : 15.00   Min.   :  4.00   Min.   :  5.00   Min.   : 2.736
 1st Qu.: 44.25   1st Qu.: 11.00   1st Qu.: 13.00   1st Qu.: 7.011
 Median : 53.00   Median : 14.00   Median : 15.50   Median : 8.721
 Mean   : 55.21   Mean   : 15.67   Mean   : 16.25   Mean   : 9.475
 3rd Qu.: 64.00   3rd Qu.: 18.00   3rd Qu.: 18.00   3rd Qu.:10.944
 Max.   :129.00   Max.   :198.00   Max.   :104.00   Max.   :27.531
     ALP.M            ALT.M            AST.M            TBL.M        dose
 Min.   :  1.00   Min.   :  2.00   Min.   :  6.00   Min.   : 3.249   A:152
 1st Qu.: 48.00   1st Qu.: 13.00   1st Qu.: 15.00   1st Qu.: 7.695   B:148
 Median : 58.00   Median : 17.00   Median : 18.00   Median : 9.576   C:148
 Mean   : 61.49   Mean   : 20.83   Mean   : 19.21   Mean   :10.691   D:158
 3rd Qu.: 70.75   3rd Qu.: 24.00   3rd Qu.: 21.00   3rd Qu.:12.825
 Max.   :341.00   Max.   :324.00   Max.   :250.00   Max.   :42.750
```

The response variables are

ALT alanine aminotransferase

AST aspartate aminotransferase

ALP alkaline phosphatase

TBL total bilirubin

The variables suffixed by `.B` are data measured at baseline (prior to treatment); an `.M` indicates post-baseline measurement (on treatment). In this study, there was only one post-baseline visit. In general, in trials which have more than one post-baseline visit it is natural to use the maximum post-baseline value for each individual.

The doses were equally spaced on a logarithmic scale (i.e. dose D is twice dose C, dose C is twice dose B, and dose B is twice dose A). Later in the analysis,

we will do some modelling with log(dose) appearing in the linear predictor, so we take a copy of the dataset and create a new variable now:

```
> liver <- liver
> liver$ndose <- as.numeric(liver$dose)
> table(liver$ndose)

  1   2   3   4
152 148 148 158
```

Before proceeding, first note that there are two small outliers in $ALP.M$: two cases with $ALP.M = 1$. Discussion with the study physician led to the conclusion that the lowest value of ALP in the dataset was impossible, and since it is large values in which we are interested anyway, we can remove it (leaving it in complicates plotting later in the analysis, but does not affect any of the results).

```
> liver <- liver[liver$ALP.M > 1,]
```

The variables in the dataset all relate to the liver. Biologically, the understanding is that liver cells release ALT and AST as they die. If a sufficient amount of the liver is destroyed that it can no longer function properly, then it ceases to be able to clear bilirubin and so bilirubin (TBL) goes up. The situation is complicated by the fact that ALT and AST can also arise from other sources (e.g. muscle), and that ALP can rise in response to a blockage in the liver.

Given the biology, ALT and AST are likely to rise early on in any drug induced liver damage and initial focus should be on those. ALT and AST are closely correlated in this example and in what follows we focus mostly on ALT.

2 Elimination of baseline effect

Since the four lab variables were measured at baseline and at a later date in the same patients, we might reasonably expect the two values to be related. Figure 1 scatterplots (on the log scale) reveal this to be true.

We can eliminate the baseline effect (and therefore reduce the variance) using a linear model. Since we have no reason to suppose the data to be normally distributed, and since Figure 1 reveals outliers not consistent with a Gaussian distribution, we will use a robust regression approach. (In general, lab safety data should never be assumed to be normally distributed.)

Following Maronna et al [4] (page 144), we prefer an MM-estimated regression line with Gaussian efficiency set to 85% and bisquare weight functions. This model can be estimated using the `rlm` function in the `MASS` package. By default, `rlm` uses 95% Gaussian efficiency, and we can obtain the required 85% by passing argument `c = 3.44` (see [4], page 30) into the function.

```
> rmod <- rlm(log(ALT.M) ~ log(ALT.B) + ndose, data=liver, c=3.44, method="MM")
> summary(rmod)
```

```
> print(
+ xyplot(ALT.M ~ ALT.B | dose, data=liver,
+        as.table = TRUE,
+        scales=list(log=2))
+ )
```

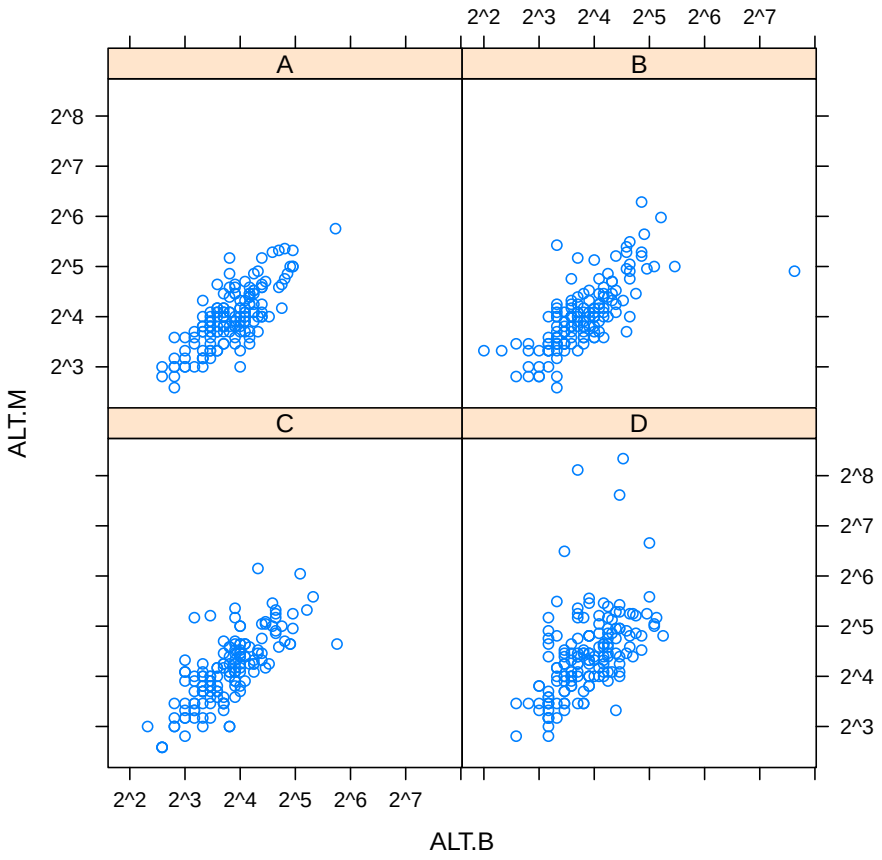
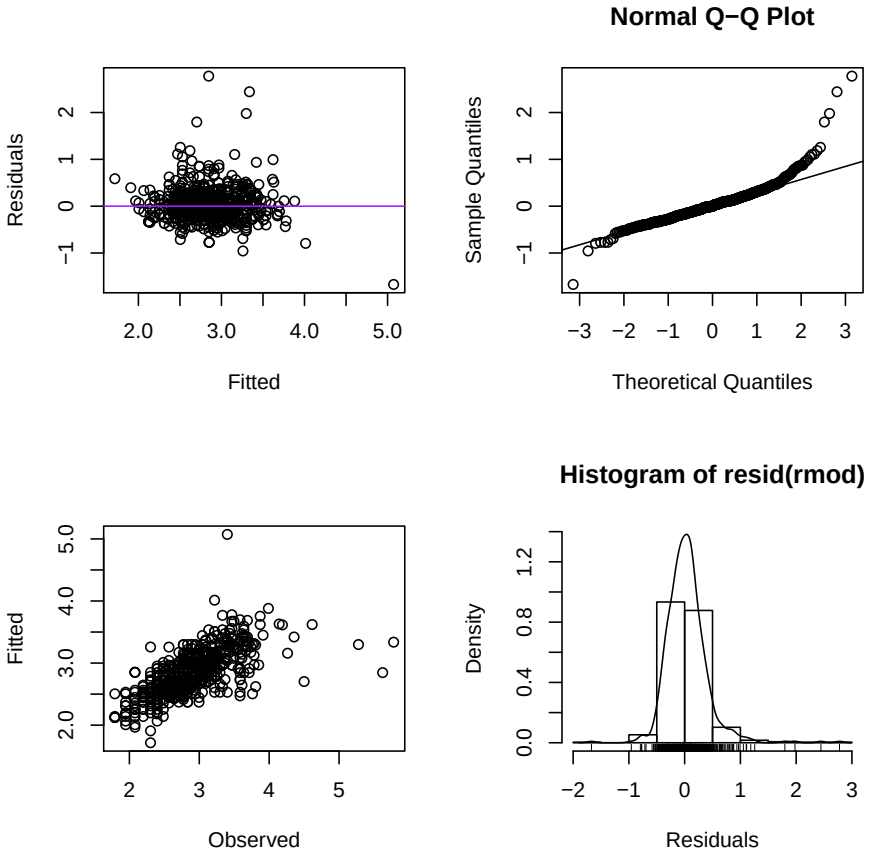


Figure 1: Scatterplots of post-baseline versus baseline ALT.

```
Call: rlm(formula = log(ALT.M) ~ log(ALT.B) + ndose, data = liver,
    c = 3.44, method = "MM")
Residuals:
      Min        1Q    Median        3Q       Max
-1.671456 -0.176330  0.002382  0.199886  2.777716

Coefficients:
            Value   Std. Error t value
(Intercept)  0.4078  0.0845     4.8241
log(ALT.B)   0.8602  0.0300    28.6478
ndose        0.0581  0.0109     5.3433

Residual standard error: 0.2848 on 601 degrees of freedom
```

```
> par(mfrow=c(2, 2))
> plot(fitted(rmod), resid(rmod), xlab="Fitted", ylab="Residuals")
> abline(h=0, col=2)
> qqnorm(resid(rmod))
> qqline(resid(rmod))
> plot(log(liver$ALT.M), fitted(rmod), xlab="Observed", ylab="Fitted")
> d <- density(resid(rmod))
> hist(resid(rmod), xlab="Residuals", prob=TRUE, ylim=range(d$y))
> lines(d)
> rug(resid(rmod))
```

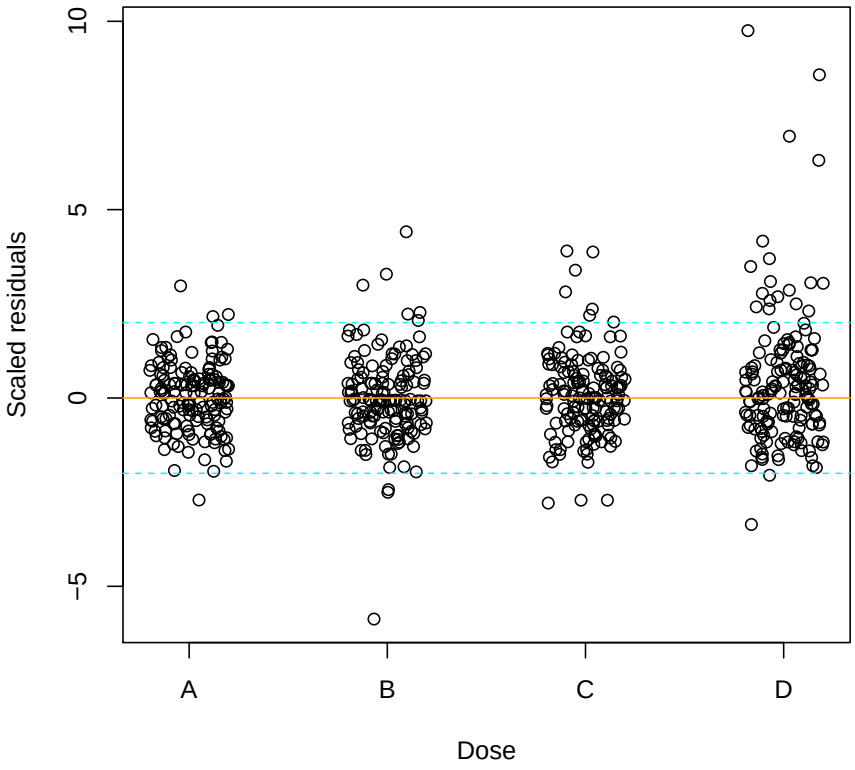


The residual plots do not give cause for concern and reveal several outliers, justifying the use of a robust method of estimation.

In practice, it would be sensible to fit a few models, possibly using `dose` as a factor and with alternative transformations of ALT. That part of the analysis is left as an exercise and we proceed on the understanding that the fitted robust linear model is adequate for eliminating the effects of baseline and dose on the central tendency of the data.

We can now obtain the residuals to be used for the extreme value modelling of ALT. We also plot scaled residuals to get a preliminary impression of any dose effect.

```
> liver$r <- resid(rmod)
> plot(jitter(liver$ndose), liver$r / rmod$s,  # Scaled residuals
+     xlab="Dose", ylab="Scaled residuals", axes=FALSE)
> box(); axis(2)
> axis(1, at=1:4, labels=LETTERS[1:4])
> abline(h=c(-2, 0, 2), lty=c(2, 1, 2), col=c(3,4,3))
```



It can be seen that there are several large outliers, out of keeping with any assumptions about normality, and that dose D appears to be associated with more, and larger, outliers.

3 Generalized Pareto distribution models for ALT

We now proceed to fit, evaluate, choose between, and ultimately make predictions from generalized Pareto distribution (GPD) models for the residuals of the ALT data obtained in Section 2.

3.1 Extreme value modelling and asymptotic motivation for the GPD

Extreme value statistical models are unusual among statistical models in that they are often required for extrapolation beyond levels observed in the data. As statisticians, we are told that extrapolation from statistical models is perilous:

our models can only be trusted in regions where we have sufficient data to calibrate and check goodness of model fit. Extreme value modelling has responded to a demand for extrapolation beyond this safe region. Since we can no longer rely on data as a check on our model's suitability, extreme value statisticians turn to mathematical arguments to bolster their confidence in their extrapolation. These arguments provide a justification for the use of a particular type of model to describe tail behaviour of random variables.

This is not a tutorial in Extreme Value Theory, for which we refer the reader to [2], which describes a range of methods for modelling the statistical properties of sample maxima, threshold excesses, extremes of dependent series and other aspects of tail behaviour.

The `texmex` package focusses on the use of *threshold exceedances*. Specifically, we fit the generalised Pareto Distribution, GPD(σ, ξ) [3] to data points in excess of suitably chosen thresholds. The GPD has distribution function

$$F_{>u}(x) = 1 - \left\{1 + \xi\left(\frac{x-u}{\sigma}\right)\right\}^{-1/\xi} \quad \text{for } x > u, \tag{1}$$

where u is the threshold for fitting and $\sigma > 0$ and $\xi \in \mathbb{R}$ are the scale and shape parameters respectively. This is the conditional distributon of observations given that the observations exceed the fitting threshold u. The range of possible values taken by realisations from the GPD depends on the parameter values, with the distribution having a finite upper end point (short tailed) if the shape parameter is negative ($u < x \leq u - \sigma/\xi$ if $\xi < 0$) and an infinite tail otherwise $u < x < \infty$ if $\xi \geq 0$. When $\xi = 0$, the GPD corresponds exactly to the Exponential distribution.

Extreme value theory tells us that under appropriate normalisation of the threshold excesses, as the threshold u tends to the distributional upper endpoint, the limiting distribution of the excesses must fall in the generalised Pareto family of distributions (given certain conditions concerning non-degeneracy of the limit distribution and smoothness of the distribution of the original variable). So whatever the original distribution of the measurements, provided we choose an appropriately high threshold, the distribution of values exceeding that threshold should be well approximated by a GPD. Diagnostic tools to aid the choice of suitable threshold are standard, and are described shortly – see also [2].

3.2 Parameterization

The usual parameterization of the GPD (as in Equation (1)) is in terms of its scale paramter σ and shape parameter ξ. There are, however, good reasons for reparameterizing in terms of $\phi = \log \sigma$:

- Experience has demonstrated that the numerical algorithms used for optimizing the log-likelihood tend to converge more reliably when working with ϕ;

- When including covariates in the model we are faced with the constraint that $\sigma > 0$ and working with a linear predictor specified in terms of $\phi = \log \sigma$ guarantees this constraint;
- When placing prior distributions on parameters, it is convenient to work with Gaussian distributions and ϕ is more likely to be close to Gaussian than is σ.

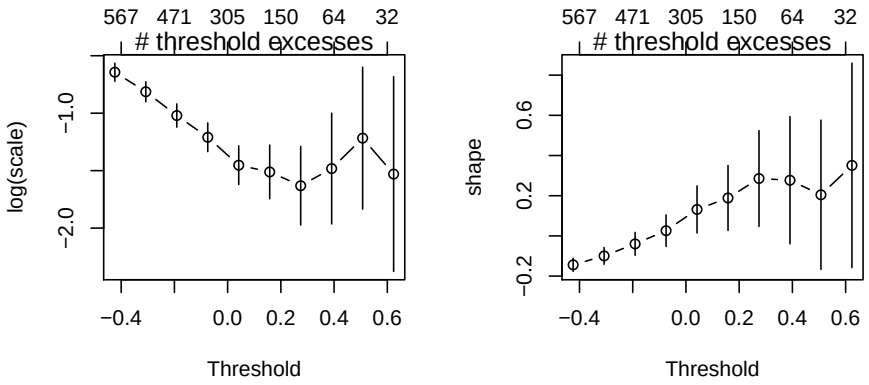
As such, some of the functions in `texmex` work with ϕ, not σ. In the case when inference is required for σ rather than ϕ, the point estimates can simply be exponentiated if maximum likelihood estimation is used. If a prior distribution is used, the point estimates are not invariant to transformation, so any transformed values should only be considered to be approximate.

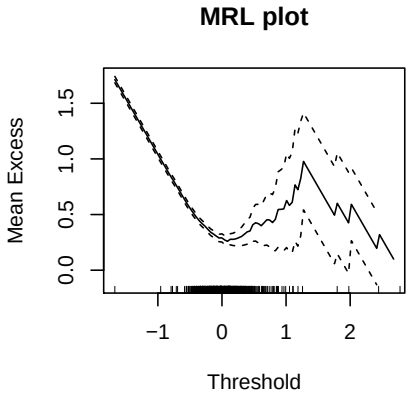
3.3 Threshold selection

GPD modelling proceeds by selecting a threshold above which the data appear to be well modelled. Standard tools for threshold selection that appear in the literature (see for example [2]) include the *mean residual life* plot, and plots of parameters estimated using a range of thresholds, *threshold stability plots*.

For a suitably chosen threshold, the mean residual life plot should be linear and the parameter estimates in threshold stability plots constant beyond the threshold (both of these requirements are assessed by taking account of sampling variability). The sign of the gradient in the linear part of the MRL plot corresponds to the sign of the shape parameter and hence indicates the shape of the tail – negative slope shows a short tailed distribution, a horizontal line (zero gradient) shows an exponential type tail and a positive slope suggests a heavy tailed distribution.

```
> par(mfrow=c(2, 2))
> gpdRangeFit(liver$r)
> mrlPlot(liver$r, main="MRL plot")
```





For our example, when fitting the GPD to the residuals from our robust regression, a threshold slightly above 0 appears to be sensible. However, we will need to do some additional diagnostics to check this. We proceed by selecting the 70^{th} percentile of the residuals as being the candidate threshold.

```
> quantile(liver$r,0.7)

     70%
0.150163
```

In many examples, we have found that the 70^{th} or quite often the 50^{th} percentile is a suitable threshold. The theory underpinning the GPD tells us that (if the underlying distribution satisfies our conditions) there has to be a threshold above which the GPD fits the data, but the theory does not specify that the threshold necessarily must be high. In our case, we are modelling residuals, and so the complete distribution will be near-symmetric and the GPD will only fit one tail. So it is unreasonable to examine thresholds lower than about the 50^{th} percentile.

3.4 Maximum penalized likelihood estimation

With small sample sizes, the GPD log-likelihood function often becomes flat and the optimiser can fail to converge. One way to overcome this is to penalize the likelihood by some function of the parameters. Experience suggests that the main problems are overcome by putting fairly modest penalties on ξ.

Thus, rather than maximize the log-likelihood $l(\phi, \xi|X)$ we maximize

$$l(\phi, \xi) - \lambda\xi^2 \tag{2}$$

for some λ.

3.4.1 Choice of λ

If we exponentiate (2), the result can be written as

$$L(\phi, \xi|X)e^{-\xi^2/2\theta^2} \tag{3}$$

in which $\theta = \sqrt{\frac{1}{2\lambda}}$. The rightmost term in (3) is proportional to a Gaussian distribution centred at 0. Thus, maximum penalized likelihood estimation has a Bayesian interpretation and corresponds to maximum a posteriori estimation.

For the GPD, $\xi = -1$ corresponds to the distribution being uniform, $\xi = 0$ corresponds to it being exponential, and $\xi = 1$ corresponds to it being so heavy-tailed that its expectation is infinite. For the kinds of data we have, $\xi = -1$ and $\xi = 1$ are implausible, and we would expect values of ξ to be fairly close to 0. This implies a prior distribution that is Gaussian with standard deviation $\theta = \frac{1}{2}$.

Since convergence issues are generally associated with ξ, we can choose a diffuse prior for ϕ, $\phi \sim N(0, 10^4)$, say.

In general, we will attempt to use MLE or penalized MLE with diffuse priors for both ϕ and ξ. Prior distribution $\xi \sim N(0, \frac{1}{4})$ independently of a diffuse prior on ϕ can be used when convergence issues arise. Such a model can be fit using

```
> pp <- list(c(0, 0), diag(c(10^4, .25)))
> pmod <- gpd(x, qu=.7, priorParameters = pp, prior="gaussian")
```

in which `priorParameters` is a list containing the mean $(0, 0)^T$ and covariance matrix of the prior Gaussian distribution.

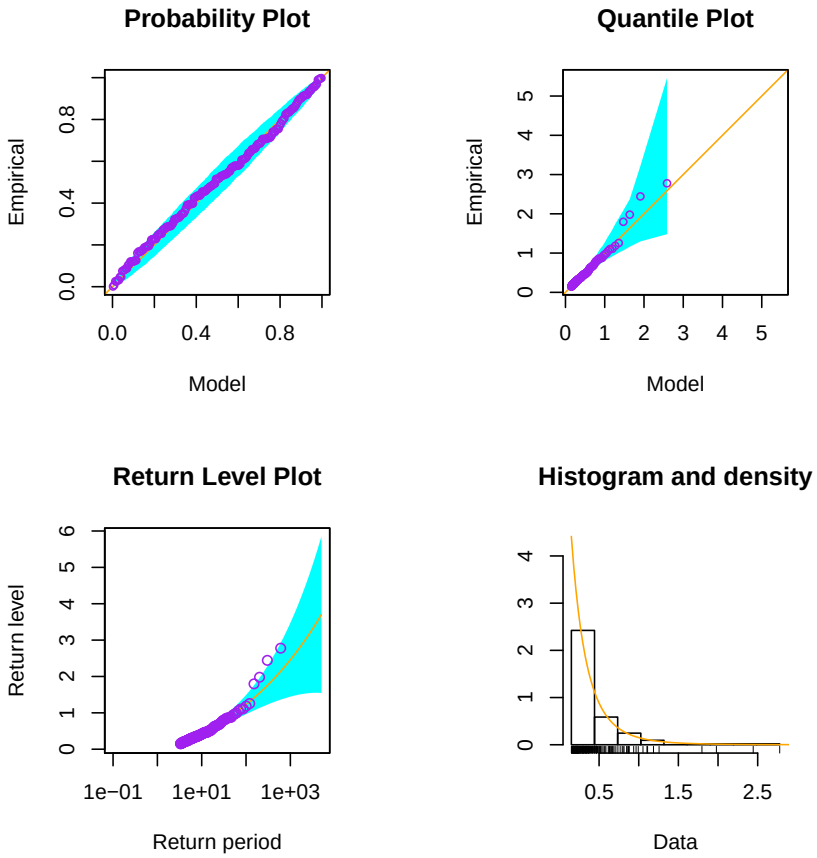
3.5 Model selection

We can fit GPD models with covariates in ϕ, in ξ, in neither, or in both. We will first fit a simple model with no covariates and examine some diagnostic plots to see if there are any obvious problems suggesting that a higher threshold needs to be used.

```
> mod <- gpd(r, data=liver, qu=.7, penalty="none")
> mod
```

```
Call: gpd.default(y = r, data = liver, qu = 0.7, penalty = "none")

Model fit by maximum likelihood.

Convergence:                TRUE
Threshold:                0.1502
Rate of excess:               0.2997

Log-lik.                AIC
53.87512                  -103.8

Coefficients:
     Value     SE
phi  -1.48482   0.10916
xi    0.18722   0.08115
```

```
> par(mfrow=c(2, 2))
> plot(mod)
```



The shaded regions in the P-P and Q-Q plots indicate pointwise 95% tolerance intervals, based on 1000 simulated datasets. The shaded region in the return level plot shows 95% pointwise confidence intervals. The plots show no systematic departure of the data from the model at this choice of threshold, so we proceed to fit various models with covariates and compare them using AIC.

```
> mod1 <- gpd(r, data=liver, qu=.7, penalty="none", phi= ~dose, xi= ~dose)
> mod2 <- gpd(r, data=liver, qu=.7, penalty="none", phi= ~ndose, xi= ~ndose)
> AIC(mod1)

[1] -106.9167

> AIC(mod2)

[1] -114.4491
```

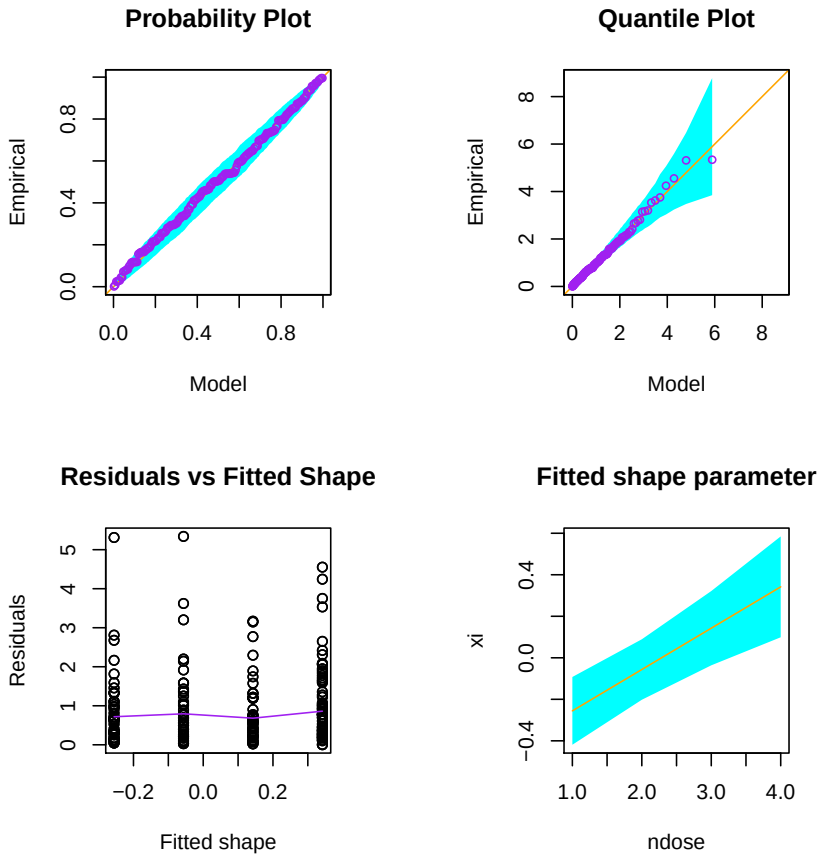
AIC is lower for `mod2` which treats `dose` as numeric variable rather than as a factor at 4 levels. So we prefer `mod2`. We proceed to try simplifications of this model:

```
> mod3 <- gpd(r, data=liver, qu=.7, penalty="none", phi= ~ndose)
> mod4 <- gpd(r, data=liver, qu=.7, penalty="none", xi= ~ndose)
> AIC(mod3); AIC(mod4); AIC(mod)

[1] -111.232

[1] -115.5584

[1] -103.7502
```

Since `mod4` has the lowest AIC we prefer that model. This is the model with ϕ constant and ξ linear in dose (on the log scale). In practice, a few more models might have been tried. Again, that part of the analysis is left as an exercise.

We now take a closer look at `mod4`.

```
> par(mfrow=c(2, 2), pty="s")
> plot(mod4)
> plot(predict(mod4,type="lp",ci.fit=TRUE),main="Fitted shape parameter")
> summary(mod4)

Call: gpd.default(y = r, data = liver, qu = 0.7, xi = ~ndose, penalty = "none")

Model fit by maximum likelihood.

Convergence:                TRUE
Threshold:                 0.15
Rate of excess:                0.3

Log-lik.                AIC
60.77922                  -116

Coefficients:
                  Value     SE        t
phi:              -1.4269    0.1088  -13.1162
xi: (Intercept)   -0.4548    0.1121   -4.0574
xi: ndose          0.1992    0.0459    4.3349

1000 simulated data sets compared against observed data QQ-plot.
Quantile of the observed MSE:  0.02
0 observations (0%) outside the 95% simulated envelope.
```



Since there is a covariate in the model the probability and quantile plots are constructed using the model residuals, which are exponential under the fitted model. We also have a plot of the residuals against the fitted parameters for any parameter that is modelled using a covariate (in this case the shape parameter ξ). A well fitting model should have homogeneity of residuals across different values of the fitted parameter. These diagnostic plots give no cause for concern.

From the estimated parameters, we see that as dose increases, so does the estimate of ξ, starting out negative for dose A (suggesting a fairly short-tailed distribution), but moving to positive at the higher doses (suggesting a heavy-tailed distribution).

3.6 Predicted return levels

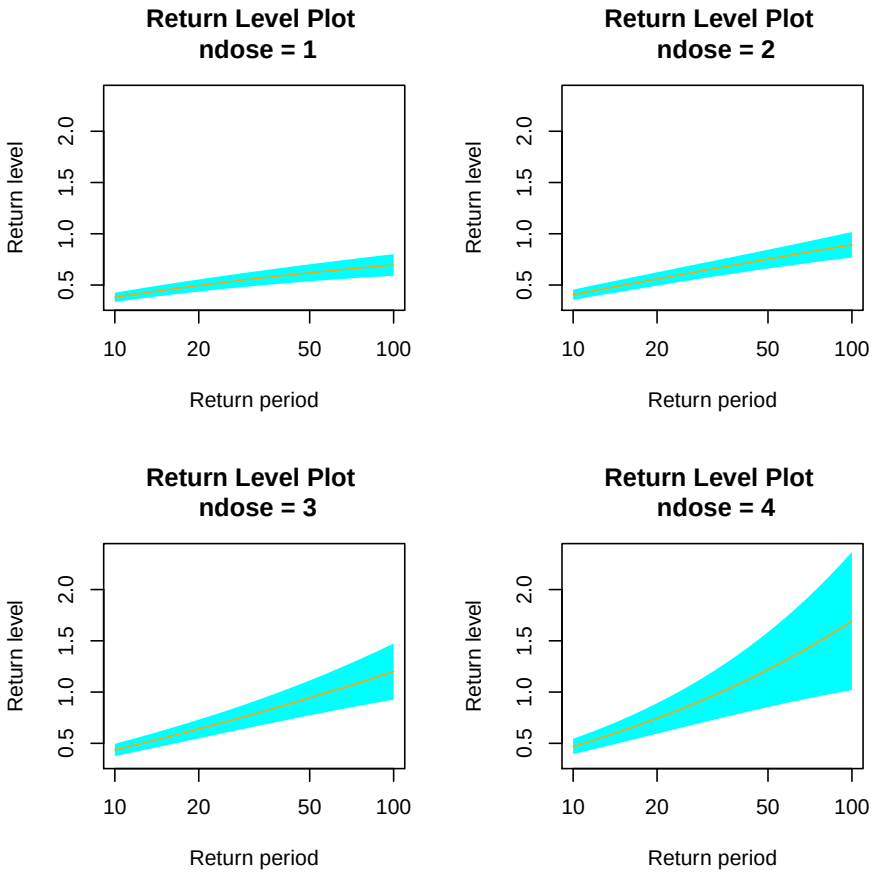
The general definition of an m-observation return level for the GPD is:

$$x_m = u + \frac{\sigma}{\xi}\{(mp)^{\xi} - 1\}. \tag{4}$$

Here p is the probability of exceeding the GPD fitting threshold u and m is a large value, so that x_m is termed the m-observation return level and represents the maximum value of x expected to be seen in m observations at a give dose.

The effect of the variable `ndose` on the GPD fitted to the residuals from our robust regression model is seen clearly when we look at return level plots for the 4 doses represented by our data:

```
> par(mfrow=c(2, 2)); plot(predict(mod4,M=10:100,ci.fit=TRUE))
```



In this plot, *Return period* is in units of *numbers of observations* and *Return level* is in the same units as the residuals to which the GPD model has been fit. For our application to the `liver` dataset, prediction of m-observation (m-patient) return levels is complicated by the relationship between baseline and post-treatment response which must be accounted for. In order to do this, we must specify a value of baseline and then the value of u in Equation (4) is calculated as the expected post-treatment value given that baseline value, plus the threshold used for fitting the GPD model to the residuals. The resulting

value of x_m is then interpreted as the m-patient return level for patients with that given baseline value.

So for our fitted model `mod4`, for a given baseline, u in equation (4) is calculated as the expected post-treatment value given that baseline plus the 70^{th} percentile of the residuals (this was the threshold used for fitting the GPD). The m-patient return level depends on *dose* via the threshold (since the expected values from the regression depend on dose) and through ξ which is also a function of dose.

The estimated robust regression and GPD models are combined in R to obtain return level estimates as follows:

```
> calcRetLevel <- function(gpdModel,MMmodel, m,base){
+
+   newdata <- data.frame(ALT.B=rep(base,4),ndose=1:4)
+
+   ElogResponse <-predict(MMmodel,newdata)
+   logRL <- ElogResponse + predict(mod4,m,newdata)[[1]][,1]
+   ElogThresh <- ElogResponse + gpdModel$threshold
+
+   names(logRL) <- names(ElogThresh) <- LETTERS[1:4]
+
+   list(u = exp(ElogThresh), RL = exp(logRL))
+ }

> PlotMpatientRetLevel <- function(Baseline,ylim,Legend=TRUE){
+
+   m <- exp(seq(log(10),log(1000),length=20))
+   RL <- t(sapply(m,function(x,gpd,rmod,base){
+                       calcRetLevel(gpd,rmod,x,base)$RL},
+                       gpd=mod4,rmod=rmod,base=Baseline))
+   u <- calcRetLevel(mod4,rmod,m[1], Baseline)$u
+
+   plot(rep(c(10,m),4),rbind(u,RL),type="n",xlab="m",
+        ylab="m-patient return level",log="x",
+        main=paste("ALT: Baseline =",signif(Baseline,3)),ylim=ylim)
+   for(i in 1:4){
+     lines(m,RL[,i],col=i)
+     abline(h=u[i],col=i,lty=2)
+   }
+   if(Legend){
+     legend(min(m),ylim[2],
+            legend=c(paste(LETTERS[1:4],",
+                     xi =",signif(predict(mod4,type="lp")[,"xi"],3)),
+                     paste("U,",LETTERS[1:4])),
+            col=rep(1:4,2),lty=rep(1:2,each=4))
+   }
+ }
```

An illustration of this calculation is given in Figures 2 and 3. Figure 2 shows estimated m-patient return levels for a patient with baseline equal to the mean observed value of baseline. For such patients, differences in the values of the thresholds U are due to the difference in expected post-treatment ALT estimated by the MM regression model, whose coefficients depend on treatment (dose).

For small values of m, estimated m-patient return levels in Figure 2 show little additional difference between treaments above that already due to the diffences in mean post-treatment ALT captured by the robust regression model. However, as m increases and we extrapolate further into the tail of the the distribution, differences in shape parameter ξ between the different doses have greater influence on the estimated return levels. Those treatments with heavier tails (treatments C and D – the higher doses) have return level estimates that grow much faster than those corresponding to treatments with short tails (treatments A and B – the lower doses).

Figure 3 shows correponding return level estimates for patients with baseline equal to the lower and upper quartiles of the observed baseline. The absolute value of m-patient return level depends on the baseline, but the ordering between the four treatments due to the fitted GPD shape parameters is the same in each case.

```
> PlotMpatientRetLevel(mean(liver$ALT.B),ylim=c(0,200))
```

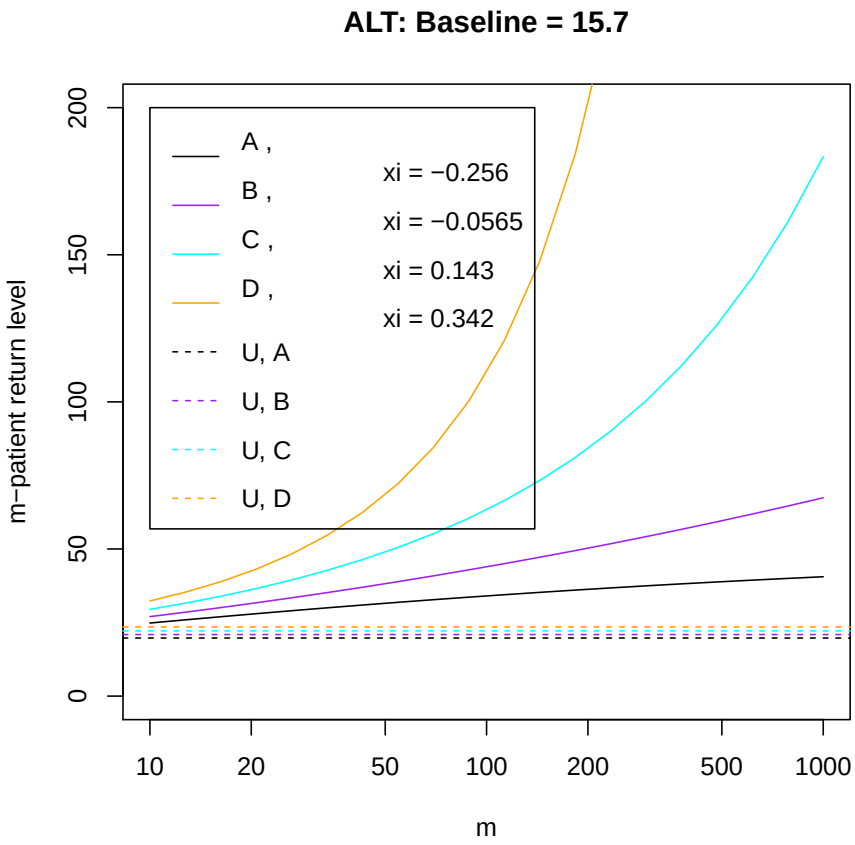


Figure 2: Point estimates of m-patient return levels under each treatment, for patients with baseline equal to the mean observed baseline.

```
> par(mfrow=c(1,2),pty="s")
> PlotMpatientRetLevel(quantile(liver$ALT.B,0.25),ylim=c(0,200),Legend=FALSE)
> PlotMpatientRetLevel(quantile(liver$ALT.B,0.75),ylim=c(0,200),Legend=FALSE)
```

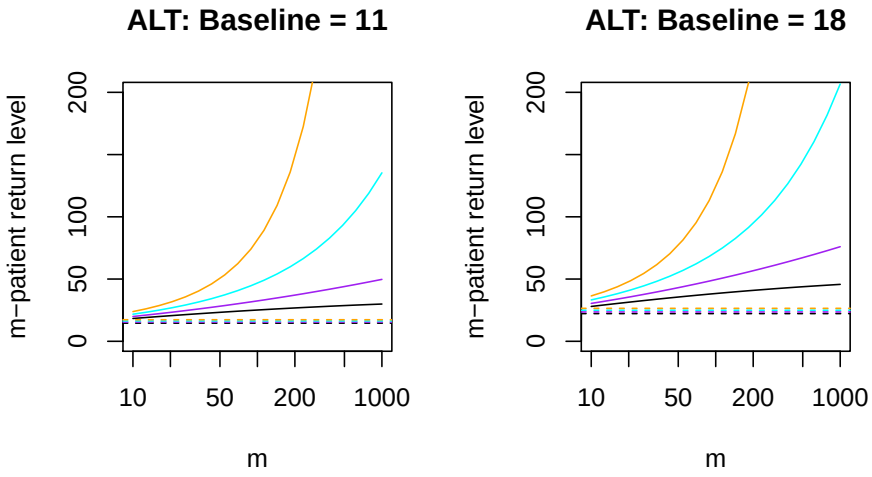


Figure 3: Point estimates of m-patient return levels under each treatment, for patients with baseline equal to the lower (left hand plot) and upper quartiles of observed baseline values. For plot legend, refer to Figure 2.

3.7 GPD parameter uncertainty

We examine briefly here Information Matrix summaries and Bootstrap estimates of GPD parameter uncertainty, before going on to use a Bayesian simulation based approach to estimation of our GPD model parameters and associated uncertainty.

3.7.1 Information matrix based approaches

When the GPD model is fit by using the default (penalized) maximum likelihood estimation, an estimate of the covariance matrix of model parameters is returned. The default procedure for estimating this covariance matrix is `cov="observed"` in which case the observed information matrix is used, as given in Appendix A of Davison and Smith [3]. The only other option is `cov = "numeric"` in which case a numerical approximation of the Hessian is used (see the help for `optim`). In some cases, particularly with small samples, the numerical approximation can be quite different from the closed form (cov="observed") result, and the value derived from the observed information should be preferred.

For our fitted model, we compare the two approaches and find that the alternative methods give almost identical estimates of the Information matrix:

```
> mod4$cov

               [,1]          [,2]          [,3]
[1,]  0.0118358625 -0.005612302 -0.0002002251
[2,] -0.0056123019  0.012566400 -0.0038710280
[3,] -0.0002002251 -0.003871028  0.0021105547

> update(mod4,cov="numeric")$cov

               [,1]          [,2]          [,3]
[1,]  0.0118353071 -0.005611724 -0.0002000668
[2,] -0.0056117236  0.012563128 -0.0038696383
[3,] -0.0002000668 -0.003869638  0.0021096777
```

For small samples, the underlying log-likelihood may be far from quadratic, and the resulting estimates of standard errors derived using either of these methods are liable to approximate poorly the true standard errors.

3.7.2 Parametric Bootstrap approach

An alternative approach to uncertainty estimation is to use a parametric bootstrap – which does capture the asymmetry of the log-likelihood surface around the maximum likelihood estimates. This is carried out for our fitted model in `texmex` as follows:

```
> boot4 <- bootgpd(mod4)
```

```
Replicate 10
Replicate 20
Replicate 30
Replicate 40
Replicate 50
Replicate 60
Replicate 70
Replicate 80
Replicate 90
Replicate 100
```

```
> summary(boot4)
```

```
bootgpd(x = mod4)
                        phi:  xi: (Intercept)  xi: ndose
Original          -1.42694281     -0.45483439 0.19915057
Bootstrap mean    -1.40542619     -0.53474312 0.22272841
Bias               0.02151662     -0.07990874 0.02357784
SD                 0.11837451      0.16503184 0.06239460
Bootstrap median -1.40657385     -0.53044590 0.21883244

Correlation:
                     phi:  xi: (Intercept)  xi: ndose
phi:              1.0000000      -0.4755323  0.1184277
xi: (Intercept) -0.4755323       1.0000000 -0.8394755
xi: ndose         0.1184277      -0.8394755  1.0000000
```

We can compare these reported standard deviations with the correponding estimates derived from the Observed Information matrix estimate – these are close although not identical, with the largest disagreement occurring for the shape parameter. This is typical behaviour of the GPD model.

```
> sqrt(diag(mod4$cov))
```

```
[1] 0.10879275 0.11209996 0.04594077
```

We can also compare the bootstrap based estimate of the parameter correlation matrix with that derived from the Observed Information matrix:

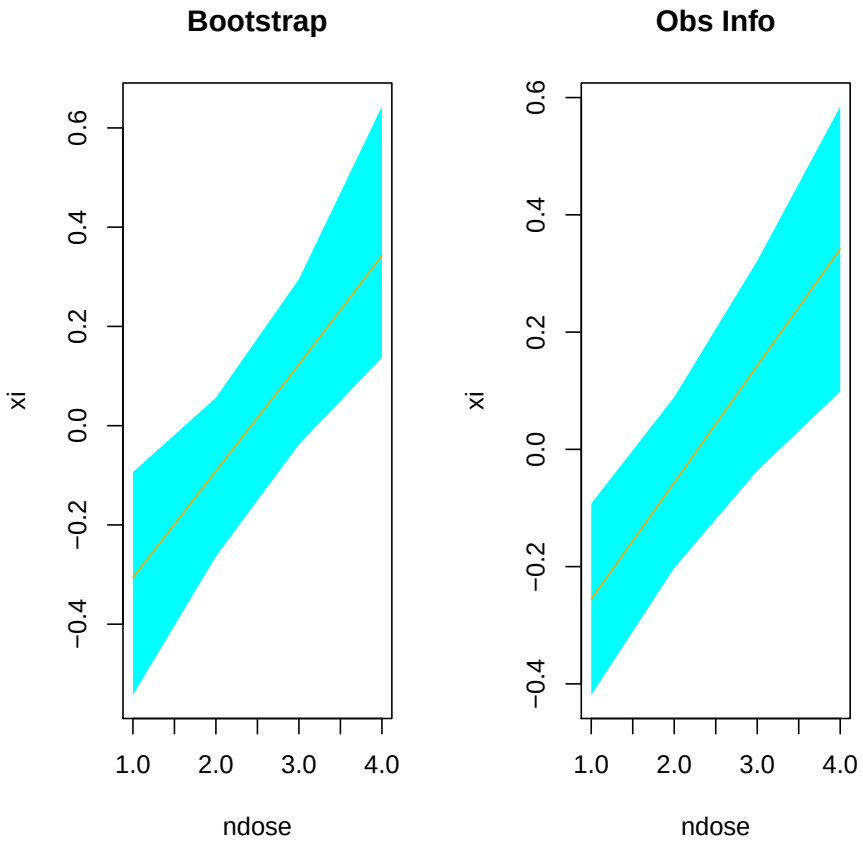
```
> cov2cor(mod4$cov)
```

```
             [,1]       [,2]        [,3]
[1,]  1.00000000 -0.4601884 -0.04006087
[2,] -0.46018836  1.0000000 -0.75166194
[3,] -0.04006087 -0.7516619  1.00000000
```

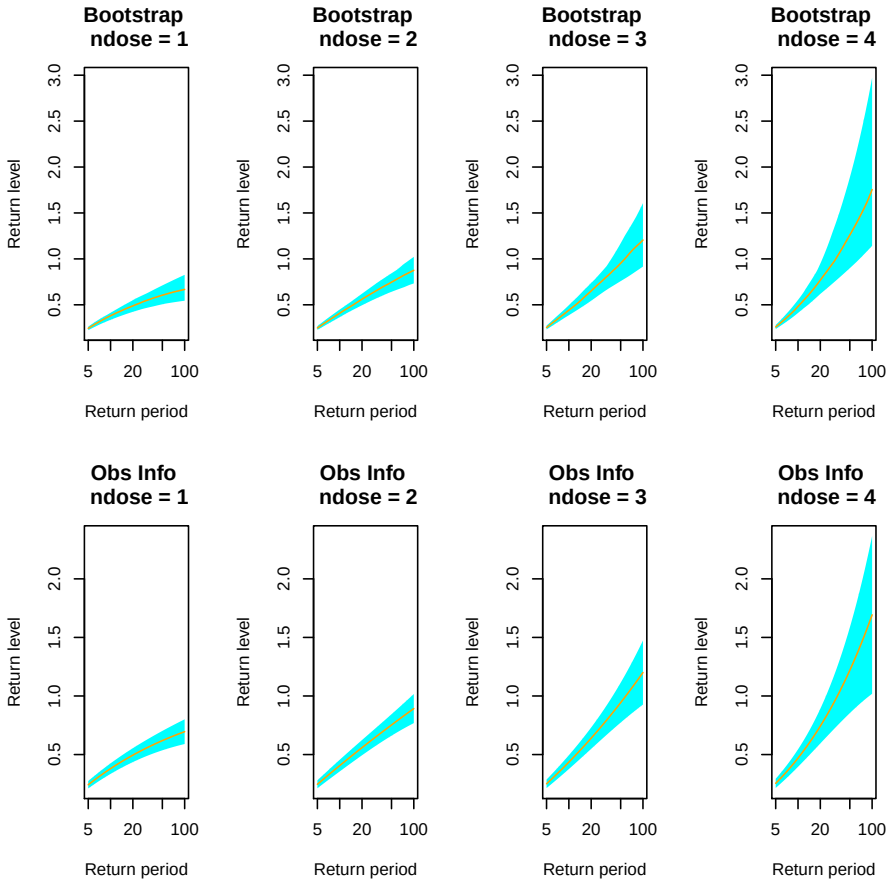
Estimates of this correlation matrix are similar although not identical, as anticipated. Focussing on the covariance matrix of the parameter estimates is

misleading and does not let us explore the asymmetric nature of the uncertainty about the parameter estimates. This can be better seen in the bootstrap based confidence intervals for the model parameters, and also particularly for return level estimates, both shown in the following plots:

```
> par(mfrow=c(1,2))
> plot(predict(boot4,type="lp",ci.fit=TRUE),main="Bootstrap")
> plot(predict(mod4,type="lp",ci.fit=TRUE),main="Obs Info")
```




```
> par(mfrow=c(2,4))
> plot(predict(boot4,M=5:100,ci.fit=TRUE),main="Bootstrap")
> plot(predict(mod4,M=5:100,ci.fit=TRUE),main="Obs Info")
```



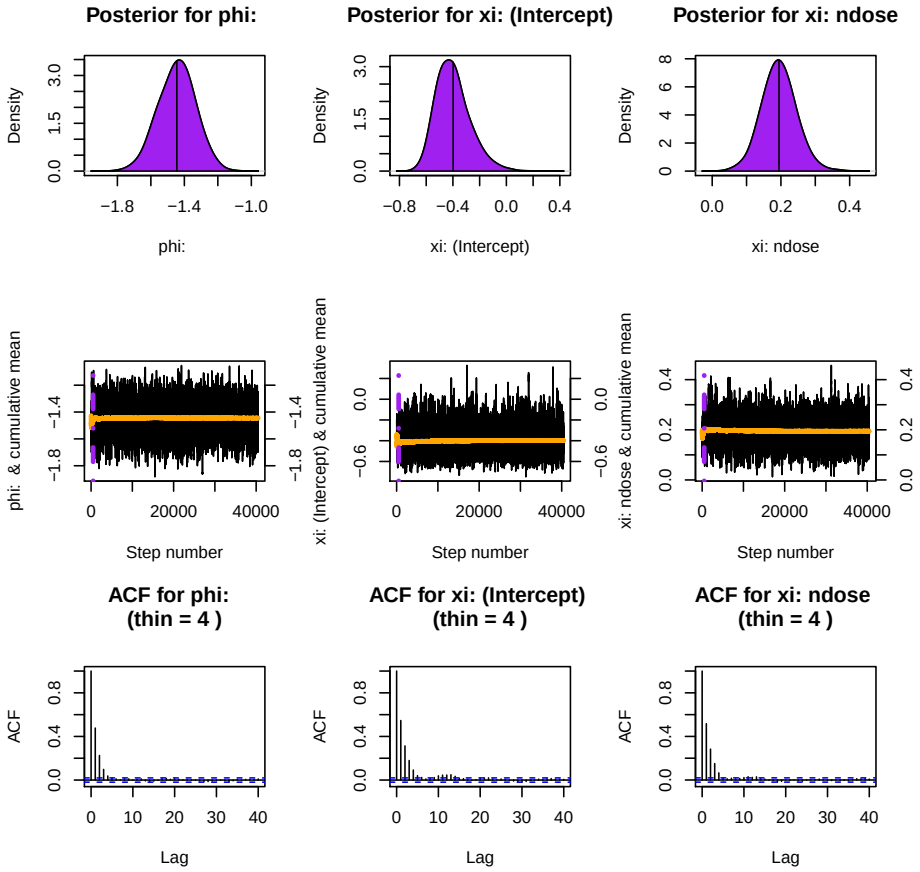
3.8 Bayesian estimation

A further alternative approach to uncertainty estimation which accurately reflects the asymmetric nature of the uncertainty is offered by Bayesian simulation based methods. In `texmex` we can simulate from the posterior distributions of the parameters by using the `gpd` function again, this time using `method = "simulate"` to tell the function to simulate from the joint posterior distribution of the parameters.

```
> bmod <- gpd(r, data=liver, qu=.7, xi= ~ndose, method="simulate")
```

Equivalently, the Bayesian estimation based on MCMC can also be instigated by the use of the function `update` on the previously chosen model. The method of estimation is changed from `"optimize"` – under which estimation is carried out using (penalized) maximum likelihood – to `"simulate"` – under which a Metropolis algorithm is used to simulate from the joint posterior distribution of the parameters. For our preferred model, `mod4`, this is implemented as follows:

```
> bmod <- update(mod4,method="simulate",trace=40000,penalty="gaussian")

40000  steps taken
Acceptance rate: 0.32

> par(mfrow=c(3, 3))
> plot(bmod)
> summary(bmod)

                Posterior mean        SD
phi:                -1.4448147 0.1107031
xi: (Intercept)     -0.3982904 0.1278481
xi: ndose            0.1944474 0.0506894
```



The plots of the Markov chains ought to look like "fat hairy caterpillars" if the algorithm has converged on its target distribution. Also, the cumulative means of the chains should converge, the acceptance rate should not be too high or too low, and the autocorrelation functions should rapidly decay to zero.

We conclude from the plots that there is no evidence against convergence of our Markov chains, although we should probably thin our output further to obtain a chain that is closer to independent (the default is to thin to every 4 observations). Here we retain the burn-in value of 500 but now discard all but every 20th observation, resulting in an autocorrelation function which decays more rapidly to zero. This restults in the retention of 2000 values after discarding the burn-in and applying the thinning. (Note that the observations are not discarded destructively and we can use the `thinAndBurn` function repeatedly to examine the impact of using different values of `burn` and `thin`.)

```
> bmod <- thinAndBurn(bmod, burn=500, thin = 20)
> nsim <- dim(bmod$param)[1]
> summary(bmod)

                Posterior mean         SD
phi:                -1.4438427 0.11058736
xi: (Intercept)     -0.4012245 0.12440685
xi: ndose            0.1954398 0.04995256

> nsim

[1] 2000
```

Since ϕ does not vary by `dose`, we can use the simulated values of ϕ to obtain predictions for any dose. However, ξ varies by dose, so we need to make sure we use the correct value for each treatment. We can use the `predict` method to obtain the linear predictors for the model parameters and for return level estimates.

```
> predict(bmod,type="lp")

Linear predictors:
      phi      xi ndose
1   -1.44 -0.2058     1
154 -1.44 -0.0103     2
302 -1.44  0.1851     3
450 -1.44  0.3805     4

> predict(bmod,M=1000)

M = 1000 predicted return level:
      res ndose
1   0.967     1
154 1.493     2
302 2.683     3
450 5.772     4
```

Setting the argument `all = TRUE` returns the linear predictors for all of the simulated parameter values in the (thinned) chains.

```
> bmodParams <- predict(bmod, type="lp", all=TRUE)
```

The returned object is a list with one item for each unique value of the covariate(s). The following shows the first five simulated values of (ϕ, ξ) for covariate `ndose = 1`:

```
> bmodParams[[1]][1:5,]
```

```
           phi          xi ndose
[1,] -1.526821 -0.06125516     1
[2,] -1.302622 -0.34874068     1
[3,] -1.305566 -0.21044040     1
[4,] -1.426107 -0.20035377     1
[5,] -1.090514 -0.41759541     1
```

Uncertainty in the GPD model parameters is combined with the other sources of uncertainty as follows.

3.9 Other sources of uncertainty

Our estimates of patient return levels on the original scale are derived by combining regression and extreme value models, and also need to reflect variation in baseline. Figure 3 illustrates how predictions can depend on patient baseline, which is inherently variable, so we also need to account for this and other sources of uncertainty. This is achieved by adopting a simulation based approach. GPD model parameters are simulated according to the Bayesian methodlogy outlined in Section 3.8. The methods of simulation used to represent further sources of uncertainy are described now.

3.9.1 Uncertainty due to variation in baseline

We simulate baseline values simply by resampling from the observed values. The regression model was fit to the logs of the data, so we resample the logs. We simulate `nsim` independent values for each of the four treatment levels (corresponding to the size of sample retained from our simulation from the posterior distribution of GPD model parameters):

```
> base <- sample(log(liver$ALT.B), size=4*nsim, replace=TRUE)
```

3.9.2 Robust regression model parameter uncertainty

We now need to simulate from the sampling distribution of our robust regression model coefficients. MM-estimates are asymptotically Gaussian distributed, so we simulate from their joint Gaussian distribution. First we need to construct the covariance of the regression parameters from the values returned by the `rlm` summary function.

```
> mycov <- summary(rmod)$cov.unscaled * summary(rmod)$stddev^2
> myloc <- coef(rmod)
> mycoefs <- rmvnorm(4*nsim, mean=myloc, sigma=mycov)
```

3.10 Return level estimation uncertainty

We now combine all the simulations from the different components of our model to obtain an estimate of uncertainty about our point estimates of m-patient return levels. Whereas before (Section 3.6) we had to fix a value of baseline, we can now average over baseline values within our sample.

Simulated expected post-treatment log ALT are obtained by combining the simulated baseline values with the simulated robust regression coefficients.

```
> ElogALT <- mycoefs[,1] + mycoefs[,2]*base + rep(1:4,each=nsim)*mycoefs[,3]
> ElogALT <- matrix(ElogALT,ncol=4)
```

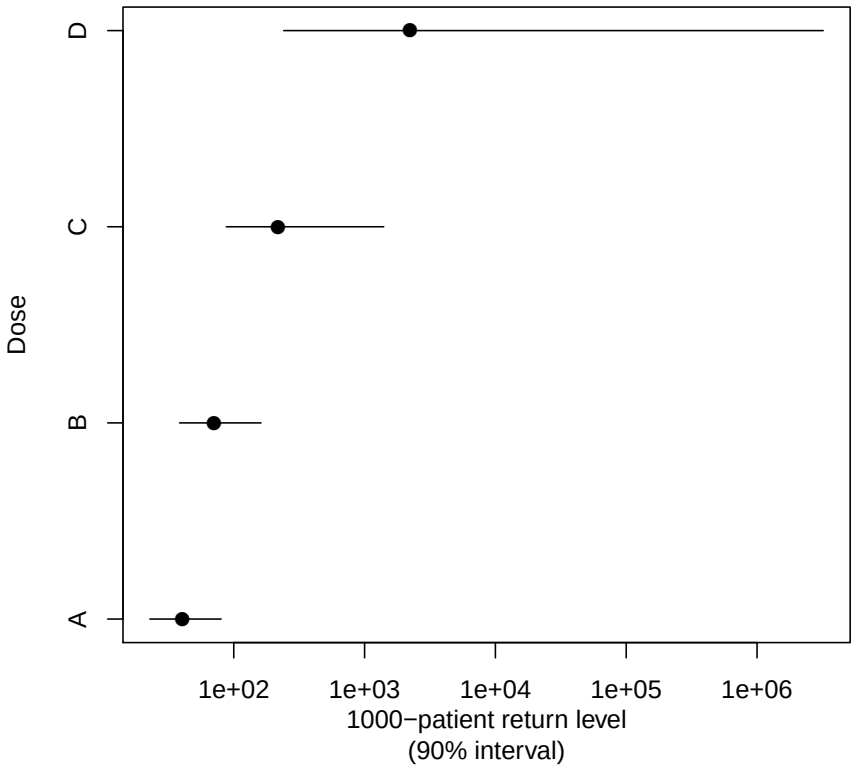
Note that the variability in the estimated regression parameters will be dwarfed by the variability due to extrapolation from the GPD model and by that due to variability in baseline values. As an exercise, try fixing the robust regression coefficients at their point estimates and see how much difference it makes to the final predictions.

The expected post-treatment values of log ALT are combined with the simulated GPD parameters to obtain simulated values of m-patient return levels (Equation 4) on the log scale:

```
> rl <- predict(bmod,M=1000,all=TRUE)[[1]]
> colnames(rl) <- LETTERS[1:4]
> rl[1:5,]

             A        B        C        D
[1,] 1.1957044 1.973071 3.639904 7.405224
[2,] 0.8229141 1.299642 2.428625 5.384815
[3,] 1.0501603 1.448361 2.134888 3.365261
[4,] 0.9667396 1.552460 2.855663 5.967568
[5,] 0.8805025 1.204856 1.803268 2.981786

> logRL <- rl + ElogALT
```

Finally, we exponentiate to get back to the scale of the raw data. Point estimates and approximate 90% credible intervals are estimated respectively by the empirical medians and 5 and 95% quantiles of our simulated return levels.

```
> srl <- exp(apply(logRL,2,quantile,probs=c(0.05,0.5,0.95)))
> par(mfrow=c(1,1))
> plot(srl[2, ], 1:4, xlim=range(srl), type="n", log="x",
+      axes=FALSE, xlab="1000-patient return level\n(90% interval)",
+      ylab="Dose")
> points(srl[2, ], 1:4, pch=16, cex=1.25)
> segments(x0=srl[1, ], x1=srl[3, ], y0=1:4)
> box()
> axis(1)
> axis(2, at=1:4, labels=LETTERS[1:4])
```



From the resulting figure, there is some evidence that dose D is associated with very large values of ALT, suggesting a potential safety issue.

3.11 Probability estimation uncertainty

We can also predict quantities such as $P(ALT > 3 \times ULN)$ by rearranging (4) on page 16:

$$\frac{1}{m} = p\left[(x_m - u)\frac{\xi}{\sigma} + 1\right]^{-1/\xi}.$$

(Note, p is the probability of exceeding the threshold for fitting the GPD.) Again, things are made complicated by the need to add the baseline effect back into the predictions. Here, we need also to account for the fact that for some fairly low values of interest (e.g. ULN, the Upper Limit of Normal), it is possible for the expected value of ALT given baseline to be greater than the specified value (in cases where baseline ALT is elevated). If the expected value of ALT is greater

than ULN, we need to consider the full distribution function $F(r) = P(R \leq r)$ of the residuals. The model for this distribution has two components: above the threshold for fitting, the excesses of the residuals over the threshold are modelled by the fitted GPD; below the threshold we use the empirical distribution $\tilde{F}(r)$ of the residuals:

$$\hat{F}(r) = \begin{cases} 1 - p\{1 + \xi(r-u)/\sigma\}^{-1/\xi} & r > u \\ \tilde{F}(r) & r \leq u. \end{cases}$$

This consideration is irrelevant for higher values of interest above which the expected value of ALT will never fall (e.g. interest is more usually in values $3 \times ULN$ and almost always $10 \times ULN$ or higher).

Noting that in the `liver` dataset, the ULN for ALT was 36 U/L, the code below does the necessary exceedance probability computations.

```
> rp <- function(xm, u, phi, xi, p, r) {
+   res <- p * (1 + xi/exp(phi) * (xm - u))^(-1/xi)
+   if (any(u > xm)){
+     res[u > xm] <- sapply(u[u>xm],
+                           function(x,r,m,p) mean((r + x - quantile(r,1-p)) > m),
+                           r=r,m=xm,p=p)
+   }
+   res[xi < 0 & xm > u - exp(phi)/xi] <- 0
+   res
+ }
> getProbs <- function(u, phi, xi, p, r, ULN, m = c(1, 3, 10, 20)) {
+   m <- log(ULN * m)
+   res <- t(sapply(m, rp, u = u, phi = phi, xi = xi, p = p, r=r))
+   res <- apply(res, 1, function(x){ c(quantile(x, c(.05, .95)),
+                Mean=mean(x))[c(1,3,2)] })
+   round(res, 4)
+ }
> DoCalc <- function(ElogALT, xi){
+   cnames <- paste("P(ALT > ", c("", "3x", "10x", "20x"), "ULN)",sep = "")
+   out <- getProbs(u = ElogALT+bmod$threshold,
+                   phi = bmodParams[[1]][,1], xi = xi,
+                   r=liver$r, p = 1-0.7,ULN = 36)
+   colnames(out) <- cnames
+   out
+ }
> rpA <- DoCalc(ElogALT = ElogALT[,1], xi = bmodParams[[1]][,2])
> rpB <- DoCalc(ElogALT = ElogALT[,2], xi = bmodParams[[2]][,2])
> rpC <- DoCalc(ElogALT = ElogALT[,3], xi = bmodParams[[3]][,2])
> rpD <- DoCalc(ElogALT = ElogALT[,4], xi = bmodParams[[4]][,2])
> rpA
     P(ALT > ULN) P(ALT > 3xULN) P(ALT > 10xULN) P(ALT > 20xULN)
5%         0.0000          0e+00               0               0
```

```
Mean       0.0399          9e-04               0               0
95%        0.2180          1e-04               0               0
```

```
> rpB
```

```
     P(ALT > ULN) P(ALT > 3xULN) P(ALT > 10xULN) P(ALT > 20xULN)
5%         0.0014         0.0000           0e+00               0
Mean       0.0609         0.0022           1e-04               0
95%        0.3030         0.0032           2e-04               0
```

```
> rpC
```

```
     P(ALT > ULN) P(ALT > 3xULN) P(ALT > 10xULN) P(ALT > 20xULN)
5%         0.0101         0.0006          0.0000          0.0000
Mean       0.0847         0.0065          0.0009          0.0004
95%        0.3449         0.0134          0.0028          0.0016
```

```
> rpD
```

```
     P(ALT > ULN) P(ALT > 3xULN) P(ALT > 10xULN) P(ALT > 20xULN)
5%         0.0204         0.0033          0.0006          0.0003
Mean       0.1097         0.0127          0.0039          0.0025
95%        0.4489         0.0272          0.0093          0.0065
```

The above tables contain summary statistics from posterior predictive distributions. When computing the return levels, we used the median as an estimate of the centre of the posterior distribution. The reason for this is that experience has shown that for some distributions with extremely heavy tails, the mean can be unduly influenced by very large values and can even be outside the 90 or 95% credible interval (as estimated by the sample quantiles). The distribution is often skewed, so the mean and median can differ substantially; in an ideal world we might display the entire posterior rather than simple summaries. For the estimation of probailities, the range of values is bounded by $[0, 1]$ and so here we report sample means instead of medians.

The tables suggest that under dose D (the highest dose), approximately 0.2% of patients (about 1 in 500) will get an ALT greater than $20 \times ULN$, a level described by CTC as *life threatening*.

References

[1] Matthias Burger, Klaus Juenemann, and Thomas Koenig. *RUnit: R Unit test framework*, 2010. R package version 0.4.26.

[2] S. Coles. *An Introduction to Statistical Modelling of Extreme Values*. Springer, 2001.

[3] A. C. Davison and R. L. Smith. Models for exceedances over high thresholds. *Journal of the Royal Statistical Society, Series B*, 53:393 – 442, 1990.

[4] Ricardo A. Maronna, R. Douglas Martin, and Victor J. Yohai. *Robust Statistics: Theory and Methods*. Wiley, 2006.

[5] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[6] H. Southworth and J. E. Heffernan. Extreme value modelling of laboratory safety data from clinical studies. *Pharmaceutical Statistics*, 2012.

[7] H. Southworth and J. E. Heffernan. *texmex: Threshold exceedences and multivariate extremes*, 2012. R package version 1.3.

[8] H. Southworth and J. E. Heffernan. Multivariate extreme value modelling of laboratory safety data from clinical studies. *Pharmaceutical Statistics*, to appear.

[9] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.