Probabilistic Risk Analysis and Bayesian Decision Theory

NOTEBOOK

Marcel van Oijen* Mark Brewer†

2025-09-11

Contents

*Independent researcher, Edinburgh, UK - VanOijenMarcel@gmail.com

†BioSS Office, The James Hutton Institute, Craigiebuckler, Aberdeen AB15 8QH

1 Practicalities

1.1 My lectures

- Can I use my own MBP, i.e. can my laptop screen be projected?

1.2 Where, When, Who

- Course: Lubeck 2025-09-24 to 26.
- Travel: out 2025-09-23, return 2025-09-27.
- Local organisers:
  - Reinard Vonthein (Lubeck, mathematics of medicine and life sci.)
  - Tobias Mette (Freising, interested in GAM)
  - Gerhard Nehmiz (Bieberach, mathematics, biostatistics)
  - Wolfgang Falk

1.3 Prerequisites

- R and R Studio installed
- PRAbook code tested?
  - New GitHub repository
- Participants will be familiar with R, but let them know which R-packages need to be installed.
- Make slides freely available to all!

1.4 Material for participants to download

2 Lectures

2.1 Agenda

We aim to cover the following six topics:

1. Introduction to PRA: Basic ideas & equations (**BC2: 18.1-18.6, PRA-BDT: 1-9**)
2. Model-based PRA (incl. NIMBLE example Spain) (**PRA-BDT: 7**)
3. Expanding PRA-theory (incl. Interacting hazards) (**BC2: 18.7, PRA-BDT: 10-11, 18**)
4. Introduction to BDT (**BC2: 19.1-19.3, BC1: 17.2, PRA-BDT: 12-15**)
5. Links between PRA and BDT (**BC2: 19.4, PRA-BDT: 16-17**)
6. General discussion (**BC2: 25.1, PRA-BDT: 19**)

There will be four sessions of 1.5 hours each, so some sessions will cover multiple topics. The progress through the material will be decided together with the audience, but roughly we can expect the following scheduling of the topics:

Session I : 1 Session II : 2 Session III: 2,3 Session IV : 4,5,6

We aim to include exercises in each session to initiate discussion.

2.2 Content ideas

- Briefly summarise our previous work: Marcel Van Oijen (2024), Marcel Van Oijen and Brewer (2022), Marcel Van Oijen and Zavala (2019), M. Van Oijen (2019), M. Van Oijen et al. (2014), M. Van Oijen et al. (2013).
- Show examples from other authors who used (parts of) our PRA approach explicitly: Ren et al. (2023), Nandintsetseg et al. (2021), He et al. (2021), Zhou et al. (2018), Kuhnert et al. (2017), Liu et al. (2020), Nandintsetseg et al. (2024).
- Study examples (with pictures) of published risk analyses that do NOT use our PRA approach and ask: do these risk analyses distinguish and quantify $x, z, p[x], p[z], E[x], E[z], E[x|..], E[z|..], V, R$?
  - Fun different example: risk of large meteorite hitting the earth [https://www.nasa.gov/solar-system/asteroids/asteroid-fast-facts/; Wheeler et al. (2024)]
- In structuring the course, mainly follow BC2, Chapters 18 and 19 (Marcel Van Oijen 2024).
  - For additional material, see also our notes on Marcel Van Oijen and Brewer (2022) in the next chapter of this document, and the other chapters on specific topics.
- What **not** to include? (Maybe mention briefly but do not treat in any depth.)
  - Gerhard suggests not to include copulas. But note that an additional use of copulas (besides our use in expressing p[x,z]) may be in spatial upscaling of local risks to larger areas (Hochrainer-Stigler 2019).
  - Extreme value theory.
- Add a line-plot of $E[z|x]$ and for the multi-threshold PRA also a barplot with the set of x-interval conditional expectations for z, i.e $E[z|thr_{i-1} \leq x < thr_i]$ where $i = 1, .., n$ and $thr_0 = -\infty$.
- Revive fingerprint-analysis for bifactorial hazard (or for mean and s.d. of hazard).
- Mention Bayesian packages?
  - brms
  - BayesX (developed and used in Germany)
    - https://cran.r-project.org/web/packages/BayesX/index.html
    - https://www.rdocumentation.org/packages/BayesXsrc/versions/3.0-6

  - ...
- Can NIMBLE be linked to external PBMs?
- Should Bayesian approaches to regression (incl. machine learning, CART) be mentioned?
  - https://proceedings.mlr.press/r1/chipman97a.html
- Can fault-tree-analysis (FTA, the nuclear industry's form of PRA) be carried out within our PRA framework? Maybe FTA can be seen as a decomposition of the hazard probability?
  - https://pipelinepodcastnetwork.com/probabilistic-risk-analysis/
- Do we need to think about exchangeability? See blog posts by Jessica Hullman and the reference in the comments there to Jaynes' article about the de Finetti representation theorem. Exchangeability is also covered in Jaynes' book.
  - https://statmodeling.stat.columbia.edu/2025/03/26/individual-probability-model-multiplicity-and-multicalibration/?utm_source=substack&utm_medium=email
- When carrying out PRA, can my standard time-series analysis (replication over time) be replaced by a spatial analysis (replication over space)?
- Mention increasing role of machine learning?
  - Salcedo-Sanz et al. (2023): "Analysis, Characterization, Prediction and Attribution of Extreme Atmospheric Events with Machine Learning: a Review"
- When to normalise V and R? Loss can be expressed in absolute terms (e.g. mˆ3 ha-1 y-1) or relative terms (%, where 100% is equated with $\overline{z_{\neg H}}$)? Generally it may be good to normalise, but it may lead to confusion in case of spatially distributed PRA if we normalise every cell separately. Assessing total risk over the whole area then becomes complicated.
  - Do we also have such problems with area-integration if we use local measures of p[H], such as the SPEI?
- Marcel to keep discussion of Bayes and model-based PRA very short: will be covered by Mark.
- In the PowerPoint file: use the term 'Graphical Model' (GM) throughout rather than BBN.
- After presenting different forms of PRA (e.g. sampling-, distribution-, and model-based PRA for the German *Picea abies* data), ask why the results differ. Which method seems most reliable? More generally, what prior information does each method require?
- BDT explained simply: http://www.statsathome.com/2017/10/12/bayesian-decision-theory-made-ridiculously-simple/
- We show, in paragraph 2.1.1 (p.10) of Marcel Van Oijen and Brewer (2022) that a uniform hazard distribution together with a linear response function leads to constant V irrespective of the choice of threshold. Maybe show this as a simple example of distribution- and model-based PRA?

2.2.1 PRA using forest data

- Explain all variants of PRA by applying them to forest mortality.
- Show naive PRA on time series German forest death (Thunen institute) and point out that *domain knowledge* is essential. For example, in a simplistic PRA we would ignore the cumulative effects that have led to droughts in recent years becoming more damaging than earlier droughts. In other words, annual drought levels should not be viewed as constituting independent hazard occurrences.
- ANotHer simplistic aspect: ignoring correlations between different environmental variables such as drought and heat and disease and pests. This is not always wrong: if the correlations are very strong than we can indeed treat them as one compound hazard. But generally the correlations are weak to moderate so *different hazard variables should not be lumped*.
- Include the topic of *reporting the results* from PRA. Examine how to put the results of PRA in accurate, informative and easily understandable language. Show how the PRA is useful to a forest manager.

2.2.2 Mark's PRA for the Spanish data (Nimble example)

- Demonstrate practical application of the PRA to real data from Spain.
- Develop the Nimble approach of PRAbook Ch. 7 into a large realistic case-study. Use Manuel's Spanish data that Mark wanted to analyse anyway with our methods. So that would be for both the course and a publication. That example will be of interest to the audience because many of them likely work on climate extremes impacts on German forests.
- Extend to data from Germany and Finland? Like Spain, these countries have long-term data on forest growth, so the three countries together could be used to analyse environmental risks in different climates.

3 Marcel Van Oijen and Brewer (2022)

3.1 Errata

- p.14 refers to 'Monte Carlo' (MC) four times but the approach here is based on numerical solutions to the conditional expectation values without any MC sampling.
- p.21, 4.3 "any deviation ... $E[z \geq thr]$" should be ... $E[z|x \geq thr]$. (But note that the whole equation for σ_R may be simplified, see next section on book improvements.)
- Eqs 2.7, 2.8 and 4.5 are not approximations but exact. Make appropriate changes in text and equations. For example, change the first sentence of 2.3 from "Even though in the case of the bivariate Gaussian no closed-form solution exists for the distribution of z given that x is below or above a threshold, there is an approximation formula for ..." to "If $p[x, z]$ is a bivariate Gaussian, then $p[z|x]$ is a Gaussian too, but the two distributions that we need for PRA, i.e. $p[z|x < thr]$ and $p[z| \geq thr]$, are not Gaussian. However there are closed-form solutions for the means and variances of these two distributions (e.g. Mee and Owen (1983)), which we show here in their most general form."
- There are errors in text, equations and underlying code of section 4.7 (pp. 26-27).
  - **Text**: "...approximately estimating ... The same authors also provided a formula to approximate the conditional variance for the same distribution: $Var[z|x < thr] = 1 + \rho\, thr\, E[z] - E[z]^2$" should not refer to approximation (the formulas are exact) and have conditional expectations on the right: "... $= 1 + \rho\, thr\, E[z|x < thr] - E[z|x < thr]^2$".
  - **Equation 4.5**: The equations as written give a value equal to 1 for the conditional variances if $\rho = 0$ and $thr \to \infty$ resp. $thr \to -\infty$. The proper limits are σ_z^2 so that is what both appearances of "1" should be replaced by.
  - **Code**: In the underlying code (not shown in the book) for functions `Varz_xbelowAppr` and `Varz_xaboveAppr`, the calculation of `Varzxbelow` resp. `Varzxabove` should operate on `mx.` and `mz.` instead of `mx` and `mz`. And for consistence we may also prefer to use `thr.` and `r.` rather than `thr` and `r` as function arguments. The old argument-names will generally not cause problems but should be changed anyway. When also correcting the above-mentioned equation-errors, the corrected code becomes:
    - `Varzxbelow` <- Vz. + r. * (thr.-mx.) * (Ezxbelow-mz.) - (Ezxbelow-mz.)^2
    - `Varzxabove` <- Vz. + r. * (thr.-mx.) * (Ezxabove-mz.) - (Ezxabove-mz.)^2
- Eq. 11.1 is (very) wrong: the four R_e should each be Q. So correct the equation and extend the paragraph with a test of the corrected equation on virtual data from a known joint spatial distribution $p[x_i, z_i]$ for an 8x8 grid, just like the other examples in the chapter.

3.2 Improvements

- Mention a new location for downloadable R-code (i.e. my Zenodo/GitHub), and explain where errors were corrected and for which topics additional code is provided.
- Include exercises?
- Remove Eq. 2.2 (not used anywhere), and modify Eq. 2.1 (any term involving $x|z$ is confusing) such that it more clearly leads to the current Eq. 2.3.
- Chapter ordering is confusing: better start from the structure of BC2 Ch. 18, or from the order used in this document.
- Move paragraphs 2.3 and 4.7 to Ch. 5? And then simplify the title of that chapter to "Distribution-Based Single-Threshold PRA" or just "Distribution-Based PRA".
- p.21, 4.3. Replace Eq. 4.4 with the simpler $\sigma_R^2 = \sigma_{p[H]}^2 \sigma_V^2 + \sigma_{p[H]}^2 V^2 + p[H]^2 \sigma_V^2$.
  - The same simplification for s_R can be introduced in the R-function of p.28.
- Also UQ to the sampling-based multi-threshold PRA of Ch.8. This can use the above-mentioned new formula: $\sigma_{R[i]}^2 = \sigma_{p[H[i]]}^2 \sigma_{V[i]}^2 + \sigma_{p[H[i]]}^2 V[i]^2 + p[H[i]]^2 \sigma_{V[i]}^2$.

- Provide some Bayesian justification for our treatment of UQ in sampling based-PRA, in particular Eqs 3.1 and 4.1 of the PRAbook. Those are unbiased estimators (which need not be the best estimators at all, see e.g. Jaynes (2003) or Oliphant (2006)). Note that the problem of "measuring" a population mean using a finite sample with sampling uncertainty from population variation is similar to that of measuring the value of a parameter from repeated measurements with measurement uncertainty. Both problems are somewhat difficult if a priori not only the mean but also the variance among individual observations is unknown. We could then still use conjugate analysis using a Normal-Gamma prior (for mean and *precision*) leading to a marginal posterior Student's distribution for the mean (Murphy (2007)) which is reasonably well approximated by our Eqs 3.1 and 4.1 if prior uncertainty is high. For a comparison of the Bayesian treatment with other methods, see Oliphant (2006).
- If we consistently point out Bayesian justifications for our formulas, we can rename the book "Bayesian Risk Analysis and Decision Theory".
- Interactions: include examples where the p[H]*i and/or the* $V\{\text{ij}\}$ are not independent.
  - Check my handwritten notes on that (separate A4).
  - Spatially interacting hazards: see my remarks about that at the end of the book (paragraph 19.8) with reference to Timonina-Farkas et al. (2013, 2015). Also check the literature for newer publications by the same authors (maybe already in my Zotero library).
- Code of Ch. 7: simplify where possible and make consistent with coding conventions in rest of the book (e.g. use of the term 'pH' rather than 'xweight' etc.).
- The function PRAm (formerly PRAn) now produces a list of length 2, containing a summary vector, and a matrix with results for all thresholds.
- Our user-defined R-functions:
  - Be consistent in argument-naming: always have a decimal point at the end. So write `x., z., m., S., thr., thr.seq.` etc.
  - Rename our R-functions: we should assume that UQ is always required in PRA, so no need to have ".UQ" at the end of algorithm-names. The few examples of code that do not include UQ should perhaps be removed.
- We formerly used the beta distribution to model uncertainty about hazard probability pH, but often summarise its posterior as mean plus-minus standard deviation. Better to calculate the 95% credibility interval associated with the asymmetric posterior beta distribution. [Or do not use the beta distribution at all but leave it at standard sampling theory.]
- Does Ch. 5 add anything substantial to what was already explained in section 4.7?
- Update references, e.g. replace Marcel Van Oijen (2020) by Marcel Van Oijen (2024).
- In the Discussion chapter summarise the key differences between sampling-, distribution-, and model-based PRA, in both formulation and applicability.
  - Only the latter two specify a function $E[z|x]$, and they are in fact closely related. For example, a Gaussian bivariate distribution is equivalent to a linear regression model.
  - Sampling-based PRA is easiest to apply, but that brings its own risks: it is necessarily non-Bayesian and thus does not have a prior that can keep results sensible even if there are only few data.
- More generally, explain the value of Bayesian approaches to PRA and BDT.
- Remove sections on model emulation? It is not specific to PRA and/or BDT.

3.2.1 BDT

- Include a decision tree (although most variables in our book are continuous including the action levels such as irrigation).
- The GM of Marcel Van Oijen (2024) Fig. 19.1 seems more natural than its predecessor Fig. 12.1 of Marcel Van Oijen and Brewer (2022).

- Include a DAG for PRA (simply x -> z), then a very simple one for BDT ({a,x} -> z with {a,z} -> u), then the one for Fig. 19.1 and explain that we can add detail as we see fit, commensurate with our knowledge of the system. Make clear that we can have the action 'a' operate on any subset of {x, z, u}. For example, if a is irrigation we could model that as an extra resource affecting the system 'z', but also as management that increases environmental water availability {x}.

- Mention the two key values of DAG for BDT:
  * 1. Visual representation of joint probability distribution,
  * 2. Showing *conditional* independencies as pairs of nodes without an arrow between them, affording an easy way to factorise the joint distribution.

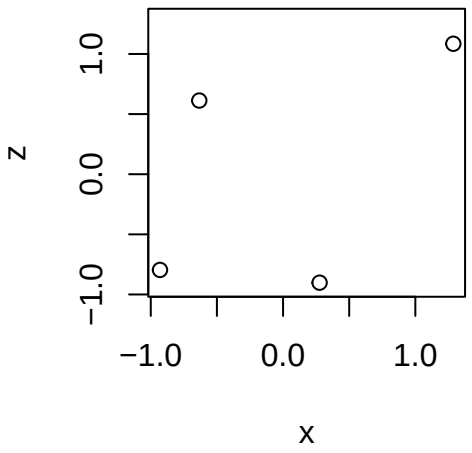
4 Data

We use several virtual and one real dataset in this notebook.

4.0.1 Sparse Dataset: `{x_4,z_4}`

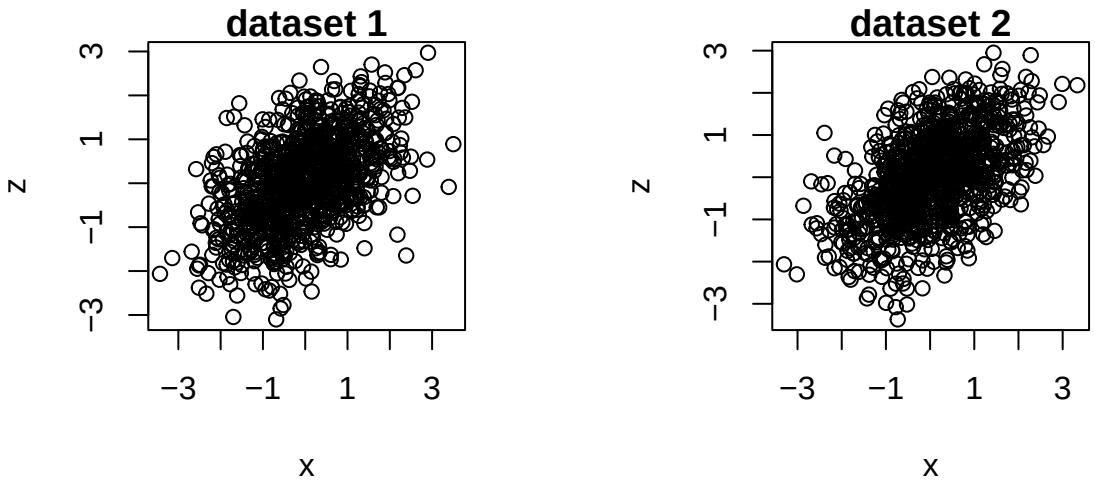
We create a sparse dataset (n=4) for $\{x, z\}$ that has a zero mean vector, unit variances and covariance 0.5 as follows:

```
set.seed(1)

n_4  <- 4 ; Sigma_4 <- matrix( c(1,0.5,0.5,1), nrow=2 )
xz_4 <- rmvnorm( n_4, c(0,0), Sigma_4 ) %*% chol( Sigma_4 )
xz_4 <- sweep( xz_4, 2, colMeans(xz_4) )
xz_4 <- xz_4 %*% solve( chol(cov(xz_4)) ) %*% chol(Sigma_4)
x_4  <- xz_4[,1]       ; z_4 <- xz_4[,2]
m_4  <- colMeans(xz_4) ; S_4 <- cov(xz_4)

par( mfrow=c(1,2), mar=c(4,4,1,4) )
plot( x_4, z_4, xlim=range(xz_4), ylim=range(xz_4), xlab="x", ylab="z" )
```



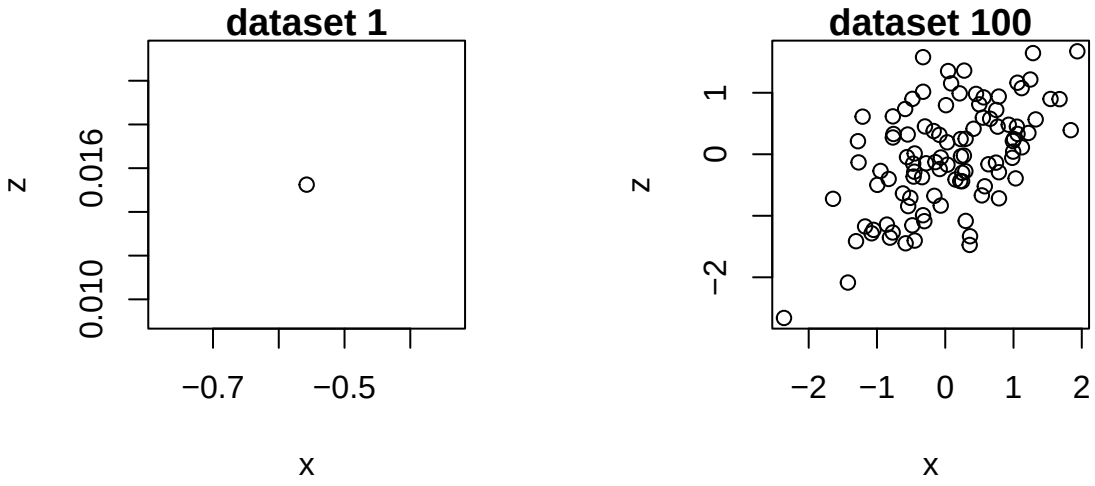
4.0.2 A Collection of Linear Datasets: `l_xz.L`

```
set.seed(1)

mu  <- c(0,0) ; Sigma  <- diag(1,2)          ; Sigma[1,2] <- Sigma[2,1] <- 0.5
n_d <- 1e2    ; l_xz.L <- vector("list",n_d) ; n <- 1e3
for(d in 1:n_d) { l_xz.L[[d]] <- rmvnorm( n, mu, Sigma ) }

par( mfrow=c(1,2), mar=c(4,4,1,4) )
plot( l_xz.L[[1]][,1], l_xz.L[[1]][,2], main="dataset 1", xlab="x", ylab="z" )
plot( l_xz.L[[2]][,1], l_xz.L[[2]][,2], main="dataset 2", xlab="x", ylab="z" )
```



4.0.3 A Sequence of Linear Datasets (n linearly increasing)

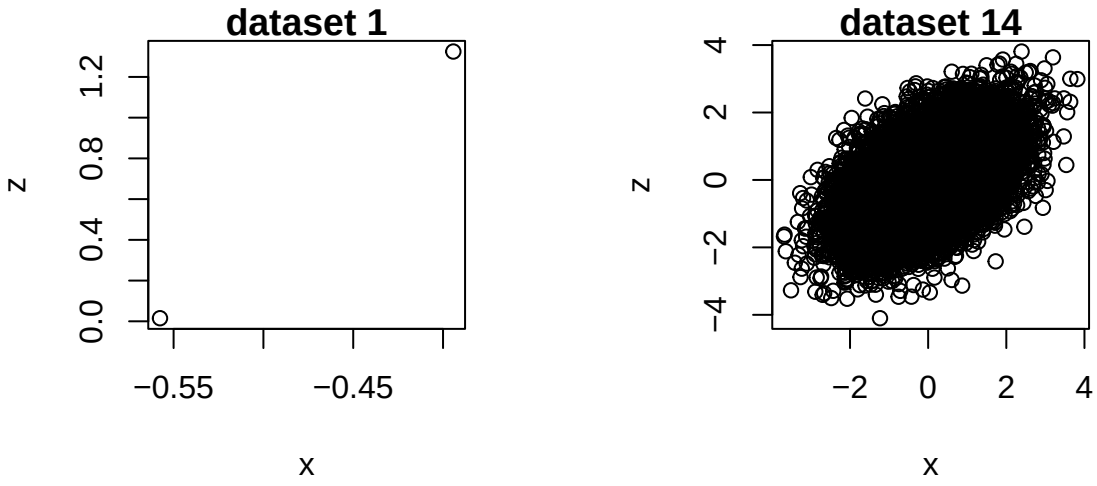
```
set.seed(1)

mu  <- c(0,0) ; Sigma  <- diag(1,2) ; Sigma[1,2] <- Sigma[2,1] <- 0.5
n_d <- 1e2    ; l_xz.L2 <- vector("list",n_d)
for(d in 1:n_d) { l_xz.L2[[d]] <- rmvnorm( d, mu, Sigma ) }

par( mfrow=c(1,2), mar=c(4,4,1,4) )
plot( l_xz.L2[[1  ]][,1], l_xz.L2[[1  ]][,2], main="dataset 1",
      xlab="x", ylab="z" )
plot( l_xz.L2[[n_d]][,1], l_xz.L2[[n_d]][,2], main=paste("dataset",n_d),
      xlab="x", ylab="z" )
```



4.0.4 A Sequence of Linear Datasets (n exponentially increasing)

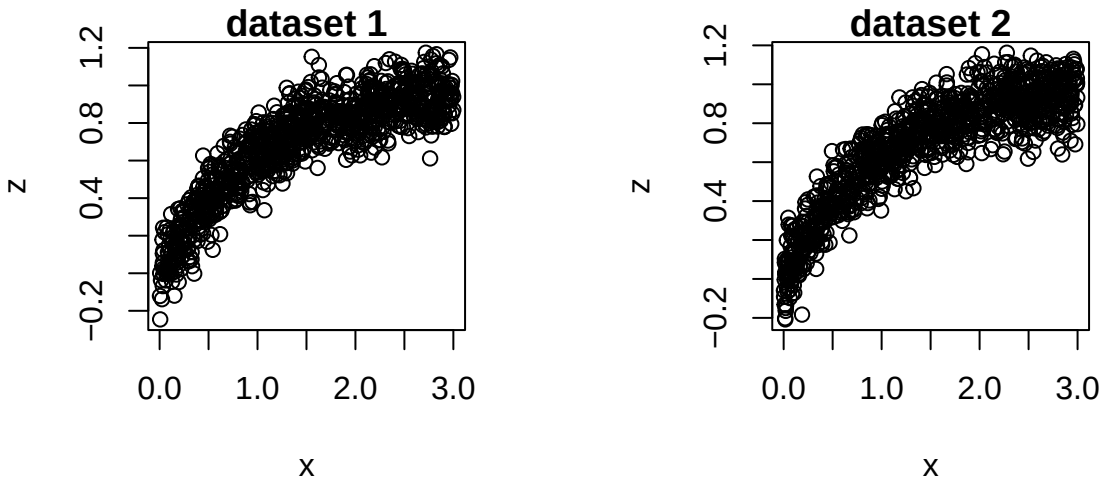
```
set.seed(1)
```

```
mu  <- c(0,0) ; Sigma  <- diag(1,2) ; Sigma[1,2] <- Sigma[2,1] <- 0.5
n_d <- 14     ; l_xz.L3 <- vector("list",n_d)
for(d in 1:n_d) { l_xz.L3[[d]] <- rmvnorm( 2^d, mu, Sigma ) }

par( mfrow=c(1,2), mar=c(4,4,1,4) )
plot( l_xz.L3[[1  ]][,1], l_xz.L3[[1  ]][,2], main="dataset 1",
      xlab="x", ylab="z" )
plot( l_xz.L3[[n_d]][,1], l_xz.L3[[n_d]][,2], main=paste("dataset",n_d),
      xlab="x", ylab="z" )
```



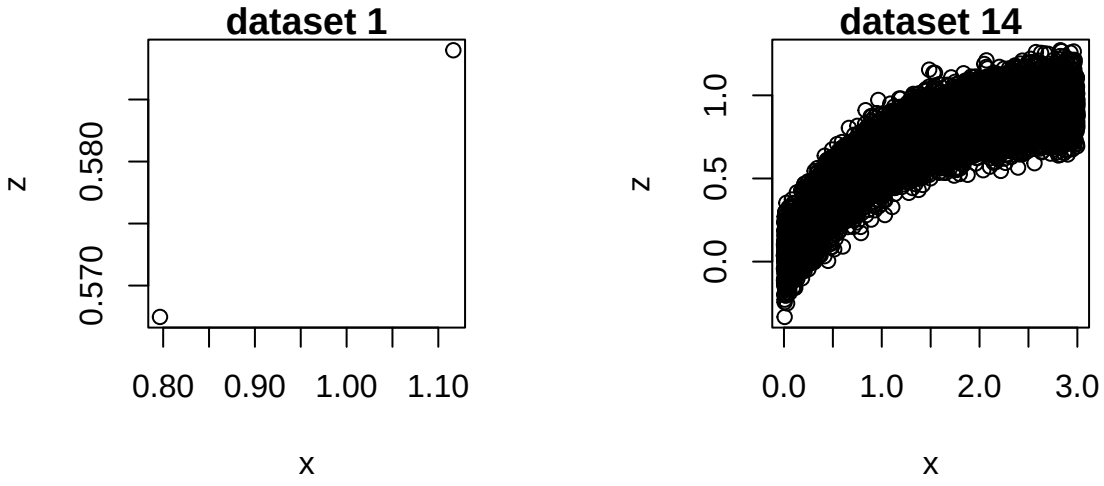
4.0.5 A Collection of Nonlinear Datasets: `l_xz.NL`

```
set.seed(1)

n_d <- 1e2 ; l_xz.NL <- vector("list",n_d) ; n <- 1e3 ; sz <- 0.1
for(d in 1:n_d) {
  x <- runif( n, 0, 3 ) ; ez <- rnorm( n, 0, sz) ; z <- 1-exp(-x) + ez
  l_xz.NL[[d]] <- cbind(x,z) }

par( mfrow=c(1,2), mar=c(4,4,1,4) )
plot( l_xz.NL[[1]][,"x"], l_xz.NL[[1]][,"z"], main="dataset 1", xlab="x", ylab="z" )
plot( l_xz.NL[[2]][,"x"], l_xz.NL[[2]][,"z"], main="dataset 2", xlab="x", ylab="z" )
```



4.0.6 A Sequence of Nonlinear Datasets (n exponentially increasing)

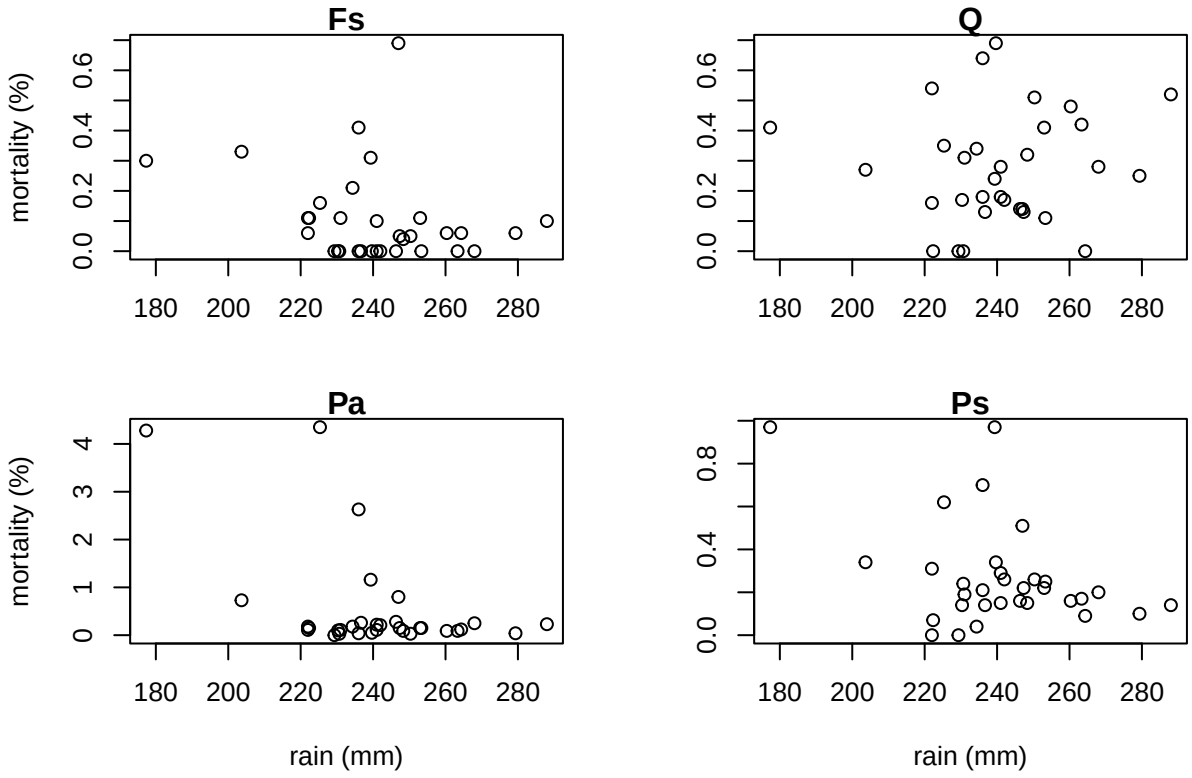
```
set.seed(1)

n_d <- 14 ; l_xz.NL3 <- vector("list",n_d) ; sz <- 0.1
for(d in 1:n_d) {
  x <- runif( 2^d, 0, 3 ) ; ez <- rnorm( 2^d, 0, sz) ; z <- 1-exp(-x) + ez
  l_xz.NL3[[d]] <- cbind(x,z) }

par( mfrow=c(1,2), mar=c(4,4,1,4) )
plot( l_xz.NL3[[1  ]][,1], l_xz.NL3[[1  ]][,2], main="dataset 1",
      xlab="x", ylab="z" )
plot( l_xz.NL3[[n_d]][,1], l_xz.NL3[[n_d]][,2], main=paste("dataset",n_d),
      xlab="x", ylab="z" )
```



4.0.7 German Forestry Data: `x_r3`, `z_Fs`, `z_Q`, `z_Pa`, `z_Ps`

```
d.data        <- "data_Germany/"
```

```
  file.data     <- paste0( d.data, "2-bis-7_abb-tab_nsh_2024-04-08.xlsx" )
  sheet.data    <- "5_Daten" ; r.data <- "B120:C153"
  y_r           <- read_excel( file.data, sheet=sheet.data, range=r.data, col_names=F )
> New names:
> * `` -> `...1`
> * `` -> `...2`
  y_r           <- round( as.matrix(y_r) )
  colnames(y_r) <- c( "year", "rain.mm" )

  file.data     <- paste0( d.data, "absterberate_EI_GFI_GKI_RBU_zeitreihe.csv" )
  y_m           <- read.csv2( file.data )
  colnames(y_m) <- c( "year", "mort.%_Fs", "mort.%_Q", "mort.%_Pa", "mort.%_Ps" )
  y_r_m         <- cbind( y_r, y_m[ , startsWith( colnames(y_m), "mort" ) ] )

  # Annual mortality vs. summer rain of last three years
  meanlast <- function(v,k){as.vector(filter(v,f=rep(1/k,k),s=1))}
  n    <- dim(y_r_m)[1] ; k <- 3
  x_r3 <- meanlast(y_r_m[,"rain.mm"],k)[k:n]

  z_Fs <- y_r_m[k:n,"mort.%_Fs"] ; z_Q  <- y_r_m[k:n,"mort.%_Q" ]
  z_Pa <- y_r_m[k:n,"mort.%_Pa"] ; z_Ps <- y_r_m[k:n,"mort.%_Ps"]

  par ( mfrow=c(2,2), mar=c(4,4,1,2) )
  plot( x_r3, z_Fs, main="Fs", xlab=""         , ylab="mortality (%)" )
  plot( x_r3, z_Q , main="Q" , xlab=""         , ylab=""              )
  plot( x_r3, z_Pa, main="Pa", xlab="rain (mm)", ylab="mortality (%)" )
  plot( x_r3, z_Ps, main="Ps", xlab="rain (mm)", ylab=""              )
```

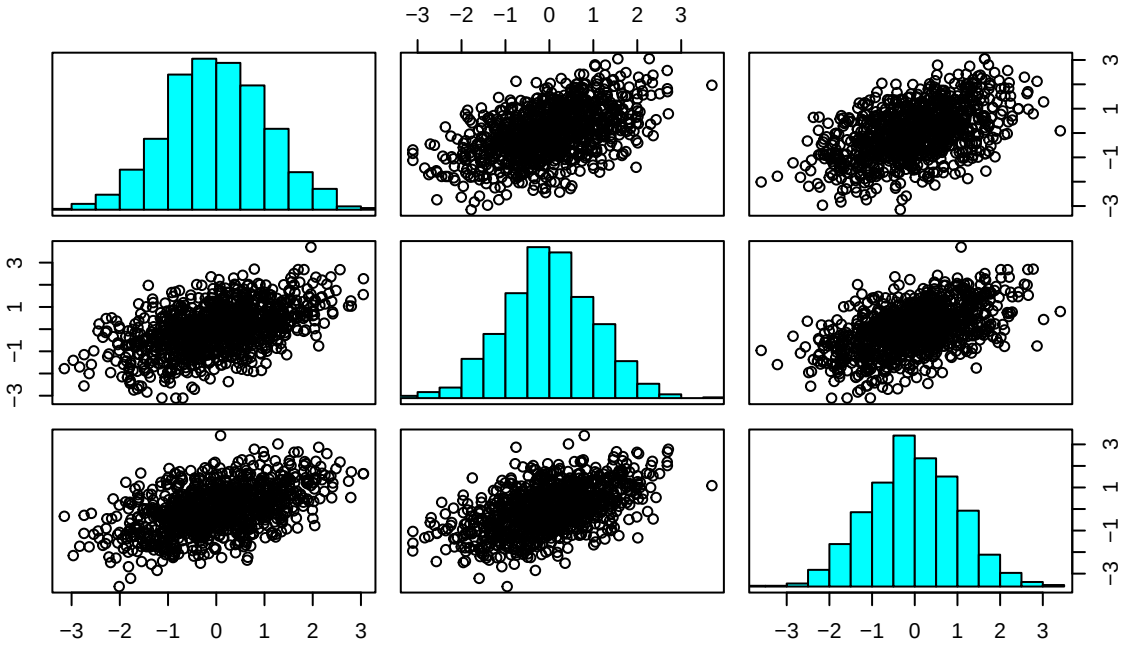


4.0.8 Trivariate Gaussian Dataset

We use the following histogram-grid function to show sample data from the joint distribution $p[x1, x2, z]$:

```
panel.hist <- function(x, ...) {
  h      <- hist(x, plot=F)
  breaks <- h$breaks; n_b <- length(breaks)
  y      <- h$counts; y   <- min(x) + (max(x)-min(x)) * y/max(y)
  rect(breaks[-n_b], min(x), breaks[-1], y, col="cyan", ...) }
```

```
set.seed(1)

n_G3  <- 1e3 ; Sigma_G3 <- matrix(0.5,nrow=3,ncol=3) ; diag(Sigma_G3) <- 1
xz_G3 <- rmvnorm( n_G3, c(0,0,0), Sigma_G3 )

pairs(xz_G3, diag.panel=panel.hist, label="" ) ; cor(xz_G3)
```




```
>           [,1]      [,2]      [,3]
> [1,] 1.0000000 0.4993916 0.4933134
> [2,] 0.4993916 1.0000000 0.5190580
> [3,] 0.4933134 0.5190580 1.0000000
```

5 Sampling-Based PRA

5.1 Single-threshold PRA

Our function `PRA` for single-threshold sampling-based PRA was already coded before (Marcel Van Oijen and Brewer (2022) p.21) and that code is repeated below, with two modifications:

- s_R is now calculated as the square root of the variance of the product of pH and V
- s_pH is now calculated following standard sampling theory as $s_{pH} = \sqrt{p_H(1-p_H)\,/\,n}$. Formerly a Bayesian Beta-posterior was used.

```
PRA <- function( x, z, thr=0 ) {
  n       <- length(x)      ; H         <- which(x < thr) ; n_H <- length(H)
  Ez_H    <- mean( z[ H] )  ; s_Ez_H    <- sqrt( var(z[ H]) /    n_H  )
  Ez_NotH <- mean( z[-H] )  ; s_Ez_NotH <- sqrt( var(z[-H]) / (n-n_H) )
  pH      <- n_H / n        ; V         <- Ez_NotH - Ez_H ; R  <- pH * V
  s_pH    <- sqrt( pH*(1-pH) / n )
  s_V     <- sqrt( s_Ez_H^2 + s_Ez_NotH^2 )
  s_R     <- sqrt( s_pH^2*s_V^2 + s_pH^2*V^2 + pH^2*s_V^2 )
  return( c(pH=pH,V=V,R=R,s_pH=s_pH,s_V=s_V,s_R=s_R) )
}
```

Sampling-based PRA requires a sufficiently large dataset to allow calculation of variances and thus UQ: $n_H \geq 2$ and $(n - n_H) >= 2$.

```
  x <- c( 200, 600 )           ; z <- c(  70,  90 )
  PRA( x, z, 500 )
>         pH          V          R       s_pH        s_V        s_R
>  0.5000000 20.0000000 10.0000000  0.3535534         NA         NA

  x <- c( 200, 400, 600, 800 ) ; z <- c(  70,  80,  90, 100 )
  PRA( x, z, 500 )
>        pH         V         R      s_pH       s_V       s_R
>  0.500000 20.000000 10.000000  0.250000  7.071068  6.373774
```

5.2 Multi-threshold PRA

We now consider PRA where the number of thresholds n_{thr} exceeds 1. The space of x then consists of $k = n_{thr} + 1$ disjoint intervals of which one is the non-hazardous region.

The following code for function `PRAm` (multiple-threshold PRA) is new in that it now includes UQ. It is an extension of the original code for function `PRAn` (Marcel Van Oijen and Brewer (2022) p.45).

```
PRAm <- function( x, z, thr=-1:1 ) {
  n   <- length(x) ; n_thr <- length(thr)
  H   <- vector("list",n_thr)
  n_H <- pH <- V <- R <- s_pH <- s_V <- s_R <- rep(NA,n_thr)
  H[[1]]  <- which( x < thr[1] ) ; n_H[1] <- length(H[[1]])
  for(i in 2:n_thr) { H[[i]] <- which( thr[i-1] <= x & x < thr[i])
                      n_H[i] <- length(H[[i]]) } ; n_NotH <- n - sum(n_H)
  H.all   <- which( x < thr[n_thr] )
  pH      <- n_H / n             ; s_pH      <- sqrt( pH*(1-pH) / n )
```

```
  Ez_NotH <- mean( z[-H.all] )   ; s_Ez_NotH <- sqrt( var(z[-H.all] ) / n_NotH )
  for(i in 1:n_thr) {
    Ez_Hi <- mean( z[ H[[i]] ] ) ; s_Ez_Hi <- sqrt( var(z[ H[[i]]]) /  n_H[i] )
    V[i]  <- Ez_NotH - Ez_Hi     ; s_V[i]  <- sqrt( s_Ez_NotH^2 + s_Ez_Hi^2 ) }
  R       <- pH * V
  s_R     <- sqrt( s_pH^2 * s_V^2 + s_pH^2 * V^2 + pH^2 * s_V^2 )
  R.sum   <- sum(R) ; pH.sum <- sum(pH) ; V.wsum <- R.sum / pH.sum
  return( list( sum = c( pH.sum=pH.sum, V.wsum=V.wsum, R.sum=R.sum ),
                seq = cbind( thr, pH, V, R, s_pH, s_V, s_R ) ) )
}
```

5.3 Comparison of Single- and Multi-threshold PRA

Here's a quick check for our sparse dataset and the first linear one:

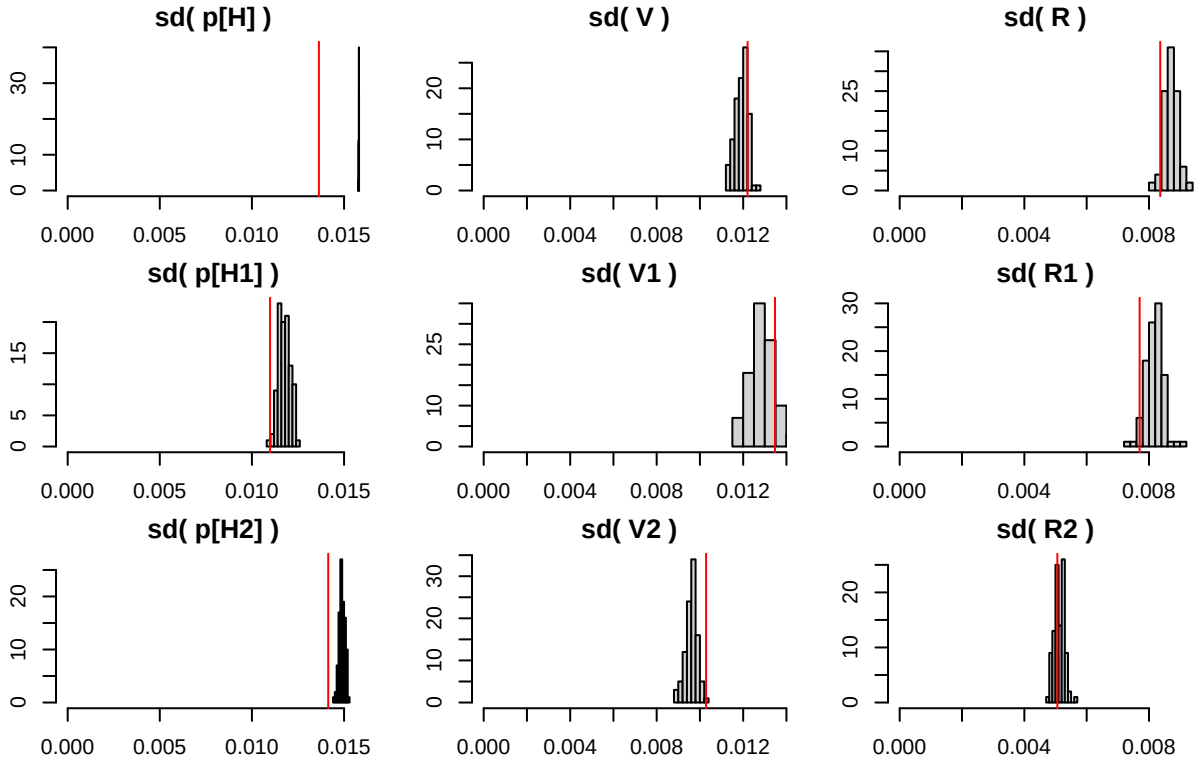
```
  PRA ( x_4, z_4,        0.5  )
>        pH         V         R      s_pH       s_V       s_R
> 0.7500000 1.4468204 1.0851153 0.2165064        NA        NA
  PRAm( x_4, z_4, c(-0.5,0.5) )
> $sum
>   pH.sum   V.wsum    R.sum
> 0.750000 1.446820 1.085115
>
> $seq
>       thr   pH        V         R      s_pH s_V s_R
> [1,] -0.5 0.50 1.176670 0.5883349 0.2500000  NA  NA
> [2,]  0.5 0.25 1.987122 0.4967805 0.2165064  NA  NA

  PRA ( l_xz.L[[1]][,1], l_xz.L[[1]][,2], 1 )
>         pH          V         R       s_pH        s_V        s_R
> 0.83600000 0.89571381 0.74881675 0.01170914 0.07718265 0.06537776
  PRAm( l_xz.L[[1]][,1], l_xz.L[[1]][,2]    )
> $sum
>    pH.sum    V.wsum     R.sum
> 0.8360000 0.8957138 0.7488167
>
> $seq
>      thr    pH         V         R       s_pH        s_V        s_R
> [1,]  -1 0.182 1.4620568 0.2660943 0.01220148 0.09645053 0.02505527
> [2,]   0 0.327 1.0057281 0.3288731 0.01483479 0.08663285 0.03204341
> [3,]   1 0.327 0.4704872 0.1538493 0.01483479 0.08352156 0.02821649
```

We verify more thoroughly that the UQ-formulas in the code chunks for PRA and PRAm are correct by using our full collection of nonlinear datasets `l_xz.NL`.

```
  l_xz <- l_xz.NL ; n_d <- length(l_xz)
```

```
  thr.seq   <- c( 0.5, 1.5 )

  PRA.tbl   <- t( sapply( 1:n_d, function(d){
    PRA ( l_xz[[d]][,"x"], l_xz[[d]][,"z"], thr=thr.seq[2]) } ) )
```

```
  PRAm.tbl1 <- t( sapply( 1:n_d, function(d){
    PRAm( l_xz[[d]][,"x"], l_xz[[d]][,"z"], thr=thr.seq)$seq[1,] } ) )
  PRAm.tbl2 <- t( sapply( 1:n_d, function(d){
    PRAm( l_xz[[d]][,"x"], l_xz[[d]][,"z"], thr=thr.seq)$seq[2,] } ) )

  # Standard deviation of the PRA-results over the n_d samples
  s_pH <- sd(PRA.tbl[,"pH"]) ; s_V <- sd(PRA.tbl[,"V"]) ; s_R <- sd(PRA.tbl[,"R"])
  c( s_pH , s_V , s_R )
> [1] 0.013633956 0.012203535 0.008360964

  # Standard deviation of the PRAm-results over the n_d samples
  # the ".1" and ".2" refer to (x <= thr[1]) and (thr[1] < x <= thr[2]) resp.
  s_pH1 <- sd(PRAm.tbl1[,"pH"]) ; s_pH2 <- sd(PRAm.tbl2[,"pH"])
  s_V1  <- sd(PRAm.tbl1[,"V" ]) ; s_V2  <- sd(PRAm.tbl2[,"V" ])
  s_R1  <- sd(PRAm.tbl1[,"R" ]) ; s_R2  <- sd(PRAm.tbl2[,"R" ])
  rbind( c( thr.seq[1], s_pH1 , s_V1 , s_R1 ),
         c( thr.seq[2], s_pH2 , s_V2 , s_R2 ) )
>      [,1]       [,2]       [,3]        [,4]
> [1,]  0.5 0.01098565 0.01347099 0.007699018
> [2,]  1.5 0.01414206 0.01028601 0.005059894
```



This all looks fine.

5.4 Bayesian alternative for hazard probability

Our code above for sampling-based PRA used the standard unbiased (and in case of pH also maximum-likelihood) estimators for pH, V, R, and their uncertainties. A Bayesian alternative would be to assign a prior probability distribution first and use the data to update to a posterior. For pH this could be done analytically

if we assign a Beta (for single-threshold PRA) or Dirichlet (multi-threshold PRA) prior. Analytical solutions then exist because Beta-binomial and Dirichlet-multinomial prior-likelihood distribution pairs are conjugate. Posterior uncertainty for pH could then be expressed using credibility intervals or less precisely but more simply as the standard deviation(s) associated with the posterior Beta resp. Dirichlet distributions. And s_pH was, at least for single-threshold sampling-based PRA, indeed calculated in that way by Marcel Van Oijen and Zavala (2019) and in the first edition of Marcel Van Oijen and Brewer (2022). However, in those references only the uncertainty associated with pH was calculated in that way but the estimate of pH itself was not the posterior Beta mean but the maximum likelihood estimator. That inconsistency is resolved here by using standard sampling theory to quantify the posterior uncertainty for pH (as $s_{pH} = \sqrt{p_H(1-p_H)\,/\,n}$, so not using the Bayesian approach. That makes the calculation of pH and s_pH now consistent with the approach we took for V and R. Comparable standard sampling theory was used there too to quantify uncertainty associated with the conditional expectations for z that we need to calculate V and its uncertainty (as for example $s_{E[z_H]} = s_{z_H}/\sqrt{n_H}$).

Here is code for the Beta approach to Bayesian single-threshold hazard probability estimation, which we could use when no full PRA is required.

```
pH_Be <- function( x, thr=0 ) {
  n  <- length(x) ; H    <- which(x < thr) ; n_H <- length(H)
  a  <- 1 + n_H   ; b    <- 1 + n - n_H
  pH <- a / (a+b) ; s_pH <- sqrt( a*b/(a+b+1) ) / (a+b)
  return( c( pH=pH, s_pH=s_pH ) )
}
```

Now let's develop code for the Dirichlet approach to Bayesian multi-threshold hazard probability estimation. We model $p[x]$ with a k-dimensional Dirichlet distribution. A priori, before considering the data, we set this distribution to be uniform, i.e. $Di_k[\mathbf{1}_k]$. This permits a conjugate Bayesian approach (Dirichlet prior and multinomial likelihood are conjugate), and the posterior probability distribution is $Di_k[\mathbf{1}_k + \{n_i\}]$, where the $\{n_i\}$ are the numbers of observations in our k disjoint intervals of x.

The Dirichlet-multinomial prior-likelihood pair is a generalisation of the Beta-binomial pair that we could have used for single-threshold hazard probability estimation. For example, a $Be[a,b]$ distribution is equivalent to a $Di_2[a,b]$ distribution.

The following code allows for threshold sequences of any length `n_thr`. The probabilities for the `n_thr` thresholds and non-hazardousness are given a uniform Dirichlet prior, which is updated to the posterior by counting the occurrences of hazards and non-hazardousness in the dataset. The posterior mean magnitude and uncertainty of the various hazard-levels are simple functions of the Dirichlet hyperparameters.

```
pHm_Di <- function( x, thr=-1:1 ){
  n      <- length(x) ; n_thr <- length(thr)
  H      <- vector("list",n_thr)
  n_H    <- pH <- s_pH <- rep(NA,n_thr)
  H[[1]] <- which( x < thr[1] ) ; n_H[1] <- length(H[[1]])
  for(i in 2:n_thr) { H[[i]] <- which( thr[i-1] <= x & x < thr[i])
                      n_H[i] <- length(H[[i]]) }
  n_NotH <- n - sum(n_H)
  a0     <- rep(1,n_thr+1) ; a1 <- a0 + c(n_H,n_NotH) ; A1 <- sum(a1)
  pH     <- a1[1:n_thr] / A1
  s_pH   <- sqrt( (a1[1:n_thr]/A1) * (1 - a1[1:n_thr]/A1) / (A1+1) )
return( cbind( pH, s_pH ) )
}
```

Differences in PRA between using the Bayesian approach to pH compared to the standard maximum likelihood sampling approach will generally be minor. One difference is that the Bayesian method will never

assign a value of zero to the hazard probability, even if no datapoint is in the hazardous regime. Except for small datasets with near-zero hazard occurrences, PRA will hardly be affected.

```
  PRAm  ( x_4, z_4, c(-0.5,0.5) )$seq[,c("pH","s_pH")]
>        pH      s_pH
> [1,] 0.50 0.2500000
> [2,] 0.25 0.2165064
  pHm_Di( x_4     , c(-0.5,0.5) )
>             pH      s_pH
> [1,] 0.4285714 0.1749636
> [2,] 0.2857143 0.1597191

  PRAm  ( l_xz.L[[1]][,1], l_xz.L[[1]][,2])$seq[,c("pH","s_pH")]
>         pH       s_pH
> [1,] 0.182 0.01220148
> [2,] 0.327 0.01483479
> [3,] 0.327 0.01483479
  pHm_Di( l_xz.L[[1]][,1]                )
>             pH       s_pH
> [1,] 0.1822709 0.01217812
> [2,] 0.3266932 0.01479427
> [3,] 0.3266932 0.01479427
```

6 Distribution-Based PRA

We must distinguish the following cases regarding the joint distribution $p[x, z]$, with increasing prior uncertainty:

- 1. Distribution known exactly (never realistic!),
- 2. Distribution known, apart from one or more hyperparameters (e.g. bivariate Gaussian with unknown mean and covariance matrix; this example happens to have a conjugate solution),
- 3. Distribution unknown (e.g. major uncertainty about tails: use copulas?)

Our examples are mainly for case 2. In particular we often assume that the data derive from a bivariate Gaussian. (Better would be to assume that case 3 applies.)

When using distribution-based PRA, the *sampling* uncertainties can be reduced to zero, but not those associated with choosing and fitting a distribution.

We tend to study the bivariate distribution p[x,z] here, but note that for *monovariate* x (or z) we have standard expressions for the uncertainty about population mean and variance as a function of the sample mean and sample variance. Unbiased estimators for the population mean and population variance are the sample mean ($m = \frac{1}{n}\sum x_i$) and the sample variance ($s^2 = \frac{1}{n-1}\sum(x_i - m)^2$). Uncertainty about m as our estimate for the population mean is given by the standard error of the mean calculated as $SEM = s/\sqrt{n}$ or the variance of the sample mean $Var[m] = s^2/n$. Uncertainty about s^2 as our estimate for the population variance is given by the variance of the sample variance calculated, *assuming the population distribution is normal*, as $var[s^2] = s^2\sqrt{\frac{2}{n-1}}$.

In fact, also for our bivariate distribution p[x,z] the sample mean $m = (m_x, m_z)^\top$ and the sample covariance matrix $cov(x, z)$ (with all four matrix entries calculated as sums of squares divided by $n - 1$) are unbiased estimators. But for that distribution, expressing the uncertainty about our estimators is harder to quantify. We can use a Bayesian approach, and if we assume a bivariate Gaussian as the underlying distribution, a conjugate one.

When we have a joint probability distribution for {x,z}, we can remove *sampling* uncertainty by taking a very large sample from the distribution. That will reduce uncertainty about p[H], V and R to nearly zero. But uncertainty about loss will of course remain, even for any specific value of x. So large uncertainty about x and z does not preclude low uncertainty about hazard probability, system vulnerability and risk. You know exactly what your chances are with roulette, and what your expected gains are, but that does not preclude large uncertainty about gains in a few turnings of the wheel. Is this a motivation to study p[loss]?

6.1 Conditional means and variances

This section deals only with the bivariate Gaussian p[x,z] for which formulas for the conditional means and variances were given by Marcel Van Oijen and Brewer (2022) Eqs 2.7 and 4.5. Here is a new implementation of those formulas (including an error correction for the conditional variance: "1 + .." becoming "Vz + .."):

```
EzVz_Gauss <- function( m.=m, S.=S, thr.=thr ) {
  mx     <- m.[1]   ; mz <- m.[2]
  Vx     <- S.[1,1] ; Vz <- S.[2,2]   ; Vxz  <- S.[1,2] ; r <- Vxz/sqrt(Vx*Vz)
  pthr   <- dnorm(thr., mx, sqrt(Vx)) ; Fthr <- pnorm(thr., mx, sqrt(Vx))
  Ez_xlo <- mz - Vxz * pthr / Fthr
  Ez_xhi <- mz + Vxz * pthr / (1-Fthr)
  Vz_xlo <- Vz + r * (thr.-mx) * (Ez_xlo-mz) - (Ez_xlo-mz)^2
  Vz_xhi <- Vz + r * (thr.-mx) * (Ez_xhi-mz) - (Ez_xhi-mz)^2
  result <- c( Ez_xlo, Ez_xhi, Vz_xlo, Vz_xhi )
  names ( result ) <- c( "Ez_xlo", "Ez_xhi", "Vz_xlo", "Vz_xhi" )
  return( result ) }
```

We can do a quick sense check of these formulas by setting x=z and checking that the formulas then reduce to the mean and variance for the monovariate truncated normal distribution:

- Eq. 2.7 => E[z|z<thr] = m - V * pthr / Fthr
- Eq. 4.5 => Var[z|z<thr] = V - (thr-m) * V * pthr/Fthr - V^2 * (pthr/Fthr)^2

These are indeed the known equations for the truncated normal. And for a numerical check, we take a large sample from the standard bivariate Gaussian with $r = 0.5$.

```
  set.seed(1)
  mu  <- c(0,0) ; Sigma <- diag(1,2) ; Sigma[1,2] <- Sigma[2,1] <- 0.5
  xz  <- rmvnorm( 1e6, mu, Sigma ) ; x <- xz[,1] ; z <- xz[,2]

  thr <- -1

  EzVz_Gauss( mu, Sigma, thr )
>     Ez_xlo     Ez_xhi     Vz_xlo     Vz_xhi
> -0.7625676  0.1438000  0.7997744  0.9074216
  c( mean( z[x< thr] ), mean( z[x>=thr] ), var( z[x< thr] ), var( z[x>=thr] ) )
> [1] -0.7598924  0.1436785  0.8010606  0.9079160
```

The numerical results confirm that the formulas work well.

6.2 PRA

The above formulas for conditional means and variances of the bivariate Gaussian lead to very simple equations for PRA (Marcel Van Oijen and Brewer (2022) p.14, Eqs 2.8), which we implement in the following R-function:

```
  PRA0_Gauss <- function( m.=m, S.=S, thr.=thr ) {
    mx <- m.[1] ; sx <- sqrt(S.[1,1]) ; Vxz <- S.[1,2]
    pH <- pnorm( thr., mx, sx )
    V  <- Vxz * dnorm(thr., mx, sx) / (pH * (1-pH))
    R  <- pH * V
    return( c( pH=pH, V=V, R=R ) ) }
```

So whenever we are certain that the joint distribution $p[x, z]$ is well represented by $N[\mu, \Sigma]$, the function call `PRA_Gauss(mu,Sigma,thr)` would give us our PRA. But generally we only have a sample from $\{x, z\}$ and even if we are convinced that a bivariate Gaussian is suitable, we then still have uncertainty about the hyperparameters of the distribution. So we need a probability distribution for the hyperparameters. For this, we use a conjugate Bayesian approach where the posterior distribution can be calculated analytically.

Say that our data are $y = \{x_i, z_i\}, i = 1..n$ with $m = (\overline{x}, \overline{z})$ and $S = cov(x, z)$. If we assign an uninformative Jeffreys prior $p[\mu, \Sigma] \propto |\Sigma|^{-3/2}$, then the posterior is an inverse Wishart distribution for Σ times a conditionally normal distribution for μ (e.g. Gelman et al. (2013)). Concretely, $p[\Sigma|y] = IWi_{n-1}(S^{-1})$ and $p[\mu|\Sigma, y] = N[m, \Sigma/n]$.

Here are three covariance matrices generated by the posterior inverse Wishart distribution found by conditioning on the first linear dataset:

```
  xz <- l_xz.L[[1]] ; n <- dim(xz)[1] ; S <- cov(xz)
```

$$\begin{bmatrix} 1.17 & 0.57 \\ 0.57 & 1.04 \end{bmatrix} \begin{bmatrix} 1.09 & 0.53 \\ 0.53 & 1 \end{bmatrix} \begin{bmatrix} 1.18 & 0.54 \\ 0.54 & 0.93 \end{bmatrix}$$

We conclude that after the Bayesian updating, we can generate a large number of realisations μ, Σ from the posterior and carry out a distribution-based PRA for each, thus permitting UQ. Here is R-code:

```
PRA_Gauss <- function( m.=m, S.=S, n.=n, thr.=thr ) {
  pH <- V <- R <- rep( NA, 1e3 )
  for(j in 1:1e3){
    S     <- riwish( n.-1, S. * (n.-1) ) ; m <- rmvnorm( 1, m., S/n. )
    PRA   <- PRA0_Gauss( m, S, thr. )
    pH[j] <- PRA["pH"] ; V[j] <- PRA["V"] ; R[j] <- PRA["R"]
  }
  return( c( pH=mean(pH),   V=mean(V),   R=mean(R),
             s_pH=sd  (pH), s_V=sd  (V), s_R=sd  (R) ) )
}
```

So this code gives us distribution-based single-threshold PRA with UQ. But note that the method only works if the covariance matrix S of the data is positive definite as required by the inverse Wishart distribution. [So we can for example not apply it to an ultra-sparse dataset with n=2.]

Here is an example with the first linear dataset:

```
l_xz <- l_xz.L    ; n_d <- length(l_xz) ; n <- dim(l_xz[[1]])[1]
xz   <- l_xz[[1]] ; m   <- colMeans(xz) ; S <- cov(xz)
x    <- xz[,1]    ; z   <- xz[,2]

PRA_Gauss( m, S, n, -1 )
>          pH           V           R        s_pH         s_V         s_R
> 0.183235924 0.897838352 0.164555092 0.009805117 0.053382045 0.013686672
```

6.2.1 PRAm

Now we implement Gaussian-distribution-based multi-threshold PRA.

```
PRAm_Gauss <- function( m.=m, S.=S, n.=n, thr.=-1:0 ) {
  n_thr <- length(thr.)
  pH    <- V <- R <- s_pH <- s_V <- s_R <- rep(NA,n_thr)
  for(i in 1:n_thr){
    pHj <- Vj <- Rj <- rep( NA, 1e3 )
    for(j in 1:1e3){
      S       <- riwish( n.-1, S. * (n.-1) ) ; m <- rmvnorm( 1, m., S/n. )
      mx      <- m[1] ; sx <- sqrt(S[1,1]) ; Vxz <- S[1,2]
      Ez_NotH <- EzVz_Gauss(m,S,thr.[n_thr])["Ez_xhi"]
      if(i==1){
        pHj[j] <- pnorm( thr.[i], mx, sx )
        Ez_Hi  <- EzVz_Gauss(m,S,thr.[i])["Ez_xlo"]
      }else{
        pHj[j] <- pnorm( thr.[i], mx, sx ) - pnorm( thr.[i-1], mx, sx )
        Ez_Hi  <- (
          pnorm(thr.[i  ],mx,sx) * EzVz_Gauss(m,S,thr.[i  ])["Ez_xlo"] -
          pnorm(thr.[i-1],mx,sx) * EzVz_Gauss(m,S,thr.[i-1])["Ez_xlo"] ) / pHj[j]
      }
      Vj[j] <- Ez_NotH - Ez_Hi
```

```
      Rj[j] <- pHj[j] * Vj[j]
    }
    pH[i]   <- mean(pHj) ;   V[i] <- mean(Vj) ;   R[i] <- mean(Rj)
    s_pH[i] <- sd  (pHj) ; s_V[i] <- sd  (Vj) ; s_R[i] <- sd  (Rj)
  }
  R.sum <- sum(R) ; pH.sum <- sum(pH) ; V.wsum <- R.sum / pH.sum
  return( list( sum = c( pH.sum=pH.sum, V.wsum=V.wsum, R.sum=R.sum ),
                seq = cbind( thr., pH, V, R, s_pH, s_V, s_R ) ) )
}

PRAm_Gauss( m, S, n, -2:-1 )
> $sum
>    pH.sum    V.wsum     R.sum
> 0.1833924 0.8988428 0.1648410
>
> $seq
>      thr.          pH         V         R        s_pH        s_V         s_R
> [1,]   -2 0.03260255 1.2950137 0.0422363 0.003798200 0.07168798 0.005581290
> [2,]   -1 0.15078987 0.8133617 0.1226047 0.006449645 0.04719842 0.008219856
```

6.3 Alternative approaches for single-threshold PRA

We can carry out the distribution-based single-threshold PRA in various ways. The following methods 2 to 5 are alternatives to the method presented above.

6.3.1 Method 2

Above we used Bayesian conjugate updating for the joint distribution of the mean vector μ and covariance matrix Σ of the bivariate Gaussian that generated our data. But there is also a Bayesian conjugate approach for the joint distribution of μ and the *precision matrix* $W = \Sigma^{-1}$. That requires a Normal-Wishart prior distribution.

Let's denote the mean as m and the precision matrix as W. The prior for the joint distribution $p[m, W] = p[m|W]\, p[W] = N[m|m_0, k_0\, W]\, Wi[W|a_0, B_0]$. We denote the data as $y = \{x_i, z_i\}, i = 1..N$.

The posterior then is:

- $p[m, W|y] = p[m|W, y]\, p[W|y] = N[m|m_N, k_N\, W]\, Wi[W|a_N, B_N] = NWi[m, W|m_N, k_N, a_N, b_N]$,
- where $m_N = (k_0 m_0 + N\bar{y})/k_N, \quad k_N = k_0 + N, \quad a_N = a_0 + N/2, \quad B_N = B_0 + (\bar{S} + (\bar{y} - m_0)(\bar{y} - m_0)^\top k_0/k_N)N/2$,
- and $\bar{y} = 1/N \sum_{i=1}^{N} y_i \quad \bar{S} = 1/N \sum_{i=1}^{N} (y_i - \bar{y})(y_i - \bar{y})^\top$.

We encode these equations in the following R-function `NWiPost()` whose input will be the data (m, S and n) and prior hyperparameter settings for the NWi distribution. The function output will be the posterior hyperparameter settings. From which we derive the expectation values (posterior means) of m and W by $Em_y = m_N$ and $EW_y = a_N\, B_N^{-1}$.

```
NWiPost <- function( m.=m, S.=S, n.=n,
                     m0=rep(0,2), k0=0.01, a0=2, B0=diag(2) ) {
  Sy <- S. * (n.-1)/n.
  kN <- k0 + n.    ; mN <- (k0*m0 + n.*m.) / kN
  aN <- a0 + n./2 ; BN <- B0 + (n./2)*( Sy + (k0/kN) * (m.-m0)%*%t(m.-m0) )
```

```
  return( list( mN=mN, kN=kN, aN=aN, BN=BN ) )
}
```

Here is the R-function that uses the posterior NWi-hyperparameter values to generate pairs of means and variances for the bivariate Gaussian.

```
rNWi <- function( n, m=rep(0,2), k=0.01, a=2, B=diag(2) ) {
  mi <- matrix(NA,nrow=n,ncol=2) ; Si <- array( NA, dim=c(2,2,n) )
  for(i in 1:n){
    Wi      <- rWishart( 1, a, solve(B) )[,,1]
    Si[,,i] <- solve( Wi )
    mi[i,]  <- rmvnorm ( 1, m , Si[,,i] / k )
  }
  return( list( m=mi, S=Si ) )
}
```

Putting it all together, this leads to the following alternative code for distribution-based PRA, which we apply to the same linear dataset as before:

```
PRA_Gauss2 <- function( m.=m, S.=S, n.=n, thr.=thr ) {
  pH <- V <- R <- rep( NA, 1e3 )
  p  <- NWiPost(m.,S.,n.) ; mN <- p$mN ; kN <- p$kN ; aN <- p$aN ; BN <- p$BN
  for(j in 1:1e3){
    mS    <- rNWi(1, mN, kN, aN, BN) ; m <- mS$m ; S <- mS$S[,,1]
    PRA   <- PRA0_Gauss( m, S, thr. )
    pH[j] <- PRA["pH"] ; V[j] <- PRA["V"] ; R[j] <- PRA["R"]
  }
  return( c( pH=mean(pH),   V=mean(V),   R=mean(R),
             s_pH=sd  (pH), s_V=sd  (V), s_R=sd  (R) ) )
}
PRA_Gauss2( m, S, n, -1 )
>         pH          V          R       s_pH        s_V        s_R
> 0.18297054 0.89142014 0.16325440 0.01130245 0.07471363 0.01840363
```

We see that the results are quite similar as before. This is because our choices for the prior hyperparameters of the NWi-distribution implied a non-informative prior distribution.

6.3.2 Method 3

Let's now assume that m and S are close to μ resp. Σ, and generate hyperparameter values (m_i, S_i) directly from the sampling distribution of mean and covariance matrix for the bivariate Gaussian.

```
PRA_Gauss3 <- function( m.=m, S.=S, n.=n, thr.=thr ) {
  pH <- V <- R <- rep( NA, 1e3 )
  for(j in 1:1e3){
    m     <- rmvnorm( 1, m., S./n. )
    # Next line from: (n-1) * S. ~ Wishart_(n-1,Sigma)
    S     <- rWishart( 1, n.-1, S./(n.-1) )[,,1]
    PRA   <- PRA0_Gauss( m, S, thr. )
    pH[j] <- PRA["pH"] ; V[j] <- PRA["V"] ; R[j] <- PRA["R"]
  }
  return( c( pH=mean(pH),   V=mean(V),   R=mean(R),
```

```
            s_pH=sd  (pH), s_V=sd  (V), s_R=sd  (R) ) )
  }
  PRA_Gauss3( m, S, n, -1 )
>         pH          V          R       s_pH        s_V        s_R
> 0.18269105 0.89479208 0.16350239 0.01008611 0.05186816 0.01346221
```

6.3.3 Method 4

Thsi time, we assume that m and S are close to μ resp. Σ, generate other virtual datasets $\{x_i, z_i\}$ from $N[m, S]$, and calculate (m_i, S_i) for each of them.

```
  PRA_Gauss4 <- function( m.=m, S.=S, n.=n, thr.=thr ) {
    pH <- V <- R <- rep( NA, 1e3 )
    for(j in 1:1e3){
      xz    <- rmvnorm( n., m., S. ) ; m <- colMeans(xz) ; S <- cov(xz)
      PRA   <- PRA0_Gauss( m, S, thr. )
      pH[j] <- PRA["pH"] ; V[j] <- PRA["V"] ; R[j] <- PRA["R"]
    }
    return( c( pH=mean(pH),   V=mean(V),   R=mean(R),
               s_pH=sd  (pH), s_V=sd  (V), s_R=sd  (R) ) )
  }
  PRA_Gauss4( m, S, n, -1 )
>          pH           V           R        s_pH         s_V         s_R
> 0.183288994 0.898747515 0.164740915 0.009828498 0.052465052 0.013198935
```

6.3.4 Method 5

Whereas the methods above all aim to generate a representative set of Gaussian hyperparameters $\{m_i, S_i\}$ for PRA, the fifth method calculates sampling uncertainties for $z|x < thr$ and $z|x \geq thr$ directly. We plunge right in and give the code (just a minor modification of the code on p.28 of Marcel Van Oijen and Brewer (2022)) plus application to the same linear dataset as before:

```
  PRA_Gauss5 <- function( m.=m, S.=S, n.=n, thr.=thr ) {
    mx   <- m.[1] ; sx <- sqrt(S.[1,1]) ; Vxz <- S.[1,2]
    pH   <- pnorm( thr., mx, sx ) ; n_H <- n. * pH
    V    <- Vxz * dnorm(thr., mx, sx) / (pH * (1-pH))
    R    <- pH * V
      VarEz_H    <- EzVz_Gauss( m., S., thr. )[3] /  n_H
      VarEz_NotH <- EzVz_Gauss( m., S., thr. )[4] / (n.-n_H)
    s_pH <- sqrt( pH*(1-pH) / n. )
    s_V  <- sqrt( VarEz_NotH + VarEz_H )
    s_R  <- sqrt( s_pH^2*s_V^2 + s_pH^2*V^2 + pH^2*s_V^2 )
    result <- c( pH, V, R, s_pH, s_V, s_R )
    names ( result ) <- c( "pH", "V", "R", "s_pH", "s_V", "s_R" )
    return( result )
  }
  PRA_Gauss5( m, S, n, -1 )
>         pH          V          R       s_pH        s_V        s_R
> 0.18300985 0.89593393 0.16396473 0.01222772 0.07544447 0.01764947
```

This method effectively assumes that our dataset's m and S are equal to the true μ resp. Σ that we are after. That allows it to use expressions for the sampling distributions of $z|x < thr$ and $z|x \geq thr$ to directly

do the PRA and UQ (Marcel Van Oijen and Brewer (2022) p.28). We defined the relevant expressions for the conditional variances above, as part of function `EzVz_Gauss`. The method uses these to calculate an "expected PRA-uncertainty" for any sample size taken from that distribution, without actually doing any sampling at all.

6.3.5 Summary

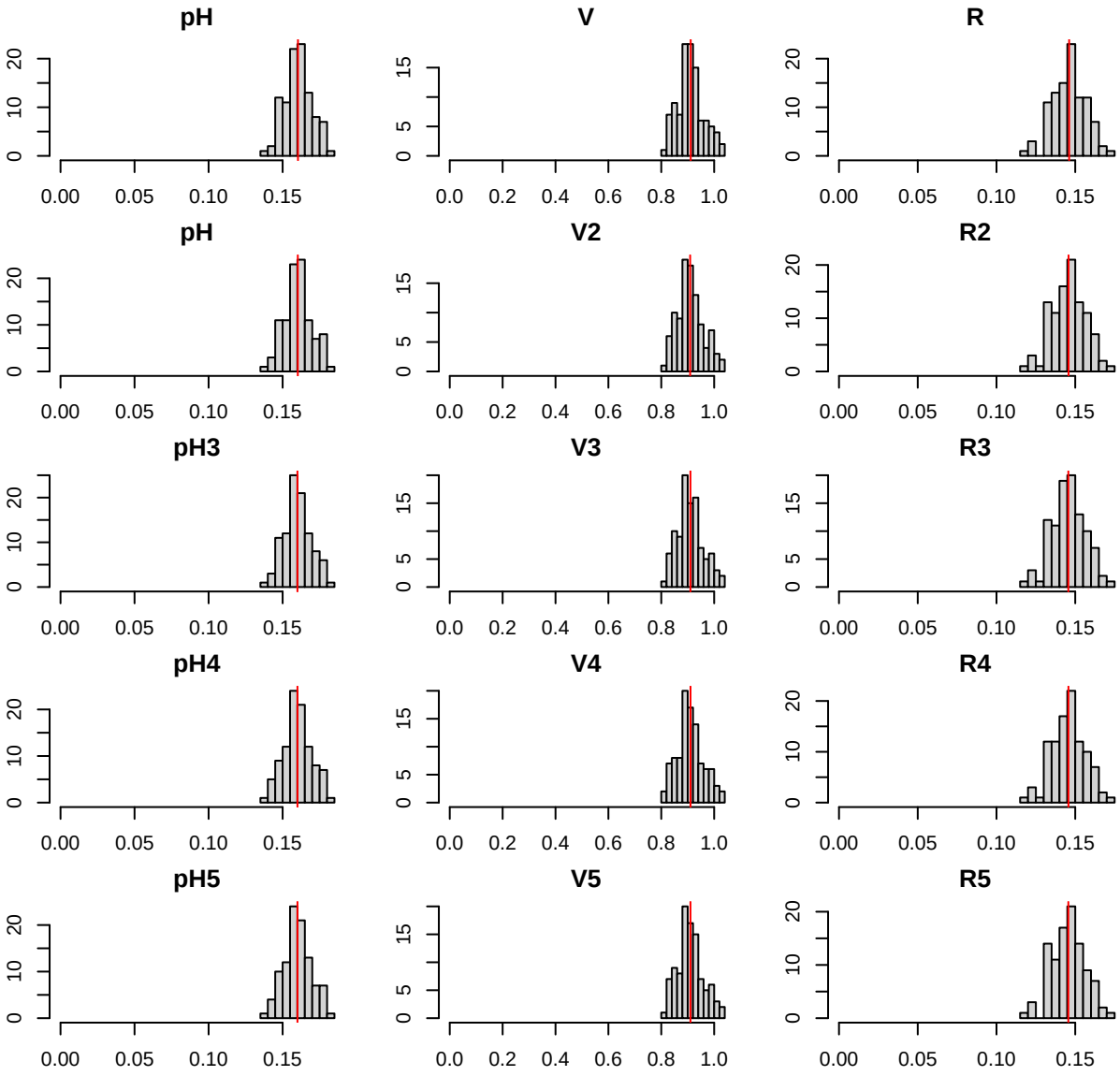
We have compared five different ways of carrying out UQ for distribution-based PRA, assuming that the data $\{x, z\}$ are generated by a bivariate Gaussian $N[\mu, \Sigma]$. We used the first linear dataset to test the methods.

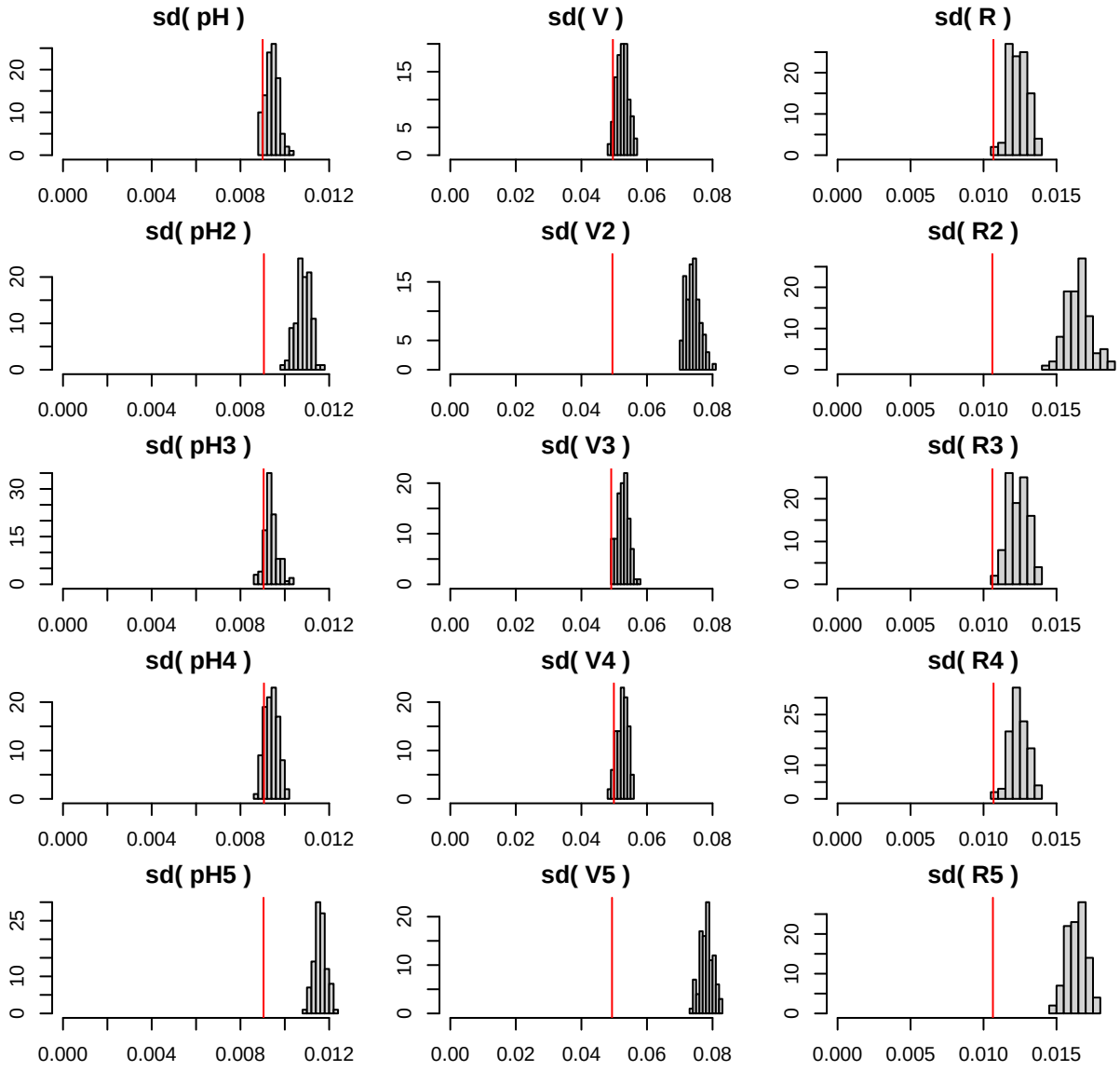
1. Assign a NIWi (Normal - Inverse Wishart) prior to (μ, Σ), use conjugate Bayesian updating to find $p[\mu, \Sigma|x, z]$, and generate a representative set of hyperparameter pairs (m_i, S_i),
2. Assign a NWi (Normal - Wishart) prior to $(\mu, W = \Sigma^{-1})$, use conjugate Bayesian updating to find $p[\mu, W|x, z]$, and generate a representative set of hyperparameter pairs (m_i, S_i),
3. Assume that m and S are close to μ resp. Σ, and generate hyperparameter values (m_i, S_i) directly from the sampling distribution of mean and covariance matrix for the bivariate Gaussian,
4. Assume that m and S are close to μ resp. Σ, generate other virtual datasets $\{x_i, z_i\}$ from $N[m, S]$, and calculate (m_i, S_i) for each of them,
5. Assume that m and S are equal to μ resp. Σ and use expressions for the sampling distributions of $z|x < thr$ and $z|x \geq thr$ to directly do the PRA and UQ (Marcel Van Oijen and Brewer (2022) p.28).

```
  set.seed(1)

  PRA( x, z, thr=-1 )
>         pH          V          R       s_pH        s_V        s_R
> 0.18200000 0.87193163 0.15869156 0.01220148 0.07483537 0.01730677

  rbind( PRA_Gauss ( m, S, n, -1 ),
         PRA_Gauss2( m, S, n, -1 ),
         PRA_Gauss3( m, S, n, -1 ),
         PRA_Gauss4( m, S, n, -1 ),
         PRA_Gauss5( m, S, n, -1 ) )
>             pH         V         R        s_pH        s_V        s_R
> [1,] 0.1831095 0.8964623 0.1641673 0.010237108 0.05345989 0.01361854
> [2,] 0.1835378 0.8987570 0.1650960 0.011660548 0.07114574 0.01812840
> [3,] 0.1835001 0.8952456 0.1643035 0.009915054 0.05169382 0.01335110
> [4,] 0.1830806 0.8966750 0.1641650 0.009896042 0.05221093 0.01305656
> [5,] 0.1830098 0.8959339 0.1639647 0.012227724 0.07544447 0.01764947

  PRAm_Gauss( m, S, n, -2:-1 )$seq
>      thr.         pH         V          R        s_pH        s_V         s_R
> [1,]   -2 0.03262083 1.2930054 0.04221427 0.003839046 0.07498527 0.005810898
> [2,]   -1 0.15048857 0.8105919 0.12196465 0.006525500 0.04761601 0.008615815
```

```
  PRA.tbl  <- t( sapply( 1:n_d, function(d){
    PRA_Gauss ( colMeans(l_xz.L[[d]]), cov(l_xz.L[[d]]), n, -1 ) } ) )
  PRA.tbl2 <- t( sapply( 1:n_d, function(d){
    PRA_Gauss2( colMeans(l_xz.L[[d]]), cov(l_xz.L[[d]]), n, -1 ) } ) )
  PRA.tbl3 <- t( sapply( 1:n_d, function(d){
    PRA_Gauss3( colMeans(l_xz.L[[d]]), cov(l_xz.L[[d]]), n, -1 ) } ) )
  PRA.tbl4 <- t( sapply( 1:n_d, function(d){
    PRA_Gauss4( colMeans(l_xz.L[[d]]), cov(l_xz.L[[d]]), n, -1 ) } ) )
```

```
  PRA.tbl5 <- t( sapply( 1:n_d, function(d){
    PRA_Gauss5( colMeans(l_xz.L[[d]]), cov(l_xz.L[[d]]), n, -1 ) } ) )

  # Mean PRA-results over the n_d samples
  m_pH  <- mean(PRA.tbl [,"pH"]) ; m_V  <- mean(PRA.tbl [,"V"]) ; m_R  <- mean(PRA.tbl [,"R"])
  m_pH2 <- mean(PRA.tbl2[,"pH"]) ; m_V2 <- mean(PRA.tbl2[,"V"]) ; m_R2 <- mean(PRA.tbl2[,"R"])
  m_pH3 <- mean(PRA.tbl3[,"pH"]) ; m_V3 <- mean(PRA.tbl3[,"V"]) ; m_R3 <- mean(PRA.tbl3[,"R"])
  m_pH4 <- mean(PRA.tbl4[,"pH"]) ; m_V4 <- mean(PRA.tbl4[,"V"]) ; m_R4 <- mean(PRA.tbl4[,"R"])
  m_pH5 <- mean(PRA.tbl5[,"pH"]) ; m_V5 <- mean(PRA.tbl5[,"V"]) ; m_R5 <- mean(PRA.tbl5[,"R"])
  # "True" mean of the PRA-results over the n_d samples
  rbind( c( m_pH , m_V , m_R  ), c( m_pH2, m_V2, m_R2 ), c( m_pH3, m_V3, m_R3 ),
         c( m_pH4, m_V4, m_R4 ), c( m_pH5, m_V5, m_R5 ) )
>           [,1]      [,2]      [,3]
> [1,] 0.1603655 0.9113649 0.1461329
> [2,] 0.1602513 0.9092742 0.1457689
> [3,] 0.1599949 0.9103209 0.1456253
> [4,] 0.1600097 0.9105589 0.1456790
> [5,] 0.1600187 0.9103723 0.1456266
  # "True" standard deviation of the PRA-results over the n_d samples
  s_pH  <- sd(PRA.tbl [,"pH"]) ; s_V  <- sd(PRA.tbl [,"V"]) ; s_R  <- sd(PRA.tbl [,"R"])
  s_pH2 <- sd(PRA.tbl2[,"pH"]) ; s_V2 <- sd(PRA.tbl2[,"V"]) ; s_R2 <- sd(PRA.tbl2[,"R"])
  s_pH3 <- sd(PRA.tbl3[,"pH"]) ; s_V3 <- sd(PRA.tbl3[,"V"]) ; s_R3 <- sd(PRA.tbl3[,"R"])
  s_pH4 <- sd(PRA.tbl4[,"pH"]) ; s_V4 <- sd(PRA.tbl4[,"V"]) ; s_R4 <- sd(PRA.tbl4[,"R"])
  s_pH5 <- sd(PRA.tbl5[,"pH"]) ; s_V5 <- sd(PRA.tbl5[,"V"]) ; s_R5 <- sd(PRA.tbl5[,"R"])
  rbind( c( s_pH , s_V , s_R  ), c( s_pH2, s_V2, s_R2 ), c( s_pH3, s_V3, s_R3 ),
         c( s_pH4, s_V4, s_R4 ), c( s_pH5, s_V5, s_R5 ) )
>            [,1]       [,2]       [,3]
> [1,] 0.009002199 0.04959882 0.01067862
> [2,] 0.009057648 0.04946223 0.01060462
> [3,] 0.009045364 0.04908991 0.01060553
> [4,] 0.009059516 0.04990815 0.01068453
> [5,] 0.009042411 0.04933911 0.01064197
```


pH
V
R
pH
V2
R2
pH3
V3
R3
pH4
V4
R4
pH5
V5
R5



6.4 Conditions for V being constant

Marcel Van Oijen and Brewer (2022) (Eq. 2.4) showed the following formula for the derivative of V with respect to the threshold t:

$$\frac{dV}{dt} = \frac{p[t]}{F_x[t](1-F_x[t])}\{E[z|x \geq t]F_x[t] - E[z|x=t] + E[z|x<t](1-F_x[t])\}. \quad (1)$$

We can check numerically that this equation provides the correct derivative of V. Here is the example of PRA on a bivariate Gaussian dataset:

```
m <- c(0,0) ; S <- matrix(c(1,0.5,0.5,1),nrow=2)

# PRAbook Eq 2.4
dVdthr2.4 <- function(m.=c(0,0), S.=matrix(c(1,0.5,0.5,1),nrow=2), thr.=0){
  mx      <- m.[1] ; sdx <- sqrt( S.[1,1] )
```

```
    mz      <- m.[1] ; sdz <- sqrt( S.[2,2] ) ; r <- S.[1,2] / (sdx * sdz)
    Ez_H    <- EzVz_Gauss( m., S., thr. )["Ez_xlo"]
    Ez_NotH <- EzVz_Gauss( m., S., thr. )["Ez_xhi"]
    Ez_thr  <- mz + r * (thr.-mx)
    pthr    <- dnorm( thr., mx, sdx)
    Fthr    <- pnorm( thr., mx, sdx)
    return( pthr/(Fthr*(1-Fthr)) * (Ez_NotH * Fthr - Ez_thr + Ez_H * (1-Fthr)) )
  }
  dVdthr2.4(m,S,1)
>    Ez_xhi
> 0.2152943

  # Numerical check
  ( ( EzVz_Gauss(m,S,1.001)["Ez_xhi"] - EzVz_Gauss(m,S,1.001)["Ez_xlo"] ) -
    ( EzVz_Gauss(m,S,1    )["Ez_xhi"] - EzVz_Gauss(m,S,1    )["Ez_xlo"] ) ) / 0.001
>    Ez_xhi
> 0.2153975
```

The derivative seems to be calculated correctly.

6.4.1 A proof

Eq. 2.4 of Marcel Van Oijen and Brewer (2022) implies that V is constant (independent of threshold value) whenever $F_x[x]$ is a linear transformation of $E[z|x]$. We prove that here for $E[z|x] = F_x[x]$ and leave the more general case of $E[z|x] = a + bF_x[x]$ to the reader. We begin by noting that for all probability distributions:

$$\begin{aligned} E[F_x[x]|x \geq t] &= \frac{1 + F_x[t]}{2}, \\ E[F_x[x]|x < t] &= \frac{F_x[t]}{2}. \end{aligned} \tag{2}$$

Therefore, writing $F = F_x[t], A = p_x[t]/(F * (1 - F))$, and using the last two formulas:

$$\begin{aligned} dV/dt &= A\left(E[z|x \geq t]F - F + E[z|x < t](1 - F)\right) \\ &= A\left(\frac{F + F^2}{2} - F + \frac{F - F^2}{2}\right) \\ &= 0. \end{aligned} \tag{3}$$

That completes the proof. We now generate datasets for which $E[z|x] = a + bF_x[x]$, each based on a different probability distribution. The procedure is (1): generate a large sample of $\{x_i\}$ from the probability distribution, (2) for each of these x-values calculate $E[z|x_i] = a + bF_x[x_i]$ fro arbitrary a and b, (3) calculate the value of z_i as that expectation plus some error drawn from a zero-mean distribution. We do this for the following five probability distributions: uniform, exponential, Beta, t-, and Gaussian. Additionally, we make a sixth dataset that like the fifth one uses a Gaussian $p[x]$ but instead of the Gaussian cumulative distribution function Φ it uses another sigmoid curve, namely the logistic.

Let's now check whether these generated datasets have a constant value of V across a wide range of threshold-values.

```
  n         <- 5e3

  x_U       <- runif(n)      ; z_U <- 0.5 + x_U/2              + rnorm(n,0,0.05)
```

```
thr.seq_U <- quantile( x_U, (1:19)/20 )
PRA.seq_U <- t( sapply( thr.seq_U, function(t){PRA(x_U,z_U,t)} ) )

x_E       <- rexp(n)      ; z_E <- 1   - exp(-x_E)/2       + rnorm(n,0,0.05)
thr.seq_E <- quantile( x_E, (1:19)/20 )
PRA.seq_E <- t( sapply( thr.seq_E, function(t){PRA(x_E,z_E,t)} ) )

x_B       <- rbeta(n,5,1) ; z_B <- 0.5 + pbeta(x_B,5,1)/2  + rnorm(n,0,0.05)
thr.seq_B <- quantile( x_B, (1:19)/20 )
PRA.seq_B <- t( sapply( thr.seq_B, function(t){PRA(x_B,z_B,t)} ) )

x_t       <- rt(n,1,30)   ; z_t <- 0.5 + pt(x_t,1,30)/2    + rnorm(n,0,0.05)
thr.seq_t <- quantile( x_t, (1:19)/20 )
PRA.seq_t <- t( sapply( thr.seq_t, function(t){PRA(x_t,z_t,t)} ) )

x_G       <- rnorm(n)     ; z_G <- 0.5 + pnorm(x_G)/2      + rnorm(n,0,0.05)
thr.seq_G <- quantile( x_G, (1:19)/20 )
PRA.seq_G <- t( sapply( thr.seq_G, function(t){PRA(x_G,z_G,t)} ) )

x_L       <- rnorm(n)     ; z_L <- 0.5 + 0.5/(1+exp(-2*x_L)) + rnorm(n,0,0.05)
thr.seq_L <- quantile( x_L, (1:19)/20 )
PRA.seq_L <- t( sapply( thr.seq_L, function(t){PRA(x_L,z_L,t)} ) )
```

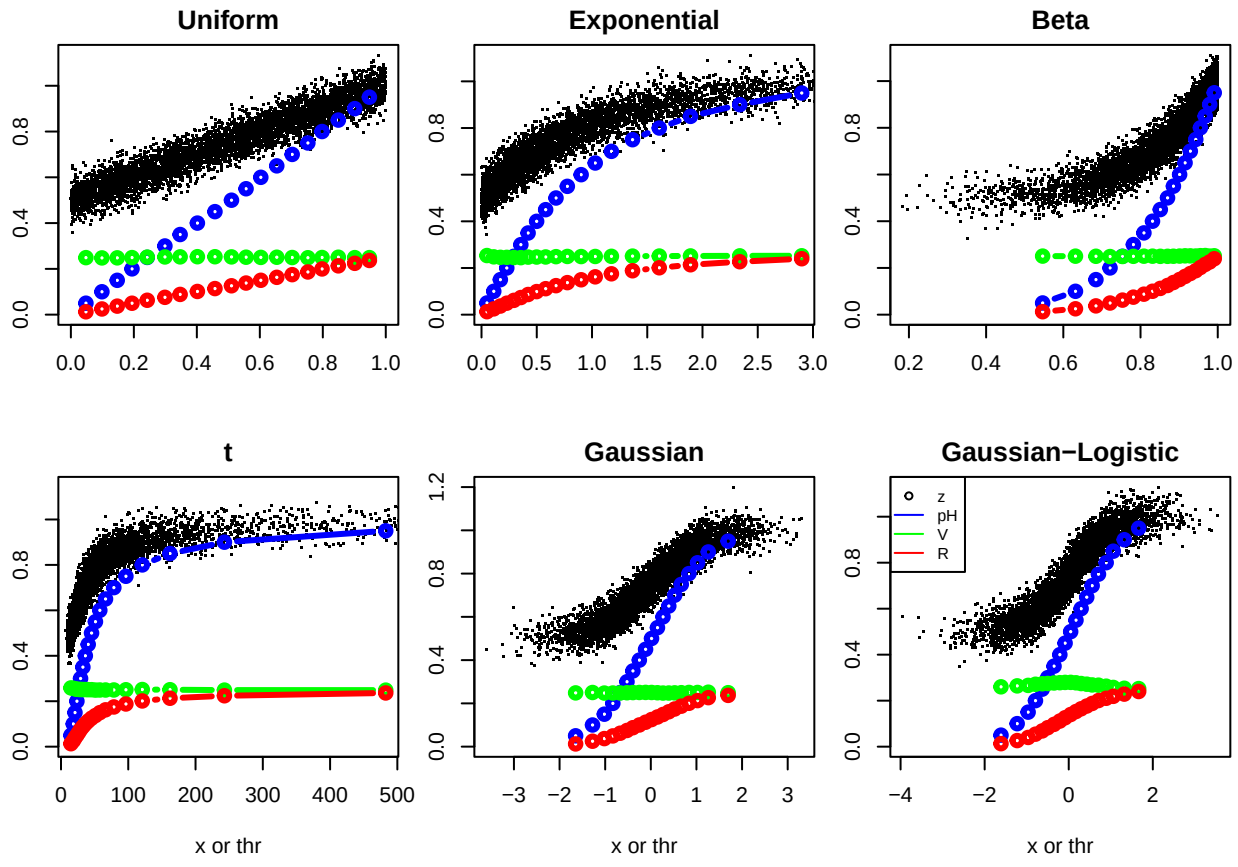


Figure 1: Six datasets {x,z} subjected to single-threshold sampling-based PRA. The blue, green and red lines show how pH, V and R change with choice of threshold.

The plots show that V is indeed constant in the first five plots, where $E[z|x]$ was a linear transformation of $F_x[x]$, and nearly constant in the last one where a close relative of $F_x[x]$ was used.

Note that multi-threshold PRA would still give for each interval a different value of V, as shown in the next figure.

```
PRAm.seq_U <- PRAm( x_U, z_U, thr.seq_U )$seq
PRAm.seq_E <- PRAm( x_E, z_E, thr.seq_E )$seq
PRAm.seq_B <- PRAm( x_B, z_B, thr.seq_B )$seq
PRAm.seq_t <- PRAm( x_t, z_t, thr.seq_t )$seq
PRAm.seq_G <- PRAm( x_G, z_G, thr.seq_G )$seq
PRAm.seq_L <- PRAm( x_L, z_L, thr.seq_L )$seq
```

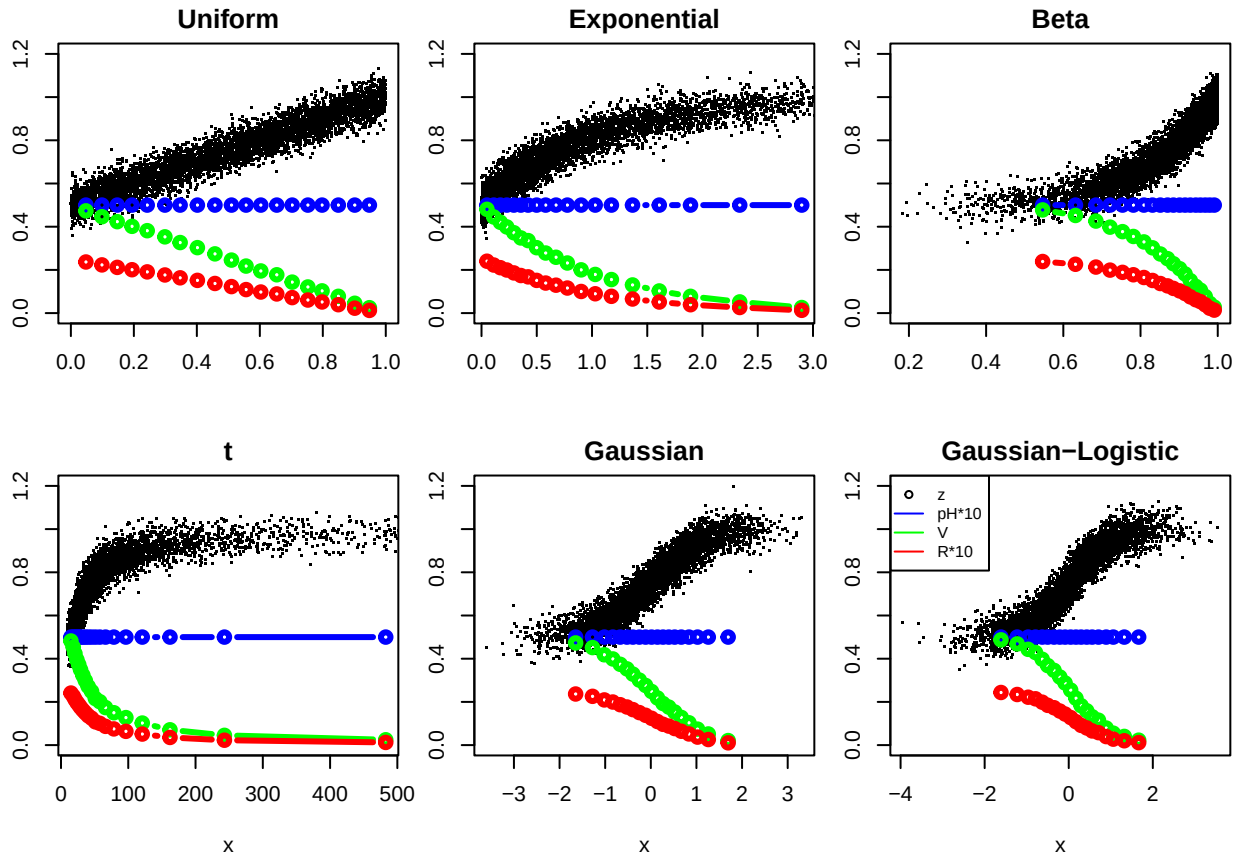


Figure 2: Six datasets {x,z} subjected to multi-threshold sampling-based PRA using 19 equal-probability x-intervals. The blue, green and red lines show how pH, V and R vary with x.

The six multi-threshold PRAs have a constant $p[H]$ rather than a constant V because we subdivided the hazardous region into nineteen equal-probability intervals (each 5%). Both V and R are continuously decreasing functions of x now, but the rate of decrease varies strongly between the six datasets. The one generated by the t-distribution has the strongest concentration of risk at the lower extreme of x.

The fact that $p[H]$ becomes constant (and R proportional to V), if we choose equal-probability x-intervals, makes the multi-threshold PRA fairly uninteresting. It is more informative to distribute the x-interval boundaries uniformly, as in the following figure.

```
thr.seqU_U  <- seq( quantile(x_U,1/10), quantile(x_U,9/10), length.out=9 )
thr.seqU_E  <- seq( quantile(x_E,1/10), quantile(x_E,9/10), length.out=9 )
thr.seqU_B  <- seq( quantile(x_B,1/10), quantile(x_B,9/10), length.out=9 )
thr.seqU_t  <- seq( quantile(x_t,1/10), quantile(x_t,9/10), length.out=9 )
thr.seqU_G  <- seq( quantile(x_G,1/10), quantile(x_G,9/10), length.out=9 )
thr.seqU_L  <- seq( quantile(x_L,1/10), quantile(x_L,9/10), length.out=9 )

PRAm.seqU_U <- PRAm( x_U, z_U, thr.seqU_U )$seq
PRAm.seqU_E <- PRAm( x_E, z_E, thr.seqU_E )$seq
PRAm.seqU_B <- PRAm( x_B, z_B, thr.seqU_B )$seq
PRAm.seqU_t <- PRAm( x_t, z_t, thr.seqU_t )$seq
PRAm.seqU_G <- PRAm( x_G, z_G, thr.seqU_G )$seq
PRAm.seqU_L <- PRAm( x_L, z_L, thr.seqU_L )$seq
```

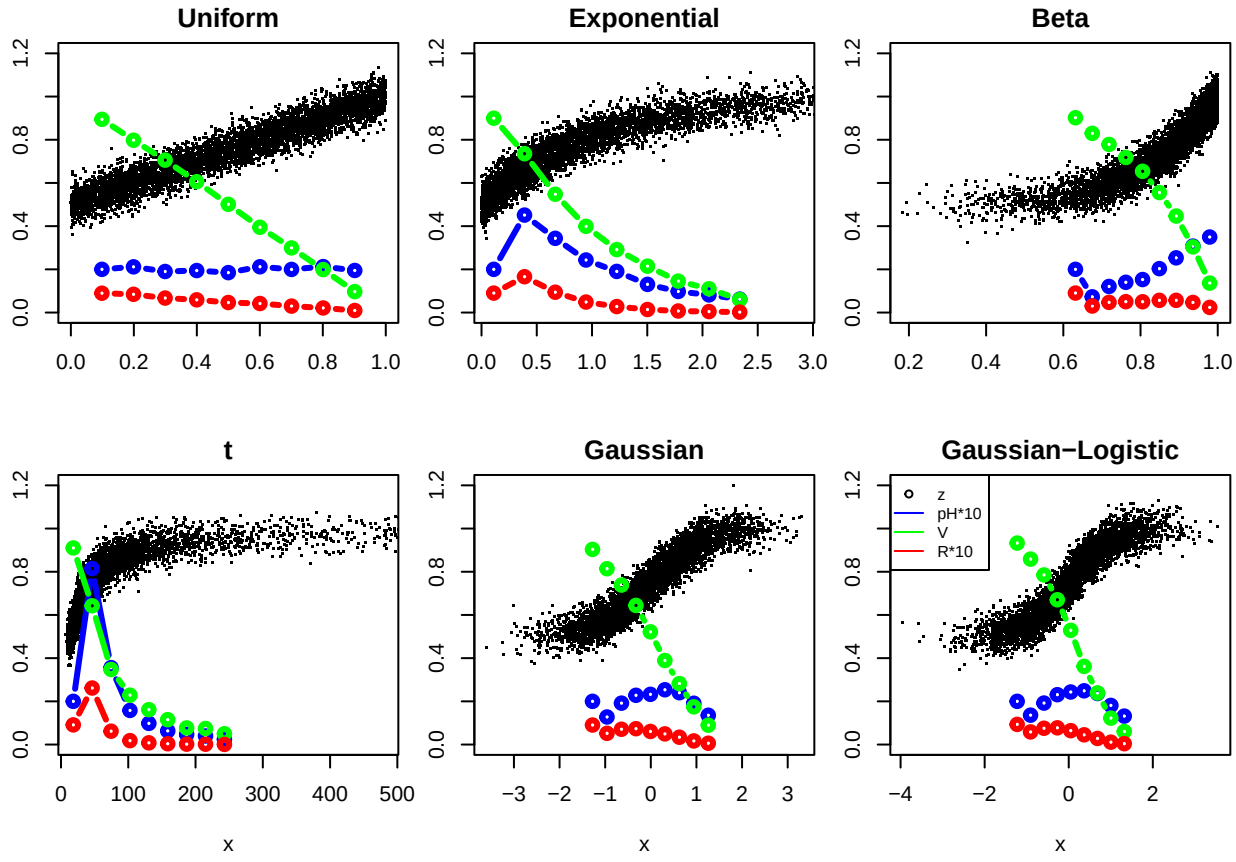


Figure 3: Six datasets {x,z} subjected to multi-threshold sampling-based PRA using 9 same-width x-intervals. The blue, green and red lines show how pH, V and R vary with x.

The last figure now shows more interesting x-interval-dependent R, and a more helpful decomposition into $p[H]$ and V. We see for example that the major risk in the t-distribution dataset is associated with the second interval, where $p[H]$ is highest and V second-highest.

7 Model-Based PRA

In model-based PRA, we parameterise a function $z = f(x, \theta)$. If we do this by means of Bayesian calibration, then we can sample from the posterior distribution for the parameters $p[\theta|\{x, z\}]$ to generate different virtual datasets. We then carry out sampling-based PRA on each of these datasets, and the average result of these PRAs, and the spread between the results, constitutes our model-based PRA with UQ.

Generally the Bayesian calibration will be carried out by MCMC, but in some very simple cases (e.g. linear regression with known measurement error variance for z) there is an analytical solution (Lindley and Smith (1972)).

8 Multiple Hazards

So far, in our exposition of PRA, we distinguished only a single hazard variable, but in practice there may well be multiple different forms of stress affecting the system. This can fairly easily be implemented in sampling-, distribution-, and model-based PRA. In this chapter, we shall show a couple of sampling-based PRA examples with two hazard variables.

The presence of multiple hazards implies that x-space, our "space of hazards", becomes complex. When considering a single hazard variable, we have just a 1-dimensional x-space (a line), and we distinguish two intervals on that x-line for single-threshold PRA (H vs. not H), more for multi-threshold PRA.

With n multiple hazards, x-space becomes n-dimensional. So even if we consider just a single threshold for each hazard, we then still need to distinguish 2^n disjoint x-regions to represent all possible combinations of the x-variables being in their hazardous interval or not. Only one of these regions represents the non-hazardous regime, all other regions have at least one variable in its hazardous state.

Example: multiple tree diseases that may occur together in a forest, or drought together with an insect pest.

Liu et al. (2020): their "risk probability" seems to refer to hazard probability, comprising two single hazards (runoff, N-pollution) whose joint distribution is expressed using copulas.

```
PRAi <- function( xz, thr=c(0,0) ) {
  x1  <- xz[,1]  ; x2 <- xz[,2] ; z <- xz[,3]
  n_c <- 2^2 - 1 ; n  <- length(x1)
  H   <- vector("list",n_c)
  n_H <- pH <- V <- R <- s_pH <- s_V <- s_R <- rep(NA,n_c)
  H[[1]]  <- which(x1 <  thr[1] & x2 <  thr[2]) ; n_H[1] <- length(H[[1]])
  H[[2]]  <- which(x1 <  thr[1] & x2 >= thr[2]) ; n_H[2] <- length(H[[2]])
  H[[3]]  <- which(x1 >= thr[1] & x2 <  thr[2]) ; n_H[3] <- length(H[[3]])
  NotH    <- which(x1 >= thr[1] & x2 >= thr[2]) ; n_NotH <- length(NotH)
  pH      <- n_H / n          ; s_pH      <- sqrt( pH*(1-pH) / n )
  Ez_NotH <- mean( z[NotH] ) ; s_Ez_NotH <- sqrt( var(z[NotH] ) / n_NotH )
  for(i in 1:n_c) {
    Ez_Hi   <-       mean( z[ H[[i]] ] )
    s_Ez_Hi <- sqrt( var ( z[ H[[i]] ] ) / n_H[i] )
    V[i]    <- Ez_NotH - Ez_Hi
    s_V[i]  <- sqrt( s_Ez_NotH^2 + s_Ez_Hi^2 ) }
  R     <- pH * V
  s_R   <- sqrt( s_pH^2 * s_V^2 + s_pH^2 * V^2 + pH^2 * s_V^2  )
  R.sum <- sum(R) ; pH.sum <- sum(pH) ; V.wsum <- R.sum / pH.sum
  return( list( sum = c( pH.sum=pH.sum, V.wsum=V.wsum, R.sum=R.sum ),
                cat = cbind( 1:3, pH, V, R, s_pH, s_V, s_R ) ) ) }
```

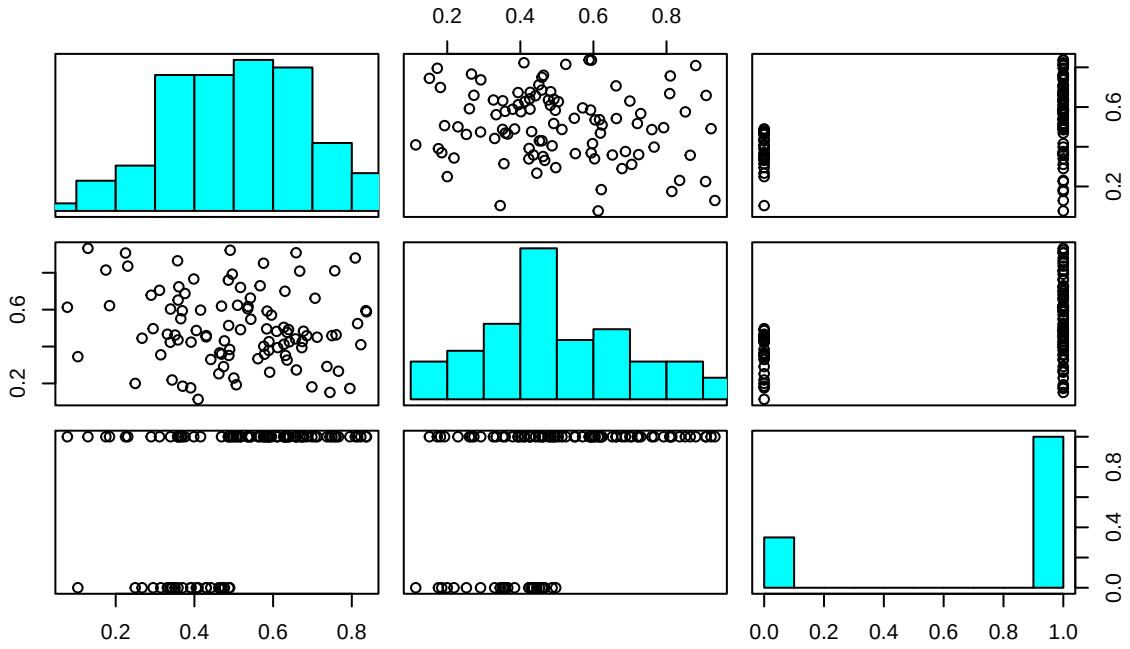
8.1 Trivariate Gaussian Dataset

```
PRAi( xz_G3, c(0,0) )
> $sum
>    pH.sum    V.wsum     R.sum
> 0.6610000 0.9509406 0.6285717
>
> $cat
>          pH         V         R       s_pH        s_V        s_R
> [1,] 1 0.342 1.2261530 0.4193443 0.01500120 0.07091047 0.03045639
```

```
> [2,] 2 0.160 0.6252191 0.1000351 0.01159310 0.08314691 0.01518055
> [3,] 3 0.159 0.6867442 0.1091923 0.01156369 0.08834228 0.01616817
```

8.2 Dataset with two uncorrelated hazard variables that exert correlated V

```
set.seed(1)

n  <- 1e2
x1 <- rbeta( n, 3, 3 ) ; x2 <- rbeta( n, 3, 3 )
z  <- as.integer( x1 >= 0.5 | x2 >= 0.5 )
xz <- cbind( x1, x2, z )

pairs( xz, diag.panel=panel.hist, label="" )
```




```
thr <- c( 0.5, 0.5 )

PRAi( xz, thr )
> $sum
> pH.sum V.wsum  R.sum
> 0.8000 0.3125 0.2500
>
> $cat
>          pH V    R       s_pH s_V        s_R
> [1,] 1 0.25 1 0.25 0.04330127   0 0.04330127
> [2,] 2 0.22 0 0.00 0.04142463   0 0.00000000
> [3,] 3 0.33 0 0.00 0.04702127   0 0.00000000
```

8.3 Bayesian alternative for hazard probability

Just like for multi-threshold PRA, we could choose a Bayesian method to estimate the various pH-values associated with different combinations of multiple hazard variables. So again using a Dirichlet prior and multinomial likelihood, but this time for data membership in 2^n disjoint n-dimensional regions rather than 1D threshold-intervals. And if we would be ambitious and aim for multi-threshold (n_{thr}) PRA for each of the n hazards, the number of disjoint n-dimensional regions in x-space would increase to $(n_{thr} + 1)^n$.

Sampling-based, single-threshold, two-hazard PRA that uses conjugate BC for the hazard distribution (Dirichlet prior, multinomial likelihood) leads to very similar results as the fully sampling-theory-based approach that we implemented in R-function `PRAi` (results not shown).

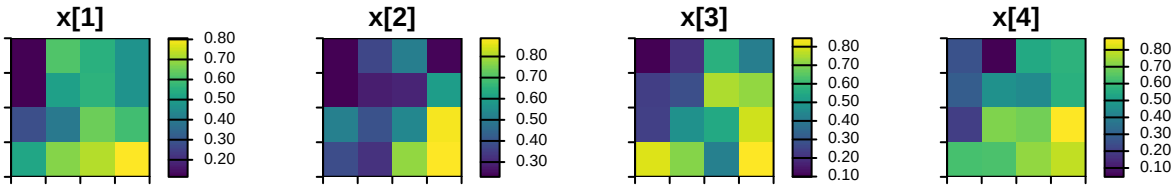
9 Three Risk-Components: p[H], V, Q

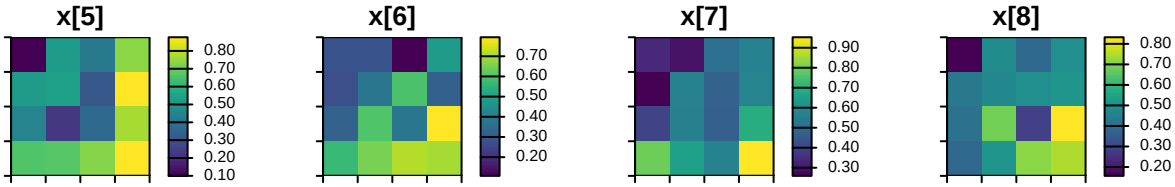
So far we have decomposed risk into two components, $p[H]$ and V. Now we consider distinguishing a third component, namely *exposure*, denoted as RQ. That extension of the PRA-theory can be implemented in all forms of PRA: sampling-, distribution- and model-based. In this chapter, we only give a sampling-based example of three-component PRA.

We create a virtual spatiotemporal dataset for $\{x, z\}$ on a spatial grid of 4×4 and for a sequence of 8 years.

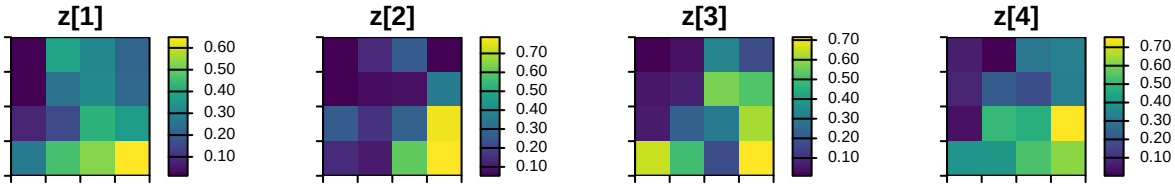
```
set.seed(1)

n_t <- 8 ; x <- z <- array( dim=c(4,4,n_t) )
for(r in 1:4){ for(c in 1:4){ x[r,c,] <- rbeta( n_t, r+c, 10-r-c ) } }
z   <- x^2
```

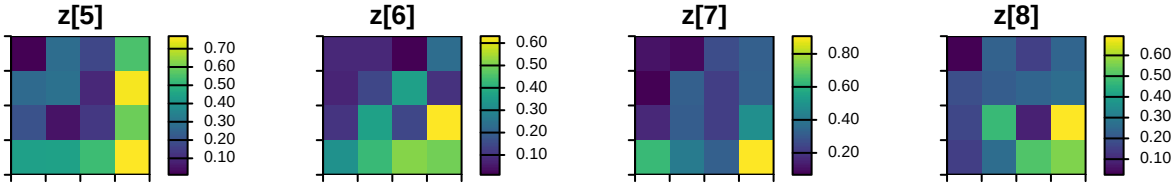
Here are the environmental data x for each of the 8 years:





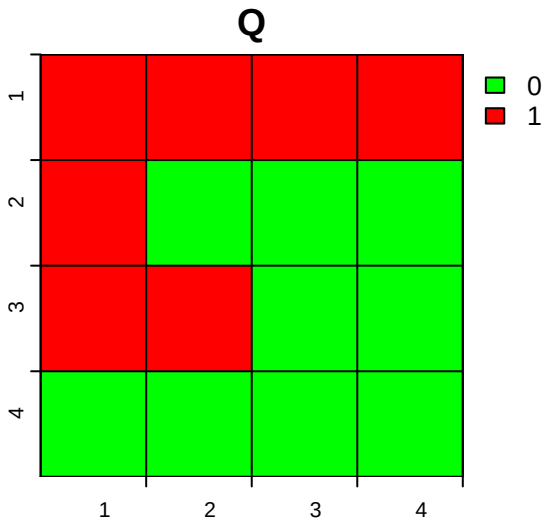
And here are the system response data:





We set a threshold of 0.25 and calculate Q from the fraction of cells where x falls below the threshold at least once. We see that this happens in 7 out of the 16 cells.

```
thr <- 0.25
```



Now we define the algorithm `PRA3` for the three-component PRA:

```
  PRA3 <- function( x=array(dim=c(nlon,nlat,n_t)),
                    z=array(dim=c(nlon,nlat,n_t)), thr.=thr ) {
    ns    <- prod( dim(x)[1:2] ) ; n_t <- dim(x)[3]
    freqH <- function(x,thr.=thr){ sum(x<thr.) }
    n_tH  <- apply(x, c(1,2), freqH)
    siteQ <- which( n_tH > 0, arr.ind=TRUE ) ; nQ <- dim(siteQ)[1]
    Q     <- nQ / ns ; s_Q <- sqrt( Q*(1-Q) / ns )

    xQ    <- sapply( 1:n_t, function(i){x[,,i][siteQ]} )
    zQ    <- sapply( 1:n_t, function(i){z[,,i][siteQ]} )
    PRAQ  <- PRA( c(xQ), c(zQ), thr. )
    pH    <- PRAQ["pH"]   ; V   <- PRAQ["V"]   ; R.Q   <- PRAQ["R"]
    s_pH  <- PRAQ["s_pH"] ; s_V <- PRAQ["s_V"] ; s_R.Q <- PRAQ["s_R"]
    R     <- Q * R.Q
    s_R   <- sqrt( s_Q^2*s_R.Q^2 + s_Q^2*R.Q^2 + Q^2*s_R.Q^2 )

    result           <- c(  Q ,  pH ,  V ,  R ,  s_Q ,  s_pH ,  s_V ,  s_R  )
    names ( result ) <- c( "Q", "pH", "V", "R", "s_Q", "s_pH", "s_V", "s_R" )
    return( result ) }
```

And here are the results for our virtual dataset.

```
  PRAst <- PRA3( x, z, thr ) ; PRAst
>           Q          pH           V           R         s_Q        s_pH
> 0.437500000 0.267857143 0.190310229 0.022301980 0.124019593 0.059177351
>         s_V         s_R
> 0.019730267 0.008500176
```

10 Continuous formulation of PRA

We can increase the number of thresholds in PRA as much as we want (provided it makes sense for the available data of course). In the limit of the number of thresholds going to infinity, we arrive at a *continuous* formulation of PRA, for which the equations were given by Marcel Van Oijen and Brewer (2022) (p.6-7). Here they are:

$$R = \int_{x=-\infty}^{thr} r(x)dx, \tag{4}$$

where

$$\begin{aligned} r(x) &= p[x]\, v(x), \\ v(x) &= E[z|x \geq thr] - E[z|x]. \end{aligned} \tag{5}$$

Note that using these equations requires that we specify $p[x]$ and $E[z|x]$ for all x. That was not necessary or even possible in sampling-based PRA, which goes no further than counting occurrences of x in hazardous intervals. So only distribution- and model-based PRA allow a continuous formulation. In distribution-based PRA, we make the distribution $p[x, z]$ explicit which gives us $p[x]$ and $E[z|x]$ immediately. In model-based PRA, we specify a function $z = f(x) + \epsilon$ such that $E[z|x] = f(x)$, and estimate that function as well as $p[x]$ from data, often by means of Bayesian parameter estimation.

Marcel Van Oijen and Brewer (2022) gave an example of single-threshold, distribution-based continuous PRA in Chapter 9. They further suggested (on p.7) that zero-threshold continuous PRA is feasible if we assume that the function $z(x)$ has a known maximum. We show an example of that approach in the next section, giving a model-based PRA example. The next and final section of this chapter shows that continuous-PRA allows deriving a continuous probability distribution for expected loss, and for that we give a distribution-based PRA example.

10.1 Zero-threshold continuous PRA

Our example will be similar to the second of the two examples shown in the chapter on model-based PRA of Marcel Van Oijen and Brewer (2022) (p.42-43). Here are the equations that capture our knowledge about $\{x, z\}$.

- $z_{max} = 1$
- $x \sim U[0, 3]$
- $E[z|x] = 1 - e^{-x}$

We initially ignore UQ. The model-based continuous zero-threshold PRA-equations then become:

- $p[x] = \frac{1}{3}; \quad x \in [0, 3]$,
- $v(x) = z_{max} - E[z|x] = e^{-x}$,
- $r(x) = p[x]v(x) = \frac{1}{3}\, e^{-x}$.

We show these results in Figure 4.

The total risk is calculated as follows:

- $R = \int_0^3 r(x)dx = \frac{1}{3}(1 - e^{-3}) \approx 0.32$.

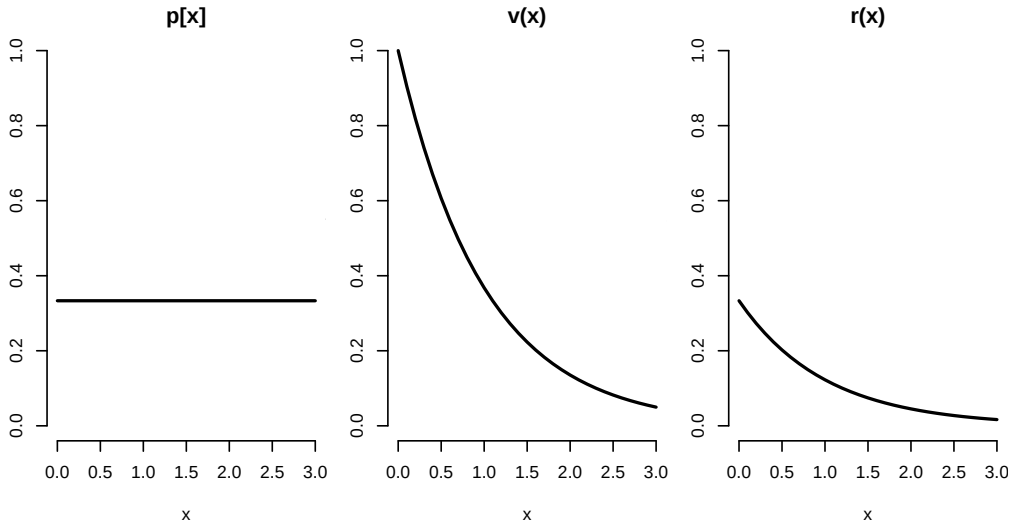


Figure 4: Continuous zero-treshold PRA with known z-maxixum equal to 1.

We see that total risk, i.e. overall expected loss due to suboptimal x for this system, is about 32%. That total risk applies to the whole domain of x. But we can calculate the PRA-terms associated with any interval $x \in [a, b]$ as follows:

- $R_a^b = \int_a^b r(x)dx = \frac{1}{3}(e^{-a} - e^{-b})$.
- $pH_a^b = \int_a^b p(x)dx = \frac{1}{3}(b - a)$.
- $V_a^b = R_a^b \,/\, pH_a^b$

Let's show the results for a sequence of 30 consecutive intervals (Fig. 5):

```
R.seq  <- ( exp(-x.seq[-n_x]) - exp(-x.seq[-1]) ) / 3
pH.seq <- ( x.seq[-1] - x.seq[-n_x] ) / 3
V.seq  <- R.seq / pH.seq
```

10.1.1 UQ

Above we assumed distribution $p[x]$, model $E[z|x]$, and their parameters to be known exactly. In practice, these will all be uncertain. Uncertainties (including those for distribution hyperparameters) can be expressed by means of probability distributions. See other chapters for details and examples.

10.2 Loss distribution

- EXERCISE. Derive the "loss-distribution" $p[loss] = p[\Delta z]$.
- Be careful: z is probabilistic even if there is no hazard present (e.g. $x > thr$). So not all Δz can be ascribed to the hazard variable.
- We do know the **expected** effect of the hazard at all levels of x, and that expected effect is 100% due to the hazard variable. Can we use that knowledge to derive a "loss-due-to-x" distribution? This obviously requires more than one hazardous region.
- Consider continuous PRA and study the loss-distribution with the simple bivariate Gaussian of Marcel Van Oijen and Brewer (2022) Ch. 9. In continuous PRA, **expected loss** for a given value of x below the threshold equates to the vulnerability to such x. So to find the loss distribution in continuous

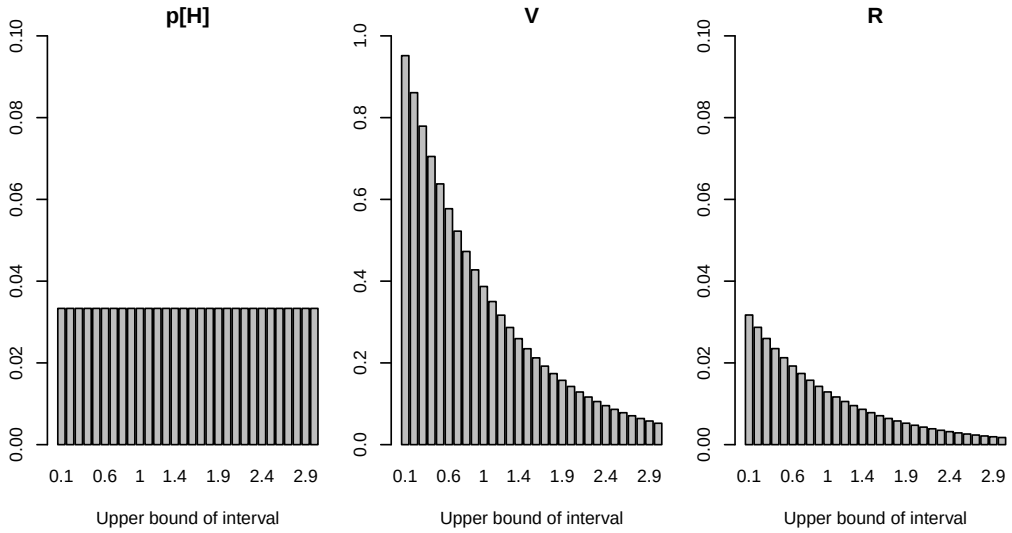


Figure 5: Multi-threshold PRA derived from zero-threshold continuous PRA.

PRA, simply plot p[x] against v(x). Note that this simple way of finding the loss-distribution only works when every value of v(x) is unique, i.e. when v(x) is strictly increasing or decreasing with x. [If there are multiple values of x with the same expected loss (vulnerability) then we would have to sum the p[x] for those different values of x to find the total probability for that expected loss.]

- The function r(x) shows whether we need to worry mainly about extreme hazard values or more moderate ones. The loss distribution is less useful: it does not show what values of x are most cause of concern. Still, the user of the PRA may want to know how variable loss is.

The following code is based on Marcel Van Oijen and Brewer (2022) Chapters 2 and 9, with extensions.

```
m     <- c(0,0) ; S  <- diag(1,2) ; S[1,2] <- S[2,1] <- 0.5
Vx    <- S[1,1] ; Vz <- S[2,2]    ; rxz    <- S[1,2] / sqrt(Vx*Vz)
mx    <- m[1]   ; mz <- m[2]

px    <- function( x, m=mx, V=Vx ){ dnorm( x, m, sqrt(V) ) }
x.seq <- seq( mx-2, mx+2, length.out=41 )
```

```
Ez_x  <- function(x){ mz + (x-mx)*rxz }
v     <- function(x,thr.=0){ EzVz_Gauss( m,S, thr.=0 )["Ez_xhi"] - Ez_x(x) }
r     <- function(x,thr.=0){ px(x) * v(x,thr.) }

thr   <- 1
p.seq <- px(x.seq) ; v.seq <- v(x.seq,thr) ; r.seq <- r(x.seq,thr)
```

See Fig. 6.

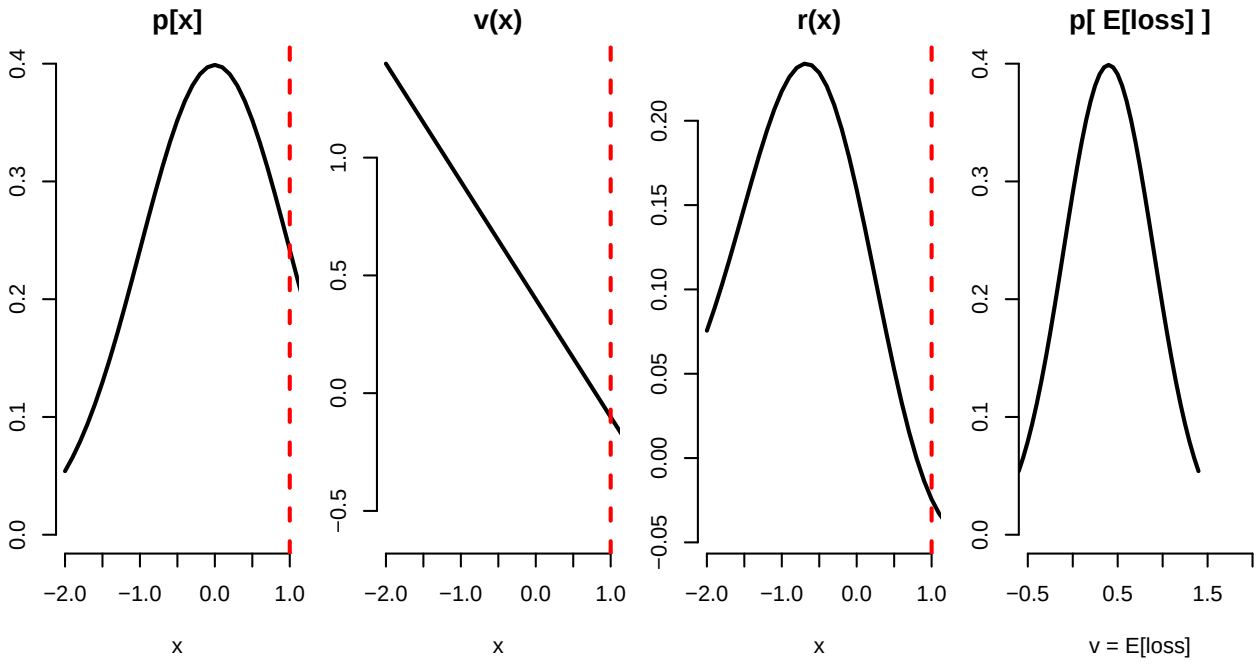


Figure 6: Continuous PRA applied to a bivariate Gaussian dsistribution.

11 Example I: Sparse Data (n = 4)

11.1 Sampling-based PRA

We first analyse the n=4 sparse dataset using sampling-based single-threshold PRA, setting the threshold at thr = 0:

```
PRA_Ss <- PRA( x_4, z_4, thr=0 )
```

11.2 Distribution-based PRA

We next fit a bivariate Gaussian (which can only be justified with so few data points if we know that bivariate Gaussians work well for such {x,z}).

```
set.seed(1)

PRA_Ds  <- PRA_Gauss ( m_4, S_4, n_4, 0 )
PRA_Ds2 <- PRA_Gauss2( m_4, S_4, n_4, 0 )
PRA_Ds3 <- PRA_Gauss3( m_4, S_4, n_4, 0 )
PRA_Ds4 <- PRA_Gauss4( m_4, S_4, n_4, 0 )
PRA_Ds5 <- PRA_Gauss5( m_4, S_4, n_4, 0 )
```

11.3 Summary of various PRAs

```
rbind( PRA_Ss, PRA_Ds, PRA_Ds2, PRA_Ds3, PRA_Ds4, PRA_Ds5 )
>               pH         V         R      s_pH       s_V       s_R
> PRA_Ss  0.5000000 0.1831088 0.0915544 0.2500000 1.2178816 0.6823538
> PRA_Ds  0.4881461 1.9946388 0.9162238 0.1813731 8.3549033 3.6354627
> PRA_Ds2 0.4987436 0.6367450 0.3276257 0.1836520 1.8208700 0.9804932
> PRA_Ds3 0.4952829 0.7699042 0.3917096 0.2171641 0.9082574 0.5219235
> PRA_Ds4 0.5043637 0.7542802 0.3783195 0.2201052 0.9704297 0.5560053
> PRA_Ds5 0.5000000 0.7978846 0.3989423 0.2500000 0.9169760 0.5500480
```

The methods produce similar values for pH, but for V and R as well as for the uncertainties there are large differences. So carrying out PRA on such small datasets carries significant risks!

12 Example II: Linear Dataset

We use the first linear dataset.

```
xz <- l_xz.L[[1]]   ; m  <- colMeans(xz) ; S  <- cov(xz)
x  <- xz[,1]        ; z  <- xz[,2]       ; n  <- length(x)
X  <- cbind( 1, x ) ; Vz <- 1            ; Sz <- diag(Vz,n)
```

12.1 Sampling-based PRA

We choose a threshold value of -1.

```
thr    <- -1
```

```
PRA_Ss <- PRA( x, z, thr )
```

12.2 Distribution-based PRA

```
PRA_Ds <- PRA_Gauss( m, S, n, thr )
```

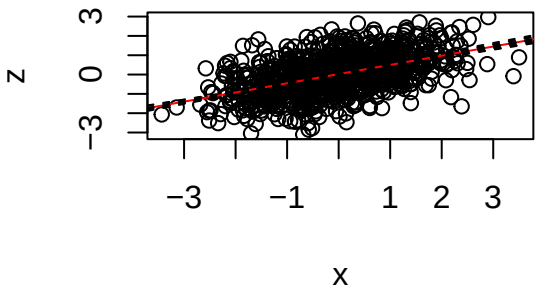
12.3 Model-based PRA with LS72 for the linear data instead of Nimble

Marcel Van Oijen and Brewer (2022) Ch. 7 used Nimble for model-based PRA. Nimble was a good choice because it can be used for any model. Here we just check what Fig. 7.2 would have looked like if we had used the Lindley and Smith (1972) equations for the linear data set instead of MCMC. Note that using the LS72 equations requires the measurement uncertainty for z to be specified as a Gaussian with known variance for each data point. This is usually done by pre-specifying a diagonal matrix for Σ_z with the same value everywhere on the diagonal. The Nimble-approach taken in Ch. 7 of Marcel Van Oijen and Brewer (2022) also assumes the same error-variance for each z-measurement, but it does not pre-specify its magnitude: it is an additional parameter to be estimated and thus to be given a prior like the two other parameters (mean and slope of the linear regression). Here we have set the measurement variance for each data point to 1.

12.3.1 Prior and posterior

```
# Prior:
mb        <- c(0,0) ; Vb <- c(1.e4,1.e4) ; Sb <- diag(Vb)
# Posterior:
Sb_y_LS72 <- solve( solve(Sb) + t(X) %*% solve(Sz) %*% X )
mb_y_LS72 <- Sb_y_LS72 %*% (solve(Sb) %*% mb + t(X) %*% solve(Sz) %*% z)
```

```
par(mfrow=c(1,2))
plot( x, z ) ; abline( mb_y_LS72, col="red" )
nsmpl <- 10  ; smpl_b <- rmvnorm( nsmpl, mean=mb_y_LS72, sigma=Sb_y_LS72 )
for( i in 1:nsmpl) { abline( smpl_b[i,], lty=2 ) }
```

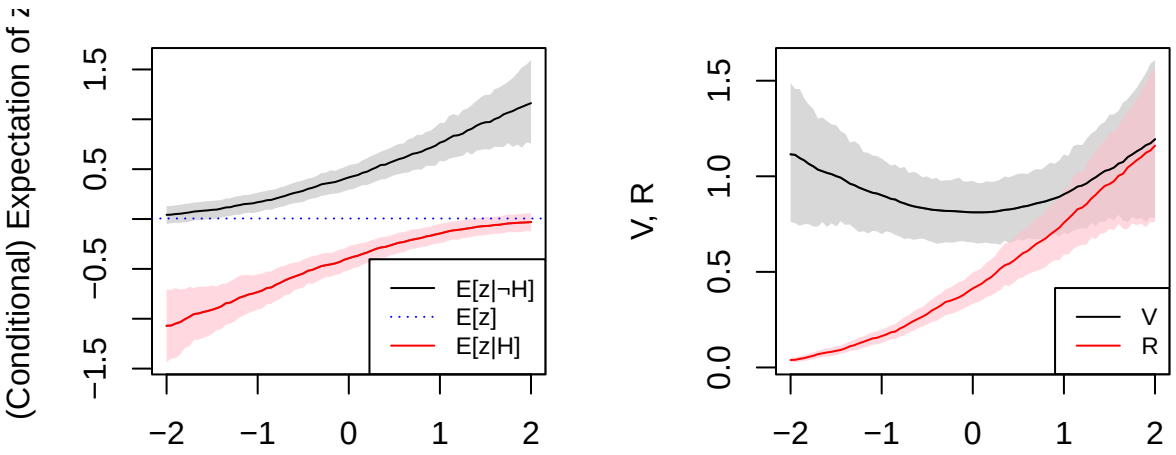


12.3.2 Model-based PRA using the posterior parameter distribution

The following code is a major simplification of the code underlying Fig. 7.2 of Marcel Van Oijen and Brewer (2022). It makes use, under the comment-line 'Method 1', of the law of total variance $Var[z] = E[Var[z|x]] + Var[E[z|x]]$ which we apply twice: to z_H and to z_{NotH}. 'Method 2' is more generally applicable and is more reliable when the data are sparse at the cost of requiring more lines of code, and having more computational demand.

```
n_unc    <- 1e3
theta    <- rmvnorm( n_unc, mean=mb_y_LS72, sigma=Sb_y_LS72 )
lm.alpha <- theta[,1]             ; lm.beta <- theta[,2]
lm.sigma <- rep( sqrt(Vz), n_unc )

thr      <- seq( -2, 2, by=0.05 ) ; n_thr <- length(thr)

EzHj     <- EzNotHj  <-   Rj  <-   Vj           <- numeric(n_unc)
EzH      <- EzNotH   <-   R   <-   V   <-   pH  <- numeric(n_thr)
                        s_R   <- s_V   <- s_pH  <- numeric(n_thr)
                          R2  <-   V2  <-   pH2 <- numeric(n_thr) # remove later?
                        s_R2  <- s_V2  <- s_pH2 <- numeric(n_thr) # remove later?
LCIR     <- UCIR     <- LCIV  <- UCIV           <- numeric(n_thr)
LCIzNotH <- UCIzNotH <- LCIzH <- UCIzH          <- numeric(n_thr)
qu       <- function( z, q=0.025 ){ quantile( z, q, na.rm=T ) }

for(i in 1:n_thr) {
  i_H    <- which( x <  thr[i] ) ; n_H     <- length(i_H)
  i_NotH <- which( x >= thr[i] ) ; n_NotH  <- length(i_NotH)
  pH[i]  <- n_H / n              ; s_pH[i] <- sqrt( pH[i]*(1-pH[i]) / n )

  # Method 1: V, R from model expectations and UQ from law of total variance
  Ex_H    <- mean( x[i_H] )          ; Ex_NotH <- mean( x[i_NotH] )
  Ez_H    <- c(1,Ex_H) %*% mb_y_LS72 ; Ez_NotH <- c(1,Ex_NotH) %*% mb_y_LS72
  V[i]    <- Ez_NotH-Ez_H
  R[i]    <- pH[i] * V[i]
  Vzi     <- function(i){ t(c(1,x[i])) %*% Sb_y_LS72 %*% c(1,x[i]) + Vz }
  Vz_H    <- sum( sapply(i_H   ,Vzi) ) / n_H    + mb_y_LS72[2]^2 * var(x[i_H])
  Vz_NotH <- sum( sapply(i_NotH,Vzi) ) / n_NotH + mb_y_LS72[2]^2 * var(x[i_NotH])
  s_V[i]  <- sqrt( Vz_H / n_H + Vz_NotH / n_NotH )
  s_R[i]  <- sqrt( s_pH[i]^2 * s_V[i]^2 + s_pH[i]^2 * V[i]^2 + pH[i]^2 * s_V[i]^2 )

  # Method 2
  for(j in 1:n_unc) {
    zj      <- lm.alpha[j] + lm.beta[j] * x + rnorm( n, 0, lm.sigma[j] )
```

```
    EzHj[j] <- mean( zj[i_H] ) ; EzNotHj[j] <- mean( zj[i_NotH] )
    Vj[j]   <- EzNotHj[j] - EzHj[j]
    Rj[j]   <- pH[i] * Vj[j]
  }
  R2[i]   <- mean( Rj ) ; V2[i]   <- mean( Vj )
  s_R2[i] <- sd  ( Rj ) ; s_V2[i] <- sd  ( Vj )

  EzH[i]      <- mean( EzHj )  ; EzNotH[i]   <- mean( EzNotHj )
  LCIzNotH[i] <- qu( EzNotHj ) ; UCIzNotH[i] <- qu( EzNotHj, 0.975 )
  LCIzH[i]    <- qu( EzHj    ) ; UCIzH[i]    <- qu( EzHj   , 0.975 )
  LCIV[i]     <- qu( Vj      ) ; UCIV[i]     <- qu( Vj     , 0.975 )
  LCIR[i]     <- qu( Rj      ) ; UCIR[i]     <- qu( Rj     , 0.975 )
}
```



These plots look similar to Fig. 7.2 of Marcel Van Oijen and Brewer (2022) (provided we started from a large dataset), despite the use of LS72 conjugate Bayesian updating rather than MCMC, and major simplification of the code.

Above, we ran model-based single-threshold PRA for a series of different threshold values. Here is the PRA for the common threshold choice for this chapter's linear dataset: thr = -1.

```
i       <- which(thr == -1)
PRA_Ms  <- c( pH[i], V[i] , R[i] , s_pH[i], s_V[i] , s_R[i]  )
PRA_Ms2 <- c( pH[i], V2[i], R2[i], s_pH[i], s_V2[i], s_R2[i] )
```

12.3.3 Simplified code: PRA_LS72

We now simplify the code even more, ending up with a simple R-function for LS72-model-based PRA. V and R are calculated from model expectations and UQ from the law of total variance (mentioned above).

```
PRA_LS72 <- function( x, z, thr=0, Vz=1 ) {
  n         <- length(x) ; X  <- cbind(1,x)
  mb        <- c(0,0)    ; Vb <- c(1.e4,1.e4)
  Sb        <- diag(Vb)  ; Sz <- diag(Vz,n)
  Sb_y_LS72 <- solve( solve(Sb) + t(X) %*% solve(Sz) %*% X )
  mb_y_LS72 <- Sb_y_LS72 %*% (solve(Sb) %*% mb + t(X) %*% solve(Sz) %*% z)

  i_H       <- which( x <  thr )       ; n_H     <- length(i_H)
```

```
  i_NotH    <- which( x >= thr )       ; n_NotH  <- length(i_NotH)
  pH        <- n_H / n                 ; s_pH    <- sqrt( pH*(1-pH) / n )
  Ex_H      <- mean( x[i_H] )          ; Ex_NotH <- mean( x[i_NotH] )
  Ez_H      <- c(1,Ex_H) %*% mb_y_LS72 ; Ez_NotH <- c(1,Ex_NotH) %*% mb_y_LS72
  V         <- Ez_NotH - Ez_H          ; R       <- pH * V
    Vzi     <- function(i){ t(c(1,x[i])) %*% Sb_y_LS72 %*% c(1,x[i]) + Vz }
  Vz_H      <- sum( sapply(i_H   ,Vzi) ) / n_H    + mb_y_LS72[2]^2 * var(x[i_H])
  Vz_NotH   <- sum( sapply(i_NotH,Vzi) ) / n_NotH + mb_y_LS72[2]^2 * var(x[i_NotH])
  s_V       <- sqrt( Vz_H / n_H + Vz_NotH / n_NotH )
  s_R       <- sqrt( s_pH^2 * s_V^2 + s_pH^2 * V^2 + pH^2 * s_V^2 )

  return( list( mb  = mb_y_LS72, Sb = Sb_y_LS72,
                PRA = c( pH=pH, V=V, R=R, s_pH=s_pH, s_V=s_V, s_R=s_R ) ) )
}
```

```
PRA_Ms_LS72 <- PRA_LS72( x, z, thr=-1, Vz=1 )
```

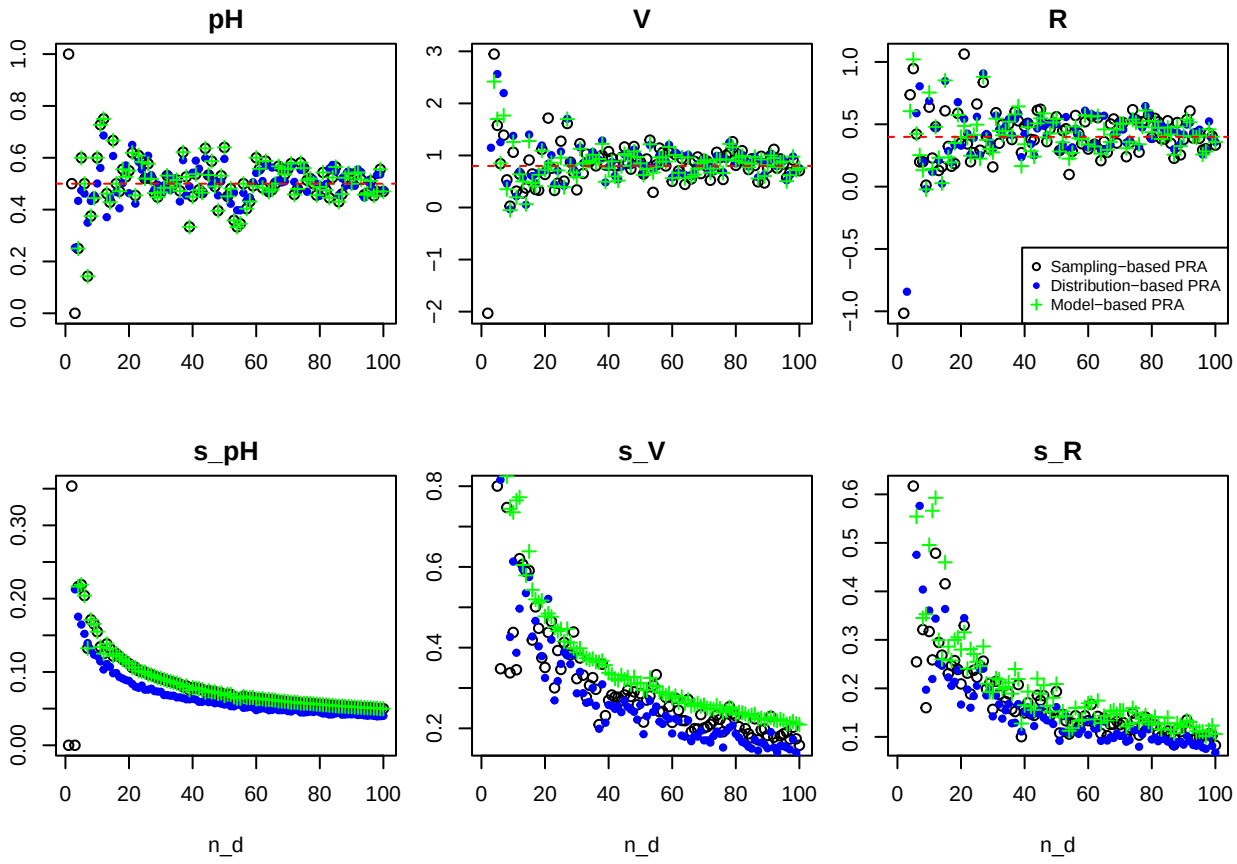
12.4 Summary of various PRAs

```
rbind( PRA_Ss, PRA_Ds, PRA_Ms_LS72$PRA, PRA_Ms, PRA_Ms2 )
>               pH         V         R       s_pH        s_V        s_R
> PRA_Ss  0.1820000 0.8719316 0.1586916 0.01220148 0.07483537 0.01730677
> PRA_Ds  0.1827669 0.8967760 0.1639762 0.00985890 0.05329017 0.01407474
>         0.1820000 0.9008213 0.1639495 0.01220148 0.08469216 0.01895965
> PRA_Ms  0.1820000 0.9008213 0.1639495 0.01220148 0.08469216 0.01895965
> PRA_Ms2 0.1820000 0.9011930 0.1640171 0.01220148 0.09947910 0.01810520
```

13 Example III: A Sequence of Linear Datasets (n linearly increasing)

```
thr <- 0
```

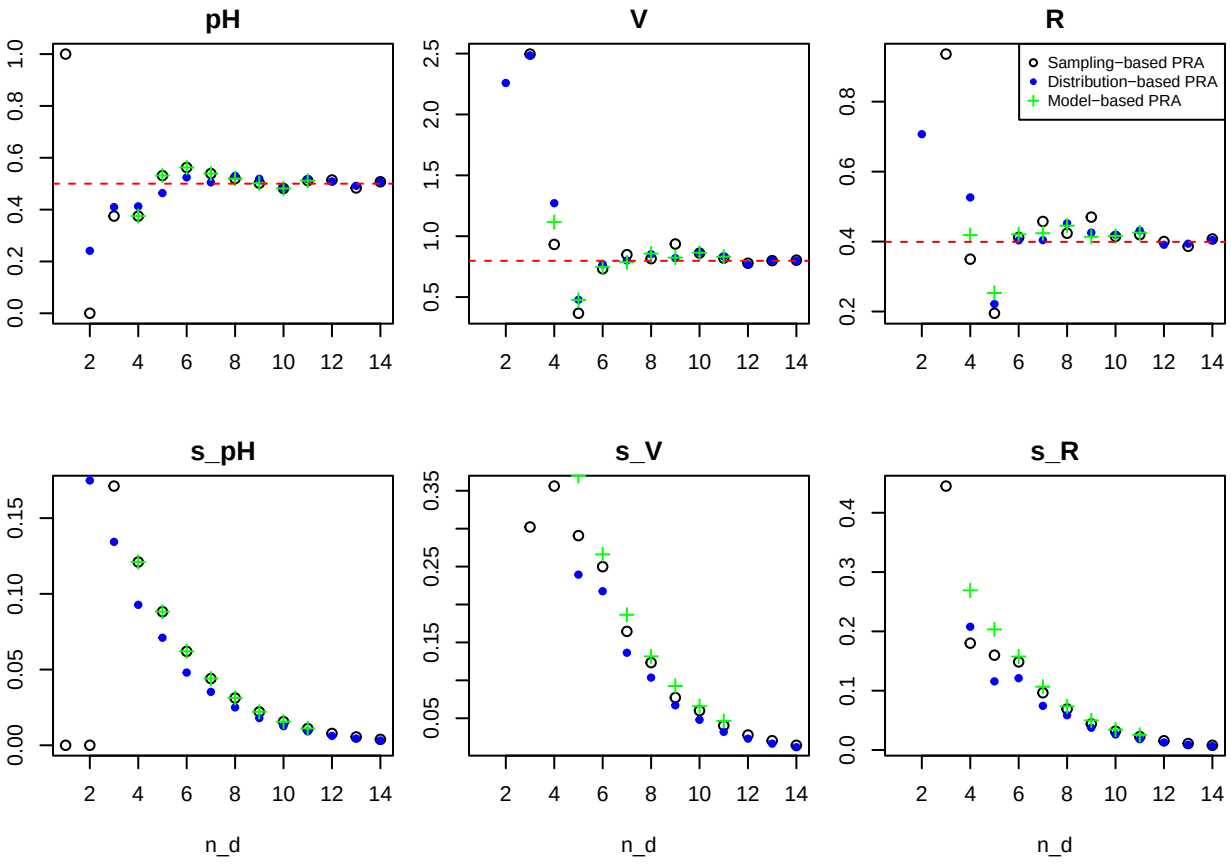
```
l_xz <- l_xz.L2 ; n_d <- length(l_xz)

PRA.tbl <- t( sapply( 1:n_d, function(d){
  PRA ( l_xz[[d]][,1], l_xz[[d]][,2], thr=thr) } ) )

i2 <- 3
PRA.tbl2 <- t( sapply( i2:n_d, function(d){
  PRA_Gauss( m.=colMeans(l_xz[[d]]), cov(l_xz[[d]]), d, thr) } ) )

i3 <- 4
PRA.tbl3 <- t( sapply( i3:n_d, function(d){
  PRA_LS72( l_xz[[d]][,1], l_xz[[d]][,2], thr=0, Vz=1 )$PRA } ) )
```



14 Example IV: A Sequence of Linear Datasets (n exponentially increasing)

```
thr <- 0
```

```
l_xz <- l_xz.L3 ; n_d <- length(l_xz)

PRA.tbl <- t( sapply( 1:n_d, function(d){
  PRA ( l_xz[[d]][,1], l_xz[[d]][,2], thr=thr) } ) )

i2 <- 2
PRA.tbl2 <- t( sapply( i2:n_d, function(d){
  PRA_Gauss( m.=colMeans(l_xz[[d]]), cov(l_xz[[d]]), 2^d, thr) } ) )

i3 <- 4 ; i3max <- 11
PRA.tbl3 <- t( sapply( i3:i3max, function(d){
  PRA_LS72( l_xz[[d]][,1], l_xz[[d]][,2], thr=0, Vz=1 )$PRA } ) )
```



15 Example V: Nonlinear Dataset

We use the first nonlinear dataset.

```
xz <- l_xz.NL[[1]] ; m <- colMeans(xz) ; S <- cov(xz)
x  <- xz[,1]       ; z <- xz[,2]       ; n <- length(x)
```

15.1 Sampling-based PRA

We choose a threshold value of 1.

```
thr <- 1

PRA_Ss <- PRA( x, z, thr )
```

15.2 Distribution-based PRA

Using the (here clearly wrong) assumption of a bivariate Gaussian for {x,z}.

```
PRA_Ds <- PRA_Gauss( m, S, n, thr )
```

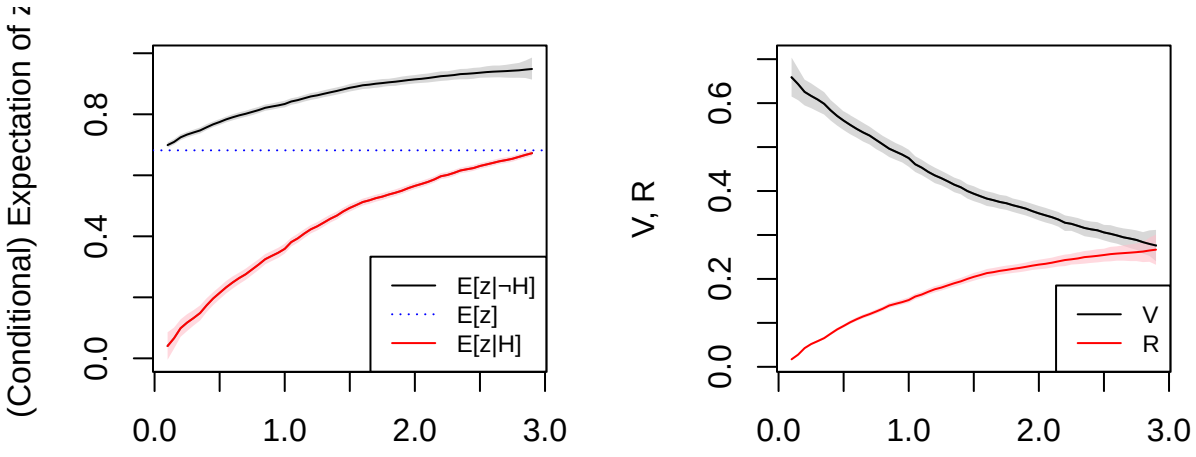
15.3 Model-based PRA using Nimble

Bayesian calibration, using Nimble, of a nonlinear exponential model for the data:

```
Model1.Code <- nimbleCode({
  lm.alpha  ~ dnorm( 0, sd=100 )
  lm.beta   ~ dnorm( 0, sd=100 )
  lm.tau    ~ dgamma( 0.01, 0.01 )
  lm.sigma <- 1 / sqrt(lm.tau)
  for(i in 1:ndata){
    lm.mu[i] <- lm.alpha + lm.beta*exp(-x[i])
    z[i]      ~ dnorm( lm.mu[i], sd=lm.sigma )
  }
} )

Model1.Constants <- list( ndata=n, x=x )
Model1.Data      <- list(z=z)
Model1.Nimble    <- nimbleModel  ( Model1.Code, constants=Model1.Constants,
                                                data=Model1.Data )
Model1.Comp      <- compileNimble( Model1.Nimble )
Model1.Conf      <- configureMCMC( Model1.Nimble, print=F )
Model1.Conf$addMonitors( c("lm.sigma"), print=F )
Model1.MCMC      <- buildMCMC    ( Model1.Conf )
Model1.MCMC.Comp <- compileNimble( Model1.MCMC )
ntheta           <- 1e4 ; nburnin <- 1e3 ; niter <- ntheta + nburnin

set.seed(1)

theta <- runMCMC( Model1.MCMC.Comp, nburnin=nburnin, niter=niter, prog=F )
```

Generating $n_unc = 1000$ PRA-results for each of $n_thr = 100$ different thresholds. The multiple PRAs for each threshold use different values taken from the posterior distribution for the model's parameters.

```
n_unc    <- 1e3
itheta   <- sample( 1:ntheta, n_unc, replace=(ntheta<n_unc) )

thr      <- seq( 0.1, 2.9, by=0.05 )  ;   n_thr <- length(thr)

EzHj     <- EzNotHj  <-   Rj  <-   Vj          <- numeric(n_unc)
EzH      <- EzNotH   <-   R   <-   V   <-   pH <- numeric(n_thr)
                       s_R   <- s_V   <- s_pH <- numeric(n_thr)
LCIR     <- UCIR     <- LCIV  <- UCIV          <- numeric(n_thr)
LCIzNotH <- UCIzNotH <- LCIzH <- UCIzH         <- numeric(n_thr)

lm.alpha <- theta[,1] ; lm.beta <- theta[,2] ; lm.sigma <- theta[,3]
for(i in 1:n_thr) {
  i_H   <- which( x < thr[i] ) ; n_H     <- length(i_H)
  pH[i] <- n_H / n             ; s_pH[i] <- sqrt( pH[i]*(1-pH[i]) / n )
  for(j in 1:n_unc) {
    zj      <- lm.alpha[itheta[j]] + lm.beta[itheta[j]] * exp(-x) +
               rnorm( n, 0, lm.sigma[itheta[j]] )
    EzHj[j] <- mean( zj[i_H] ) ; EzNotHj[j] <- mean( zj[-i_H] )
    Vj[j]   <- EzNotHj[j] - EzHj[j]
    Rj[j]   <- pH[i] * Vj[j] }
  R[i]   <- mean( Rj ) ; V[i]   <- mean( Vj )
  s_R[i] <- sd  ( Rj ) ; s_V[i] <- sd  ( Vj )

  EzH[i]      <- mean( EzHj )  ; EzNotH[i]   <- mean( EzNotHj )
  LCIzNotH[i] <- qu( EzNotHj ) ; UCIzNotH[i] <- qu( EzNotHj, 0.975 )
  LCIzH[i]    <- qu( EzHj    ) ; UCIzH[i]    <- qu( EzHj   , 0.975 )
  LCIV[i]     <- qu( Vj      ) ; UCIV[i]     <- qu( Vj     , 0.975 )
  LCIR[i]     <- qu( Rj      ) ; UCIR[i]     <- qu( Rj     , 0.975 )
}
```




```
i      <- which(thr == 1)
PRA_Ms <- c( pH[i], V[i], R[i], s_pH[i], s_V[i], s_R[i] )
```

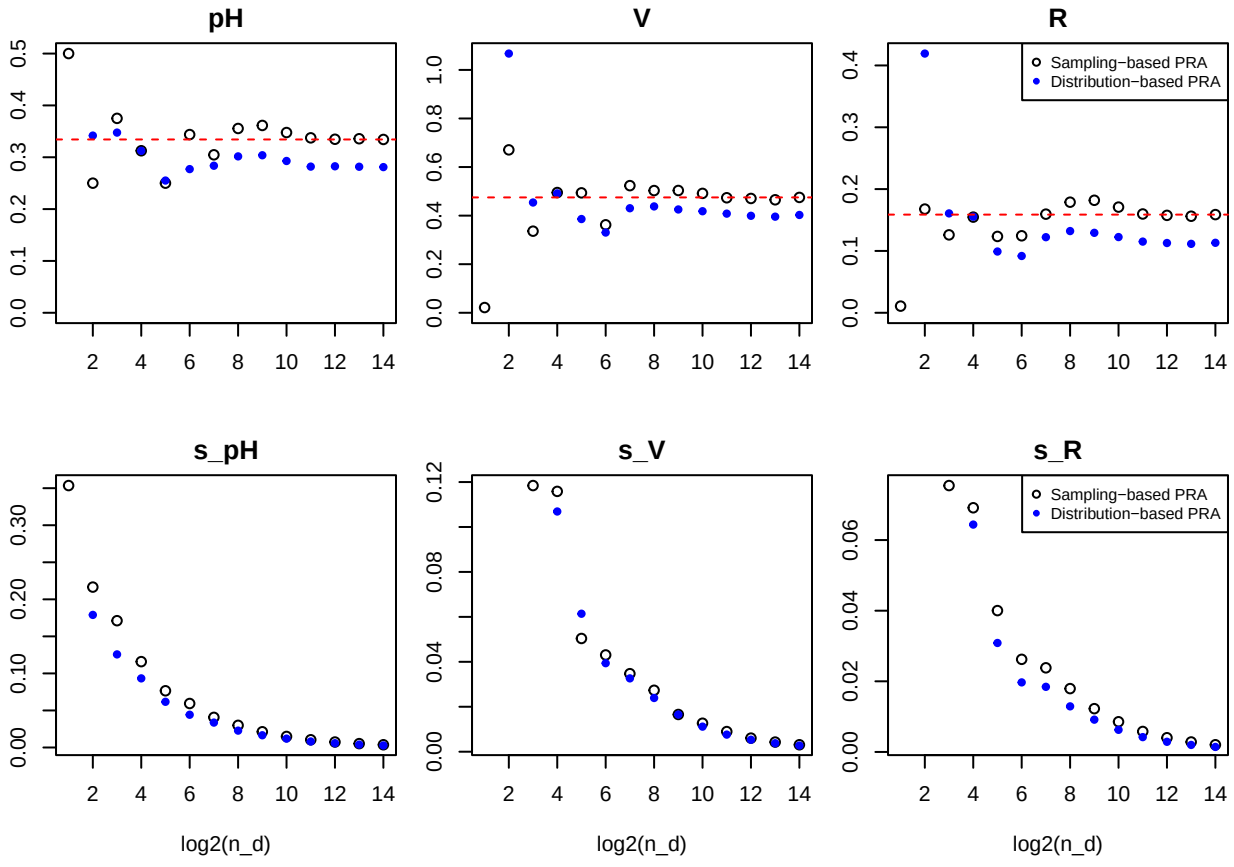
15.4 Summary of various PRAs

```
> rbind( PRA_Ss, PRA_Ds, PRA_Ms )
>              pH         V         R       s_pH         s_V         s_R
> PRA_Ss 0.3200000 0.4732057 0.1514258 0.01475127 0.012989111 0.008126446
> PRA_Ds 0.2819098 0.3958725 0.1116121 0.01157636 0.011043550 0.005780242
> PRA_Ms 0.3200000 0.4748138 0.1519404 0.01475127 0.009593264 0.003069845
```

16 Example VI: A Sequence of Nonlinear Datasets (n exponentially increasing)

```
thr <- 1
```

```
l_xz <- l_xz.NL3 ; n_d <- length(l_xz)

PRA.tbl <- t( sapply( 1:n_d, function(d){
  PRA( l_xz[[d]][,1], l_xz[[d]][,2], thr=thr ) } ) )

i2 <- 2
PRA.tbl2 <- t( sapply( i2:n_d, function(d){
  PRA_Gauss( m.=colMeans(l_xz[[d]]), cov(l_xz[[d]]), 2^d, thr ) } ) )
```



17 Example VII: German Forestry Data

17.1 Sampling-based PRA

In the following two examples of sampling-based PRA (single-threshold, multi-threshold), our system variable initially will be survival (% y-1) which we calculate as *100 - mortality*. But all outcomes of the PRA would remain the same if we had chose simply *-mortality* because the PRA looks at differences rather than absolute values of the system variable, so any constant (like the 100 in **100-mortality**) drops out.

We chose **survival** as our system variable, rather than mortality, because we see PRA as the analysis of expected loss of system performance. But mathematically it of course makes little difference if we choose mortality instead, it would only make V and R negative quantities. We will show that for the multi-threshold example.

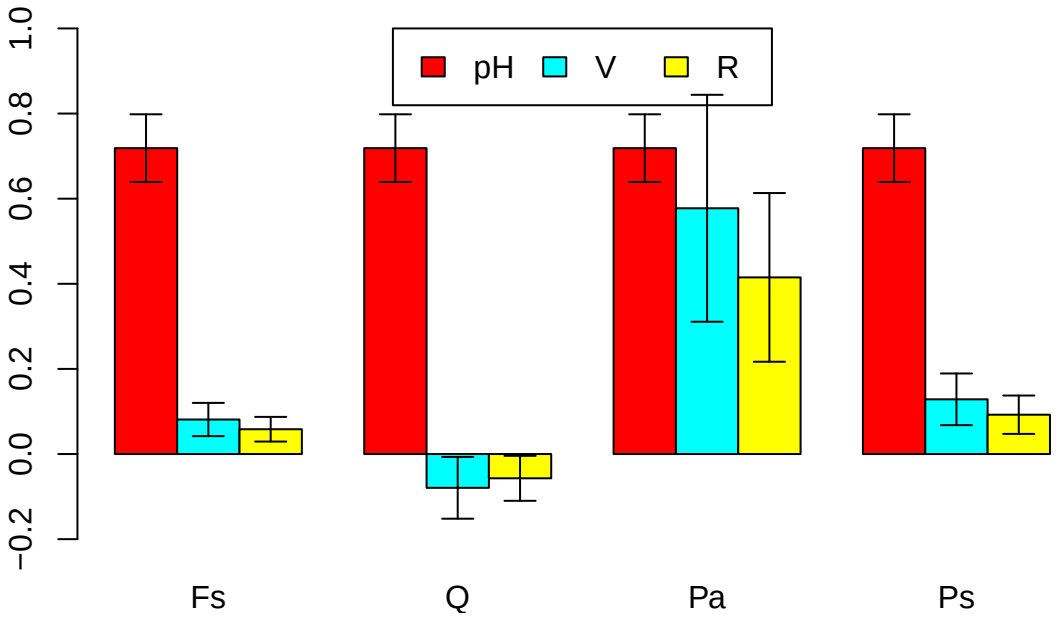
17.1.1 Single-threshold PRA

Sampling-based single-threshold PRA for survival = 100 - mortality (%):

```
thr <- 250
```

```
PRA_Ss.Fs <- PRA( x_r3, 100-z_Fs, thr )
PRA_Ss.Q  <- PRA( x_r3, 100-z_Q , thr )
PRA_Ss.Pa <- PRA( x_r3, 100-z_Pa, thr )
PRA_Ss.Ps <- PRA( x_r3, 100-z_Ps, thr )
```

```
PRA._ <- as.matrix( cbind(PRA_Ss.Fs, PRA_Ss.Q, PRA_Ss.Pa, PRA_Ss.Ps) )
colnames(PRA._) <- c( "Fs", "Q", "Pa", "Ps" )
m     <- PRA._[1:3,] ; s <- PRA._[4:6,]

par(mfrow=c(1,1))
col   <- c( "red", "cyan", "yellow" )
bp    <- barplot( m, beside=T, ylim=c(-0.2,1), col=col,
                  legend.text=row.names(m),
                  args.legend=list(x="top",hor=T) )
segments( bp, m - s, bp, m + s )
ew    <- (bp[2,1]-bp[1,1])/4
segments( bp - ew, m - s, bp + ew, m - s )
segments( bp - ew, m + s, bp + ew, m + s )
```



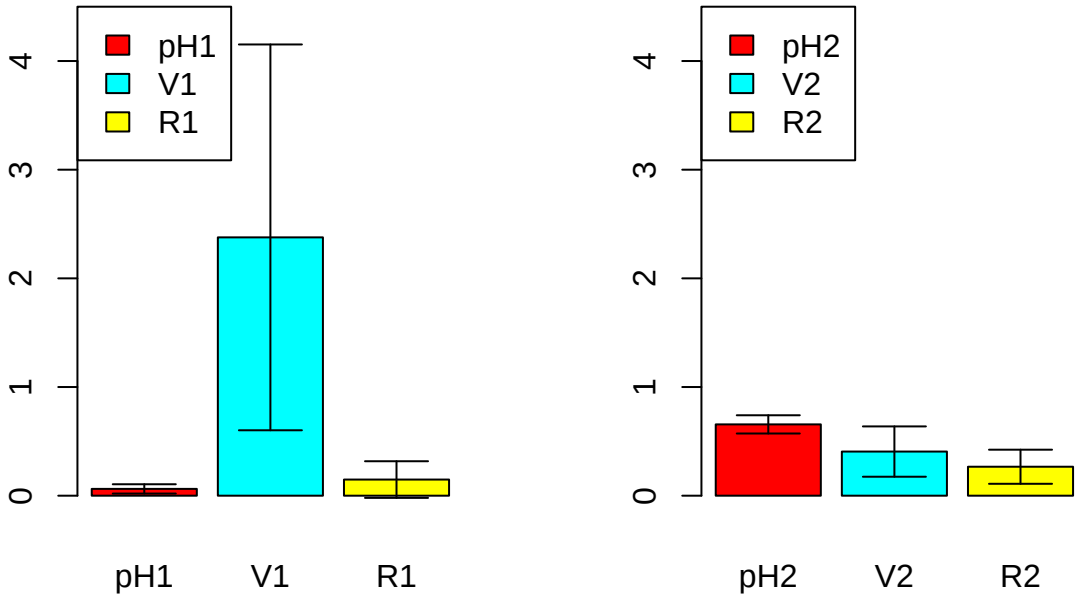
17.1.2 Multi-threshold PRA

```
thr.seq <- c(220,250) ; n_thr <- length(thr.seq)
```

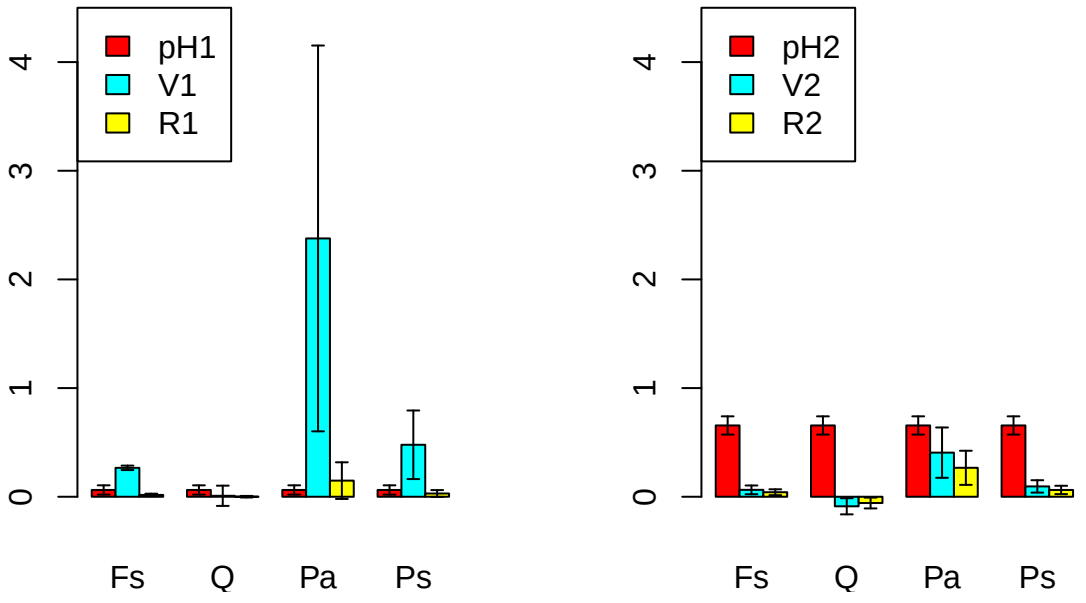
```
# PRAm for survival (= 100 - mortality)
PRA_Sm.Fs  <- PRAm(x_r3, 100-z_Fs, thr.seq)$seq
PRA_Sm.Q   <- PRAm(x_r3, 100-z_Q , thr.seq)$seq
PRA_Sm.Pa  <- PRAm(x_r3, 100-z_Pa, thr.seq)$seq
PRA_Sm.Ps  <- PRAm(x_r3, 100-z_Ps, thr.seq)$seq
PRA_Sm.tbl <- rbind( PRA_Sm.Fs, PRA_Sm.Q, PRA_Sm.Pa, PRA_Sm.Ps )

pHm.tbl    <- matrix(PRA_Sm.tbl[,"pH"  ], nrow=n_thr)
Vm.tbl     <- matrix(PRA_Sm.tbl[,"V"   ], nrow=n_thr)
Rm.tbl     <- matrix(PRA_Sm.tbl[,"R"   ], nrow=n_thr)
s_pHm.tbl  <- matrix(PRA_Sm.tbl[,"s_pH"], nrow=n_thr)
s_Vm.tbl   <- matrix(PRA_Sm.tbl[,"s_V" ], nrow=n_thr)
s_Rm.tbl   <- matrix(PRA_Sm.tbl[,"s_R" ], nrow=n_thr)

m             <- rbind(   pHm.tbl,   Vm.tbl,   Rm.tbl )
s             <- rbind( s_pHm.tbl, s_Vm.tbl, s_Rm.tbl )
colnames( m ) <- c( "Fs" , "Q"  , "Pa", "Ps" )
rownames( m ) <- c( "pH1", "pH2", "V1", "V2", "R1", "R2" )
colnames( s ) <-                colnames( m )
rownames( s ) <- paste0( "sd_", rownames( m ) )
```

```
par(mfrow=c(1,2))
col    <- c( "red", "cyan", "yellow" )

list.i <- list( i1=c(1,3,5), i2=c(2,4,6) )
sp     <- "Pa"
for( j in 1:2 ) {
```

```
  i  <- list.i[[j]]
  bp <- barplot( m[i,sp], beside=T, ylim=c(-0.2,4.5), col=col,
                 legend.text=row.names(m)[i],
                 args.legend=list(x="topleft",hor=F) )
  segments( bp, (m-s)[i,sp], bp, (m+s)[i,sp] )
  ew <- (bp[2,1]-bp[1,1]) / 4
  segments( bp - ew, (m-s)[i,sp], bp + ew, (m-s)[i,sp] )
  segments( bp - ew, (m+s)[i,sp], bp + ew, (m+s)[i,sp] ) }
```



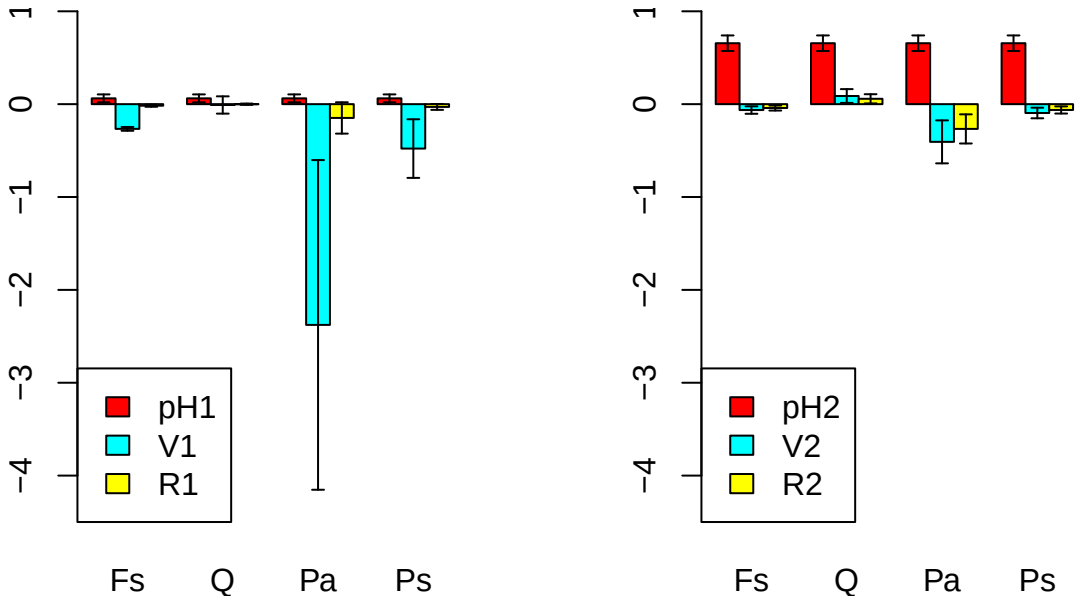

```
par(mfrow=c(1,2))
col <- c( "red", "cyan", "yellow" )

list.i <- list( i1=c(1,3,5), i2=c(2,4,6) )
for( j in 1:2 ) {
  i   <- list.i[[j]]
  bp  <- barplot( m[i,], beside=T, ylim=c(-0.2,4.5), col=col,
                  legend.text=row.names(m)[i],
                  args.legend=list(x="topleft",hor=F) )
  segments( bp, (m-s)[i,], bp, (m+s)[i,] )
  ew  <- (bp[2,1]-bp[1,1]) / 4
  segments( bp - ew, (m-s)[i,], bp + ew, (m-s)[i,] )
  segments( bp - ew, (m+s)[i,], bp + ew, (m+s)[i,] ) }
```



17.1.3 Multi-threshold PRA for mortality rather than survival

Now we repeat the previous multi-threshold PRAm but let mortality be our system variable rather than survival. How will that affect the analysis?

```
# PRAm for mortality
PRA_Sm.Fs  <- PRAm(x_r3, z_Fs, thr.seq)$seq
PRA_Sm.Q   <- PRAm(x_r3, z_Q , thr.seq)$seq
PRA_Sm.Pa  <- PRAm(x_r3, z_Pa, thr.seq)$seq
PRA_Sm.Ps  <- PRAm(x_r3, z_Ps, thr.seq)$seq
PRA_Sm.tbl <- rbind( PRA_Sm.Fs, PRA_Sm.Q, PRA_Sm.Pa, PRA_Sm.Ps )
PRA_Sm.tbl <- rbind( PRA_Sm.Fs, PRA_Sm.Q, PRA_Sm.Pa, PRA_Sm.Ps )
```



We see that all values remain the same apart from their sign in the case of V and R. So $p[H]$ values remain positive whereas most values for V and R become negative.

17.2 Distribution-based PRA

We are going to try a distribution-based PRA of the German forestry data. We choose a simple bivariate Gaussian to represent p[x,z]. This is of course a bad choice for the distribution, given the non-Gaussian mortality rates, so this is no more than an academic exercise or quick first-try.

```
xz <- cbind( x_r3, 100-z_Pa ) ; m <- colMeans(xz) ; S <- cov(xz)
n  <- length(x_r3)

PRA_Ds.Pa <- PRA_Gauss( m, S, n, 250 )
```

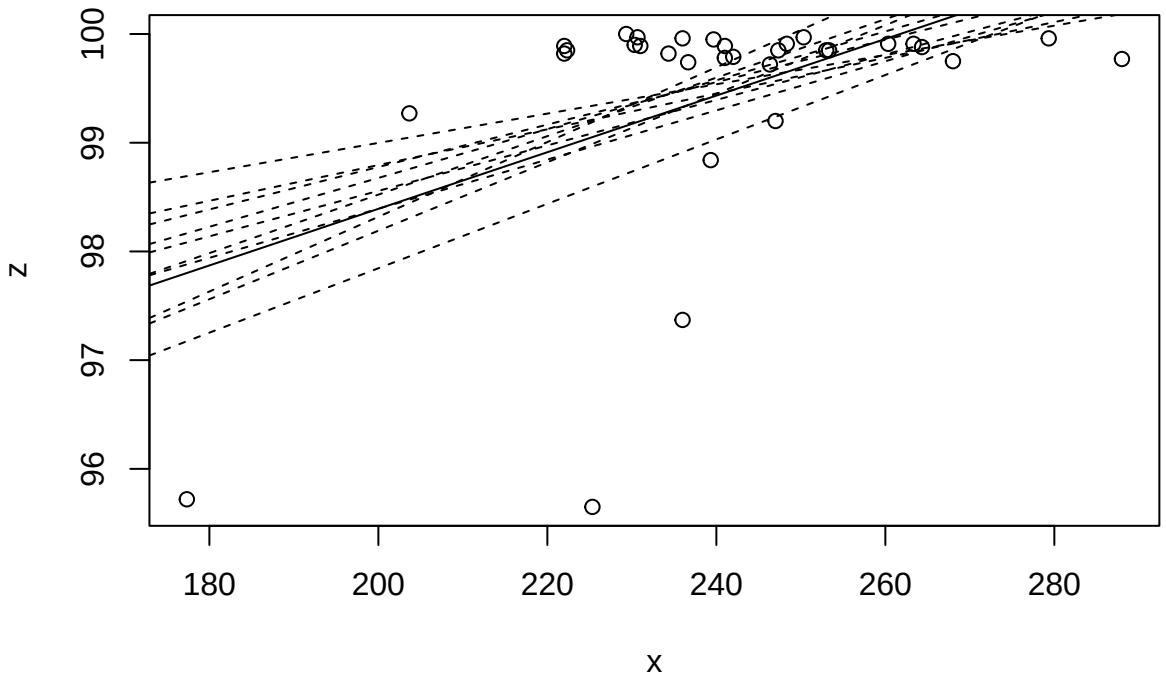
17.3 Model-based conjugate PRA

17.3.1 Survival

We use the equations derived by Lindley and Smith (1972).

```
set.seed(1)

x  <- x_r3    ; z  <- 100-z_Pa     ; n  <- length(x)
Vz <- 1       ; Sz <- diag(Vz,n)   ; X  <- cbind(1,x)

mb <- c(99,0) ; Vb <- c(1.e4,1.e4) ; Sb <- diag(Vb)

# Lindley & Smith (1972)
Sb_y_LS72 <- solve( solve(Sb) + t(X) %*% solve(Sz) %*% X )
mb_y_LS72 <- Sb_y_LS72 %*% (solve(Sb) %*% mb + t(X) %*% solve(Sz) %*% z)
```

```
plot( x, z ) ; abline( mb_y_LS72 )

nsmpl  <- 10
smpl_b <- rmvnorm( nsmpl, mean=mb_y_LS72, sigma=Sb_y_LS72 )
for( i in 1:nsmpl) {
  abline( smpl_b[i,], lty=2 )
}
```




```r
set.seed(1)

thr <- 250

i_H <- which( x<thr ) ; i_NotH <- which( x >= thr )
n_H <- length( i_H )  ; n_NotH <- length( i_NotH )
pH  <- n_H / n        ; s_pH   <- sqrt( pH*(1-pH) / n )

# Method 1: V, R from model expectations and UQ from law of total variance
Ex_H    <- mean( x[i_H ] )          ; Ex_NotH <- mean( x[i_NotH] )
Ez_H    <- c(1,Ex_H) %*% mb_y_LS72 ; Ez_NotH <- c(1,Ex_NotH) %*% mb_y_LS72
V       <- Ez_NotH-Ez_H
R       <- pH * V
Vzi     <- function(i){ t(c(1,x[i])) %*% Sb_y_LS72 %*% c(1,x[i]) + Vz }
Vz_H    <- sum( sapply(i_H   ,Vzi) ) / n_H     + mb_y_LS72[2]^2 * var(x[i_H])
Vz_NotH <- sum( sapply(i_NotH,Vzi) ) / n_NotH + mb_y_LS72[2]^2 * var(x[i_NotH])
s_V     <- sqrt( Vz_H / n_H + Vz_NotH / n_NotH )
s_R     <- sqrt( s_pH^2 * s_V^2 + s_pH^2 * V^2 + pH^2 * s_V^2 )

# Method 2: V, R and UQ from sampling
n_unc    <- 1e3
theta    <- rmvnorm( n_unc, mean=mb_y_LS72, sigma=Sb_y_LS72 )
lm.alpha <- theta[,1] ; lm.beta <- theta[,2]
lm.sigma <- rep( sqrt(Vz), n_unc )
for(j in 1:n_unc) {
  zj      <- lm.alpha[j] + lm.beta[j] * x + rnorm( n, 0, lm.sigma[j] )
  EzHj[j] <- mean( zj[i_H] ) ; EzNotHj[j] <- mean( zj[i_NotH] )
  Vj[j]   <- EzNotHj[j] - EzHj[j]
  Rj[j]   <- pH * Vj[j]
}
R2   <- mean( Rj ) ; V2   <- mean( Vj )
s_R2 <- sd  ( Rj ) ; s_V2 <- sd  ( Vj )
```

```
PRA_Ms.Pa  <- c( pH, V , R , s_pH, s_V , s_R  )
PRA_Ms2.Pa <- c( pH, V2, R2, s_pH, s_V2, s_R2 )
```

17.3.2 Mortality

If we again use the Lindley and Smith (1972) equations followed by PRA, but this time for forest mortality rather than survival, we get the exact same results for parameters, PRA-results and uncertainties, apart from minor exceptions. [We show no code or detailed results for this.] These are some changes in sign (V, R) and in the posterior mean intercept-parameter for mortality whose value is 100 minus its equivalent for survival.

We can now also conclude that there is no need to centre the data before these analyses. This is because mortality is always close to 0% whereas survival clusters just below 100% without that affecting the results.

17.4 Summary of various PRAs

We only compare the single-threshold PRAs for *Picea abies*.

```
  rbind( PRA_Ss.Pa, PRA_Ds.Pa, PRA_Ms.Pa, PRA_Ms2.Pa )
>                   pH         V         R       s_pH       s_V       s_R
> PRA_Ss.Pa  0.7187500 0.5774396 0.4150347 0.07948043 0.2666308 0.1981961
> PRA_Ds.Pa  0.6604514 0.9449208 0.6225614 0.06920415 0.3177211 0.2160892
> PRA_Ms.Pa  0.7187500 0.8528872 0.6130127 0.07948043 0.4305022 0.3186045
> PRA_Ms2.Pa 0.7187500 0.8614740 0.6191845 0.07948043 0.4842879 0.3480819
```

18 Example VIII: Spatially Distributed Risk

Marcel Van Oijen and Brewer (2022) (Chapters 14, 15, 17, 18) used WorldClim weather data for Scotland plus a simplistic forest model to show application of PRA and BDT across a spatially diverse region. Here we repeat the same examples with the same model but using data for Germany.

Part of the R-code was based on https://damariszurell.github.io/EEC-QCB/Occ3_EnvData.html.

18.1 Chapter 14 of vO&B (2022)

18.1.1 Data

```
spdf_DEU <- gadm( country="DEU", level=0, path="data" ) # Germany
```

```
r_alt_DEU       <- elevation_30s( country='DEU', path='data', mask=F )
r_alt.hires_DEU <- crop( r_alt_DEU, ext_DEU )
r_alt_DEU       <- resample( r_alt.hires_DEU, s_prec_DEU )
```

```
r_tree_DEU <- landcover(var='trees', path='data', download=F) # Global
r_tree_DEU <- crop( r_tree_DEU, ext_DEU )
r_tree_DEU <- mask( r_tree_DEU, r_alt.hires_DEU )
```

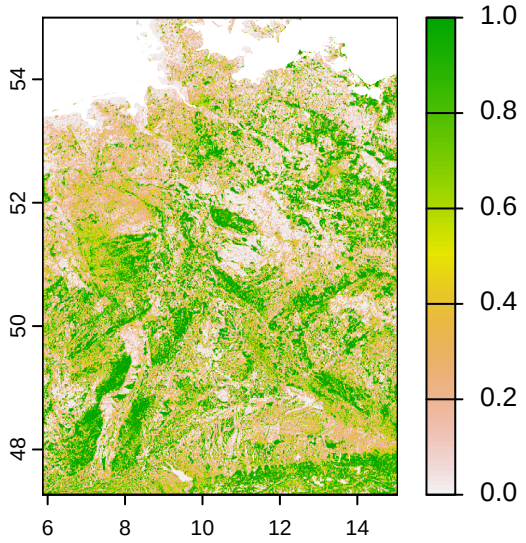


Figure 7: Tree cover in Germany in 2020 (ESA WorldCover data set).

18.1.2 A simple model for forest yield class (YC)

$$YC = 10 * (1 - exp(-R/1000)) * (1 - A/1000)$$

```
YC <- function( x1, x2 ) {
  alt <- x1  ; prec <- x2
  YC  <- max(0, 10 * (1-exp(-prec/1000)) * (1-alt/1000) )
  return( YC ) }
```

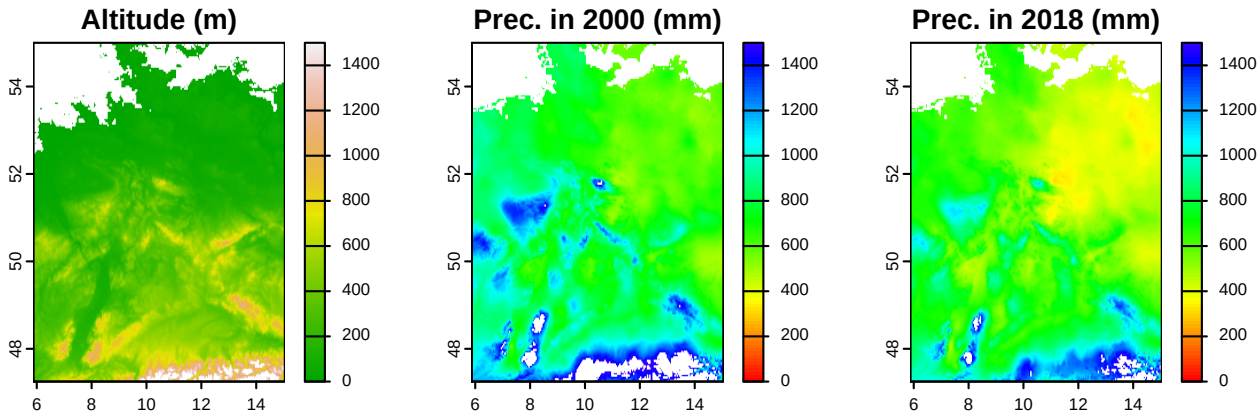


Figure 8: Germany: altitude and precipitation in 2000 and 2018.

```
r_prec_2000_DEU <- s_prec_DEU[[1]]
r_YC_DEU <- xapp( r_alt_DEU, r_prec_2000_DEU, fun=YC )
```

18.1.3 Emulation

```
n_design    <- 50
alt_design  <- runif( n_design, 0, 1000 )
prec_design <- runif( n_design, 0, 3000 )
YC_design   <- sapply( 1:n_design, function(i) {
  YC( alt_design[i], prec_design[i] ) } )
```

```
x1 <- alt_design   ; x2  <- prec_design          ; y  <- YC_design
ny <- length(y)
Vy <- 1            ; phi <- 1000
x  <- cbind(x1,x2) ; dx  <- as.matrix( dist(x) ) ; Sy <- exp( -dx/phi ) * Vy
```

```
GP.est <- function( x, y, Sy, mb, Sb, X=cbind(1,x) ) {
  Sb_y <- solve( solve(Sb) + t(X) %*% solve(Sy) %*% X )
  mb_y <- Sb_y %*% ( solve(Sb) %*% mb + t(X) %*% solve(Sy) %*% y )
  return( list( "mb_y"=mb_y, "Sb_y"=Sb_y ) ) }

mb   <- c(0,0,0) ; Sb <- diag(1,3) ; X <- cbind(1,x)
b_y  <- GP.est( x, y, Sy, mb=mb, Sb=Sb, X=X )
mb_y <- b_y$mb_y; Sb_y <- b_y$Sb_y
```

18.1.4 Application of the emulator

```
GP.pred <- function(x0,x,y,Sy,phi,mb_y,Sb_y,X0=c(1,x0),X=cbind(1,x)) {
  dx0  <- if( is.vector(x) ) { abs(x-x0)
         } else { sapply(1:length(y),function(i){dist(rbind(x0,x[i,]))}) }
  C0   <- Sy[1] * exp( -dx0/phi )
  m0_y <- X0 %*% mb_y - t(C0) %*% solve(Sy) %*% (X %*% mb_y - y)
```

```
    a <- X0 - t(C0) %*% solve(Sy) %*% X
  S0_y <- Sy[1] - t(C0) %*% solve(Sy) %*% C0 + a %*% Sb_y %*% t(a)
  return( list( "m0_y"=m0_y, "S0_y"=S0_y ) ) }

x0   <- c(0,0) ; X0 <- c(1,x0)
y0_y <- GP.pred( x0, x, y, Sy, phi, mb_y, Sb_y, X0=X0, X=X )
m0_y <- y0_y$m0_y ; S0_y <- y0_y$S0_y
```

```
YC_em <- function( x1, x2 ) {
  x0 <- cbind(x1,x2)
  YC <- GP.pred( x0, x, y, Sy, phi, mb_y, Sb_y,
                 X0=c(1,x0), X=cbind(1,x) )$m0_y
  return( YC ) }
YC_em.S <- function( x1, x2 ) {
  x0 <- cbind( x1, x2 )
  S  <- GP.pred( x0, x, y, Sy, phi, mb_y, Sb_y,
                 X0=c(1,x0), X=cbind(1,x) )$S0_y
  return( S ) }
```

```
r_YC_em_DEU   <- xapp( r_alt_DEU, r_prec_2000_DEU, fun=YC_em  )
r_YC_em.S_DEU <- xapp( r_alt_DEU, r_prec_2000_DEU, fun=YC_em.S  )
```

18.2 Chapter 15 of vO&B (2022)

```
ky <- 30 ; r_U_DEU <- r_YC_DEU * ky ; r_U_em_DEU <- r_YC_em_DEU * ky
```

```
IRRIG <- 500 ; ka <- 0.1

r_YC.IRRIG_DEU    <- xapp( r_alt_DEU, r_prec_2000_DEU + IRRIG, fun=YC    )
r_YC_em.IRRIG_DEU <- xapp( r_alt_DEU, r_prec_2000_DEU + IRRIG, fun=YC_em )
r_U.IRRIG_DEU     <- r_YC.IRRIG_DEU    * ky - IRRIG * ka
r_U_em.IRRIG_DEU  <- r_YC_em.IRRIG_DEU * ky - IRRIG * ka
```

18.2.1 Multiple action levels

```
nI    <- 10 ; layers <- 1:nI ; IRRIG <- (layers-1)*50

s_U.a_DEU <- NULL
for(i in layers) {
  r_YC.a_DEU <- xapp( r_alt_DEU, r_prec_2000_DEU + IRRIG[i], fun=YC )
  r_U.a_DEU  <- r_YC.a_DEU * ky - IRRIG[i] * ka
  s_U.a_DEU  <- c( s_U.a_DEU, r_U.a_DEU ) }
s_U.a_DEU <- rast(s_U.a_DEU)
a.opt_DEU <- (which.max( s_U.a_DEU ) - 1) * 50
```

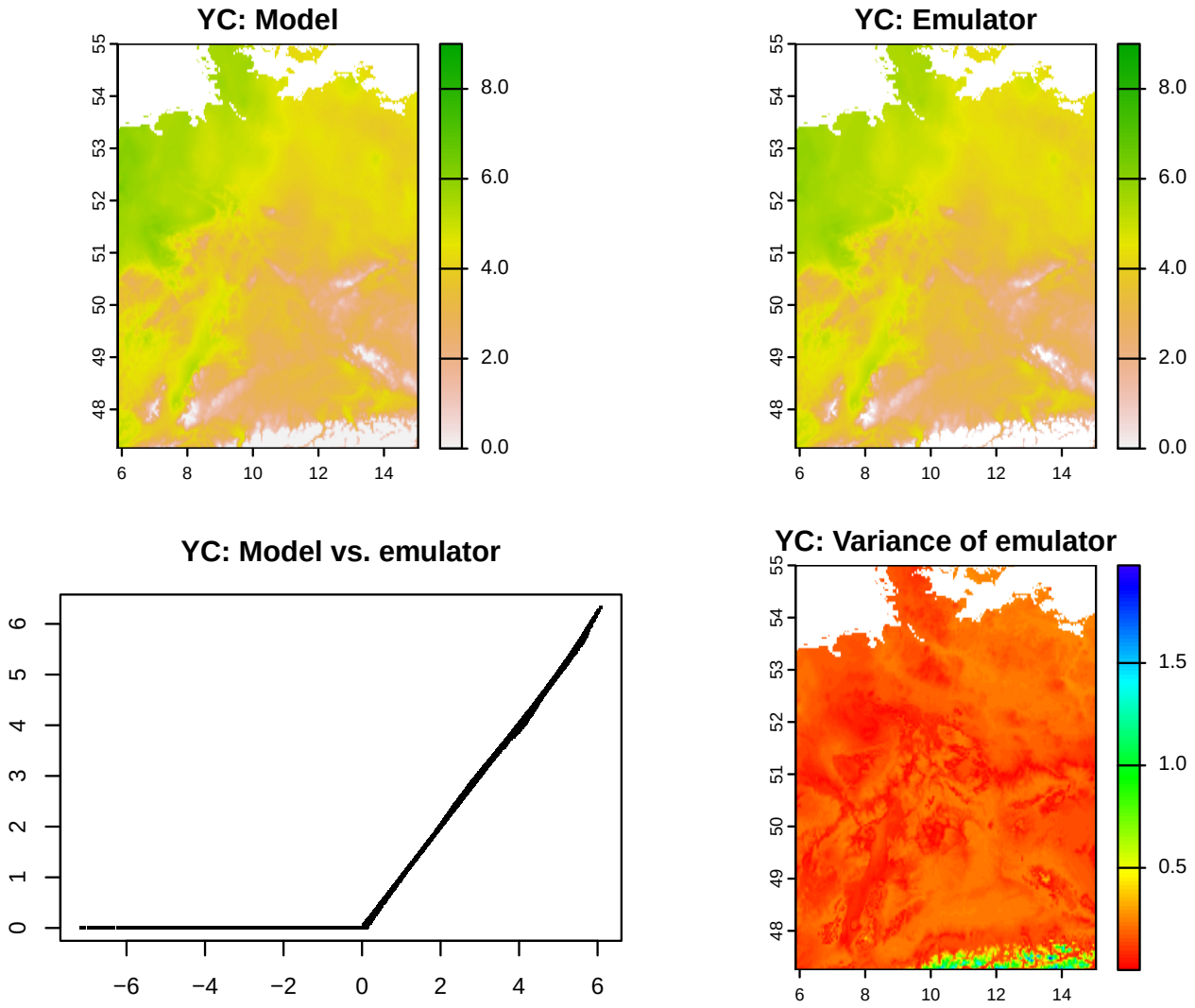


Figure 9: Top left: outputs from the original nonlinear model for yield class (YC, $m^3\,ha^{-1}\,y^{-1}$) applied to precipitation data from the year 2000. Top right: emulated YC. Bottom left: original outputs vs. emulated values. Bottom right: emulator uncertainty (kriging variance).

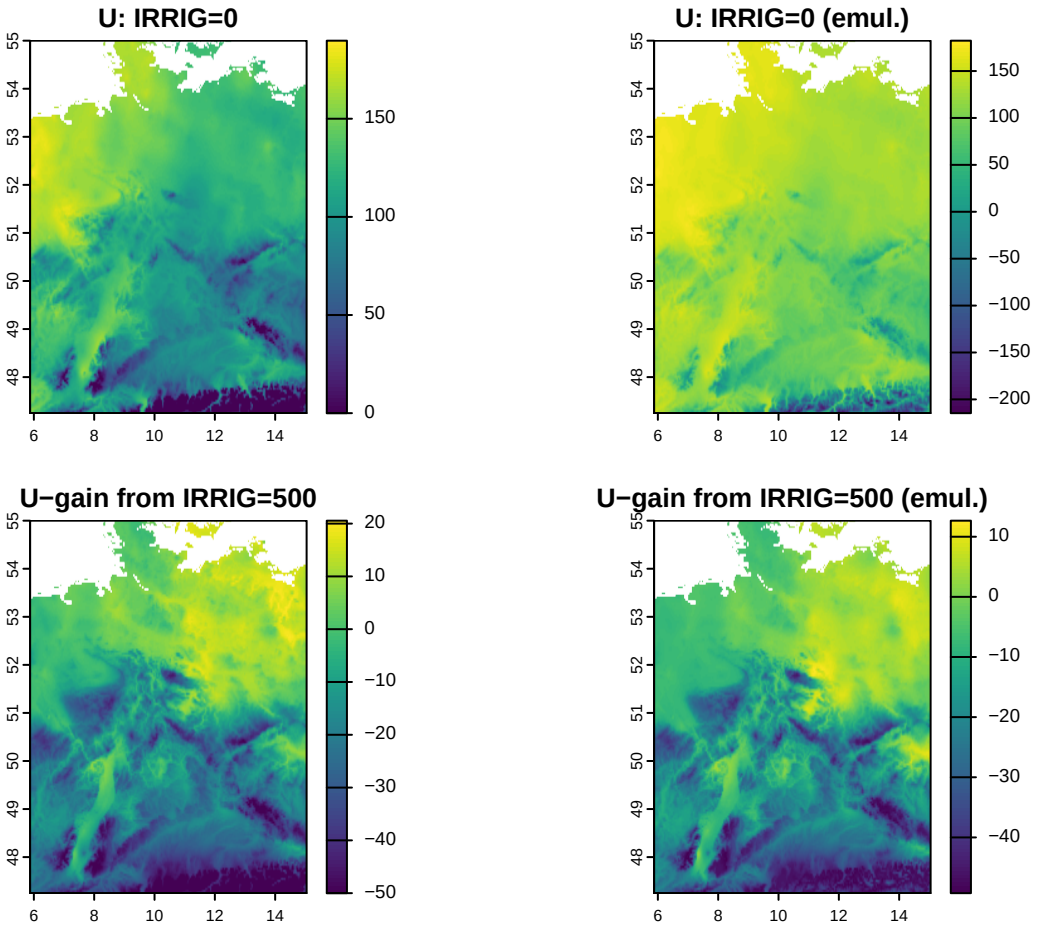


Figure 10: Top: utility in the year 2000 without irrigation. Bottom: gain in utility with irrigation of 500 $mm\,y^{-1}$. Left: original model. Right: emulator.

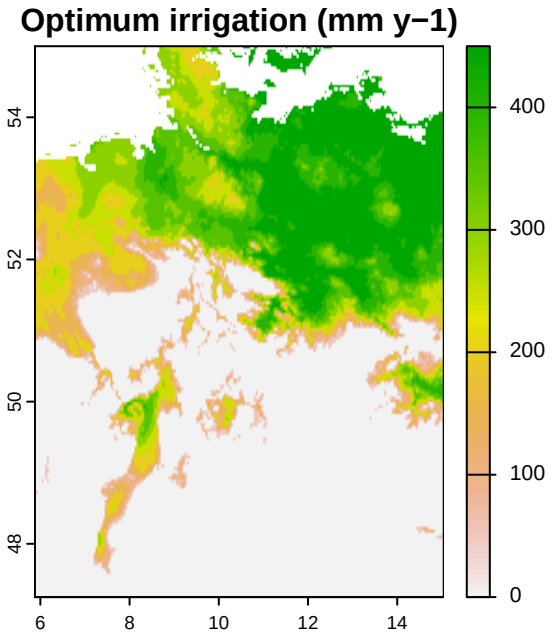


Figure 11: Outcome from a decision problem where action choice was between a large number of irrigation levels.

18.3 Chapter 17 of vO&B (2022)

```
IRRIG <- 500

s_YC_DEU       <- rast( sapply(1:19, function(i){
  xapp( r_alt_DEU, s_prec_DEU[[i]]      , fun=YC )}) )
s_YC.IRRIG_DEU <- rast( sapply(1:19, function(i){
  xapp( r_alt_DEU, s_prec_DEU[[i]]+IRRIG, fun=YC )}) )
```

```
ky        <- 30 ; ka <- 0.1

s_U_DEU       <- s_YC_DEU       * ky
s_U.IRRIG_DEU <- s_YC.IRRIG_DEU * ky - IRRIG * ka
```

```
par( mfrow=c(2,4), mar=c(1,0,1,0) )

years <- c(1:2,NA,19)

for( y in years ) {
  if(is.na(y)) plot(-1:1,rep(0,3),axes=0,xlim=c(-5,5),pch=19)
  else { plot( s_U_DEU[[y]], range=c(0,230),
               main=paste0(y+1999,"\nIRRIG=0"), xaxt='n', yaxt='n' ) } }

for( y in years ) {
  if(is.na(y)) plot(-1:1,rep(0,3),axes=0,xlim=c(-5,5),pch=19)
  else { plot( s_U.IRRIG_DEU[[y]], range=c(0,230),
               main=paste0(y+1999,"\nIRRIG=",IRRIG), xaxt='n', yaxt='n' ) } }
```

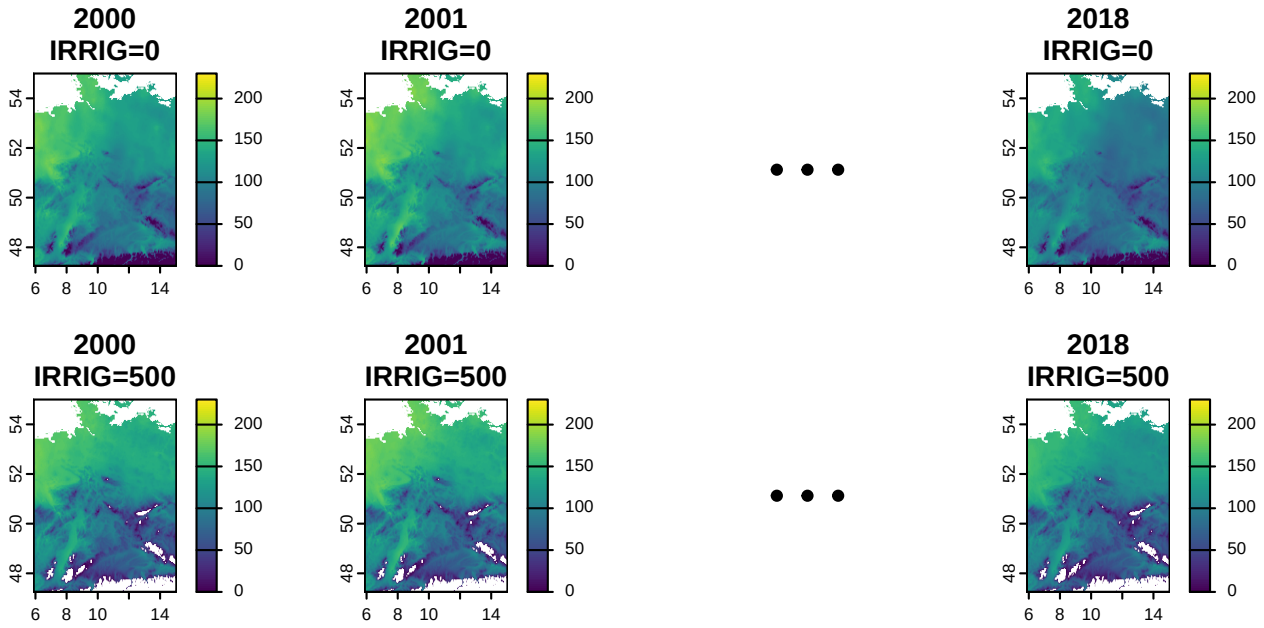


Figure 12: Utility in 2000, 2001 and 2018. Top row: no irrigation. Bottom row: irrigation at 500 $mm\,y^{-1}$.

```
r_U_DEU       <- mean( s_U_DEU )
r_U.IRRIG_DEU <- mean( s_U.IRRIG_DEU )
```

```
par( mfrow=c(1,3), mar=c(1,1,1,3) )

plot( r_U_DEU                , main="E[ u | IRRIG=0 ]",
      xaxt='n', yaxt='n', range=c(  0,180) )
plot( r_U.IRRIG_DEU          , main=paste0("E[ u | IRRIG=",IRRIG," ]"),
      xaxt='n', yaxt='n', range=c(  0,180) )
plot( r_U.IRRIG_DEU - r_U_DEU, main="d(u)",
      xaxt='n', yaxt='n', range=c(-50, 20) )
```

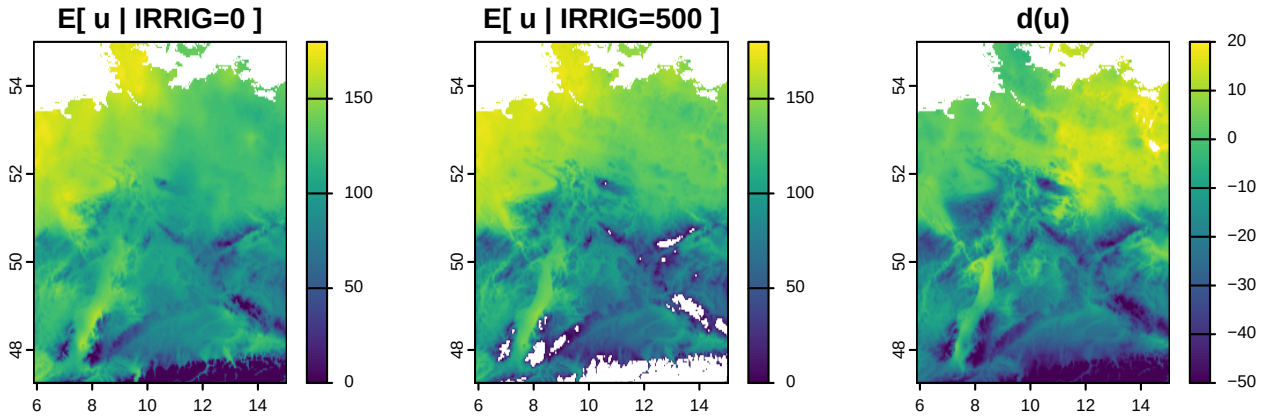


Figure 13: Mean utility. Left: no irrigation. Middle: irrigation = 500 $mm\,y^{-1}$. Right: utility-gain from irrigation.

```
PRA.Ez_notH <- function( x, z, thr=1000, rel=F ) {
  n       <- length(x) ; H  <- which(x < thr) ; notH   <- which(x >= thr)
  n_H     <- length(H) ; pH <- n_H / n        ; n_notH <- length(notH)
  Ez_H    <- mean( z[H]    ) ; s_Ez_H    <- sqrt( var(z[H   ]) / n_H    )
  Ez_notH <- mean( z[notH] ) ; s_Ez_notH <- sqrt( var(z[notH]) / n_notH )
  if(n_H==0){ Ez_H <- min(z) } ; if(n_notH==0){ Ez_notH <- max(z) }
  V       <- Ez_notH - Ez_H  ; R         <- pH * V
  s_pH    <- sqrt( pH*(1-pH) / n )
  s_V     <- sqrt( s_Ez_H^2 + s_Ez_notH^2 )
  s_R     <- sqrt( s_pH^2*s_V^2 + s_pH^2*V^2 + pH^2*s_V^2 )
  if(rel){V <- V/Ez_notH ; R=R/Ez_notH; s_V=s_V/Ez_notH; s_R=s_R/Ez_notH}
  return( c(Ez_notH=Ez_notH, pH=pH, V=V, R=R, s_pH=s_pH, s_V=s_V, s_R=s_R) )
}
```

```
s_PRA_DEU           <- xapp( s_prec_DEU      , s_U_DEU      , fun=PRA.Ez_notH )
s_PRA.IRRIG_DEU     <- xapp( s_prec_DEU+IRRIG, s_U.IRRIG_DEU, fun=PRA.Ez_notH )
r_Ez_notH_DEU       <- s_PRA_DEU$Ez_notH
r_Ez_notH.IRRIG_DEU <- s_PRA.IRRIG_DEU$Ez_notH
r_pH_DEU            <- s_PRA_DEU$pH
r_pH.IRRIG_DEU      <- s_PRA.IRRIG_DEU$pH
r_pH_DEU            <- mask(r_pH_DEU,r_alt_DEU)
r_pH.IRRIG_DEU      <- mask(r_pH.IRRIG_DEU,r_alt_DEU)
r_V_DEU <- s_PRA_DEU$V ; r_V.IRRIG_DEU <- s_PRA.IRRIG_DEU$V
r_R_DEU <- s_PRA_DEU$R ; r_R.IRRIG_DEU <- s_PRA.IRRIG_DEU$R
```

```
s_PRA_DEU2           <- xapp( s_prec_DEU      , s_U_DEU      , fun=PRA.Ez_notH, rel=T )
s_PRA.IRRIG_DEU2     <- xapp( s_prec_DEU+IRRIG, s_U.IRRIG_DEU, fun=PRA.Ez_notH, rel=T )
r_Ez_notH_DEU2       <- s_PRA_DEU2$Ez_notH
r_Ez_notH.IRRIG_DEU2 <- s_PRA.IRRIG_DEU2$Ez_notH
r_pH_DEU2            <- s_PRA_DEU2$pH
r_pH.IRRIG_DEU2      <- s_PRA.IRRIG_DEU2$pH
r_pH_DEU2            <- mask(r_pH_DEU2,r_alt_DEU)
r_pH.IRRIG_DEU2      <- mask(r_pH.IRRIG_DEU2,r_alt_DEU)
r_V_DEU2 <- s_PRA_DEU2$V ; r_V.IRRIG_DEU2 <- s_PRA.IRRIG_DEU2$V
r_R_DEU2 <- s_PRA_DEU2$R ; r_R.IRRIG_DEU2 <- s_PRA.IRRIG_DEU2$R
```

```
r_Rc_DEU       <- r_R_DEU       - r_Ez_notH_DEU
r_Rc.IRRIG_DEU <- r_R.IRRIG_DEU - r_Ez_notH.IRRIG_DEU
```

```
par( mfrow=c(2,3), mar=c(1,1,1,3) )

plot( r_pH_DEU      , main="pH", xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0, 1) ) ; plot( gadm5, add=T)
plot( r_V_DEU       , main="V" , xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0,50) ) ; plot( gadm5, add=T)
plot( r_R_DEU       , main="R" , xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0,50) ) ; plot( gadm5, add=T)

plot( r_pH.IRRIG_DEU, main="pH.IRRIG", xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0, 1) ) ; plot( gadm5, add=T)
plot( r_V.IRRIG_DEU , main="V.IRRIG" , xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0,50) ) ; plot( gadm5, add=T)
```

```
plot( r_R.IRRIG_DEU , main="R.IRRIG" , xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0,50) ) ; plot( gadm5, add=T)
```

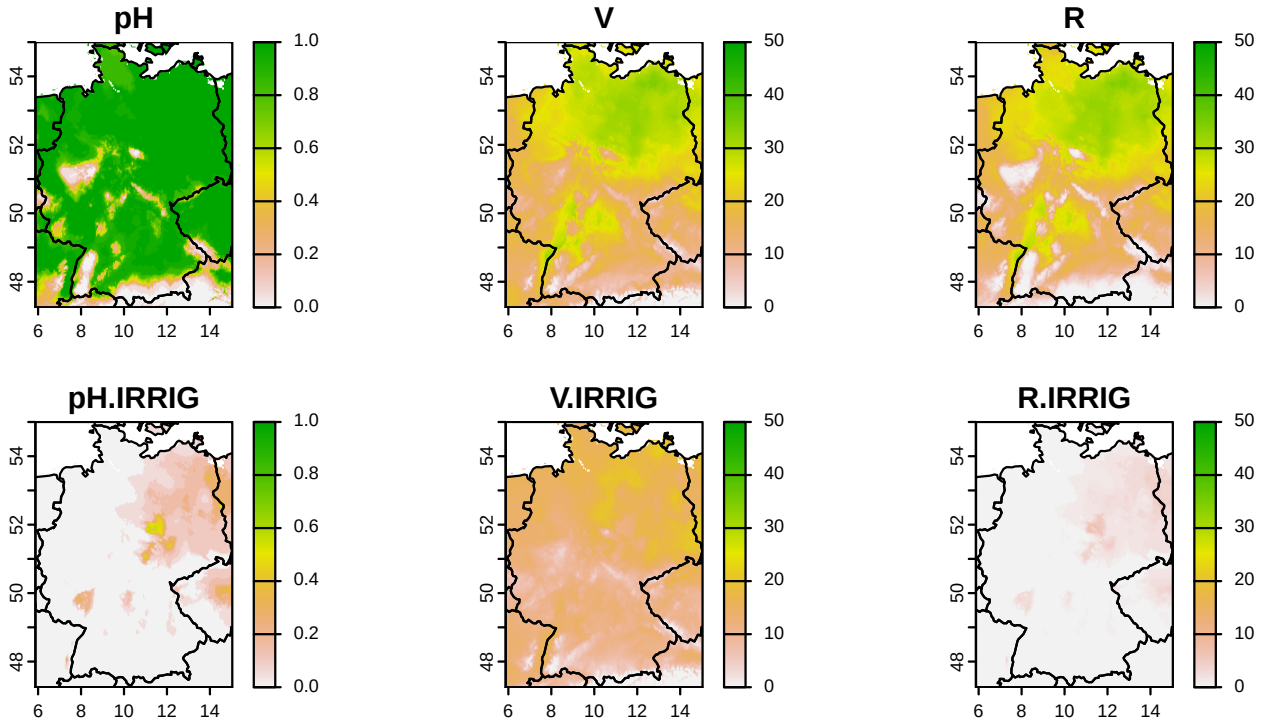


Figure 14: PRA based on data for 2000-2018. Top row: no irrigation. Bottom row: irrigation = 500 $mm\,y^{-1}$.

```
par( mfrow=c(2,3), mar=c(1,1,1,3) )

plot( r_pH_DEU2      , main="pH", xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0, 1) ) ; plot( gadm5, add=T)
plot( r_V_DEU2       , main="V" , xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0, 0.25) ) ; plot( gadm5, add=T)
plot( r_R_DEU2       , main="R" , xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0, 0.25) ) ; plot( gadm5, add=T)

plot( r_pH.IRRIG_DEU2, main="pH.IRRIG", xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0, 1) ) ; plot( gadm5, add=T)
plot( r_V.IRRIG_DEU2 , main="V.IRRIG" , xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0, 0.25) ) ; plot( gadm5, add=T)
plot( r_R.IRRIG_DEU2 , main="R.IRRIG" , xaxt='n',yaxt='n',
      col=terrain.colors(100,rev=T), range=c( 0, 0.25) ) ; plot( gadm5, add=T)
```

```
par( mfrow=c(1,3), mar=c(1,1,1,3) )

plot( r_Rc_DEU                , main="Rc"      ,
      xaxt='n', yaxt='n', range=c(-270, 0) )    ; plot( gadm5, add=T)
plot( r_Rc.IRRIG_DEU          , main="Rc.IRRIG",
      xaxt='n', yaxt='n', range=c(-270, 0) )    ; plot( gadm5, add=T)
```

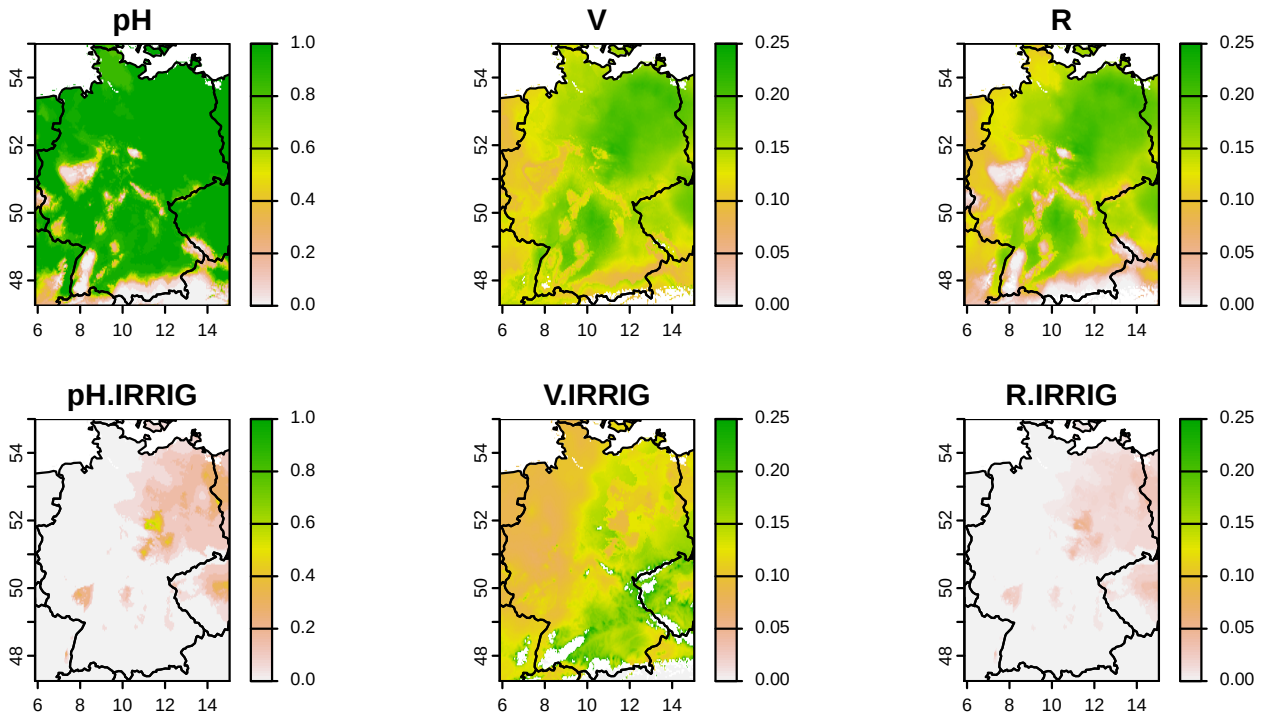


Figure 15: Relative PRA based on data for 2000-2018. Top row: no irrigation. Bottom row: irrigation = 500 $mm\,y^{-1}$.

```
plot( r_Rc_DEU - r_Rc.IRRIG_DEU, main="-d(Rc)"   ,
      xaxt='n', yaxt='n', range=c( -50,20) )     ; plot( gadm5, add=T)
```

18.4 Chapter 18 of vO&B (2022)

```
exposure <- function( pH, ... ){
  pH <- pH[!is.na(pH)]          ; n <- length(pH)
  nE <- length( pH[pH>1/21] ) ; Q <- nE/n
  return( Q ) }
```

```
r_Q4_DEU <- aggregate( r_pH_DEU, fact=4, fun=exposure )
```

```
s_prec4_DEU <- aggregate( s_prec_DEU, fact=4, fun="c" )
s_U4_DEU    <- aggregate( s_U_DEU   , fact=4, fun="c" )
  msk_DEU   <- ifel( r_Q4_DEU>0 & !is.na(s_prec4_DEU[[1]]), 1, NA )
s_PRA4_DEU  <- xapp( s_prec4_DEU, s_U4_DEU, fun=PRA.Ez_notH )
r_pH4_DEU   <- s_PRA4_DEU$pH ; r_pH4_DEU <- mask( r_pH4_DEU, msk_DEU )
r_V4_DEU    <- s_PRA4_DEU$V  ; r_V4_DEU  <- mask( r_V4_DEU , msk_DEU )
r_R4_DEU    <- s_PRA4_DEU$R  ; r_R4_DEU  <- mask( r_R4_DEU , msk_DEU )
```

```
par( mfrow=c(2,3), mar=c(1,1,1,3) )

plot( mean(s_prec4_DEU), main="mean(prec)", xaxt="n", yaxt="n",
```

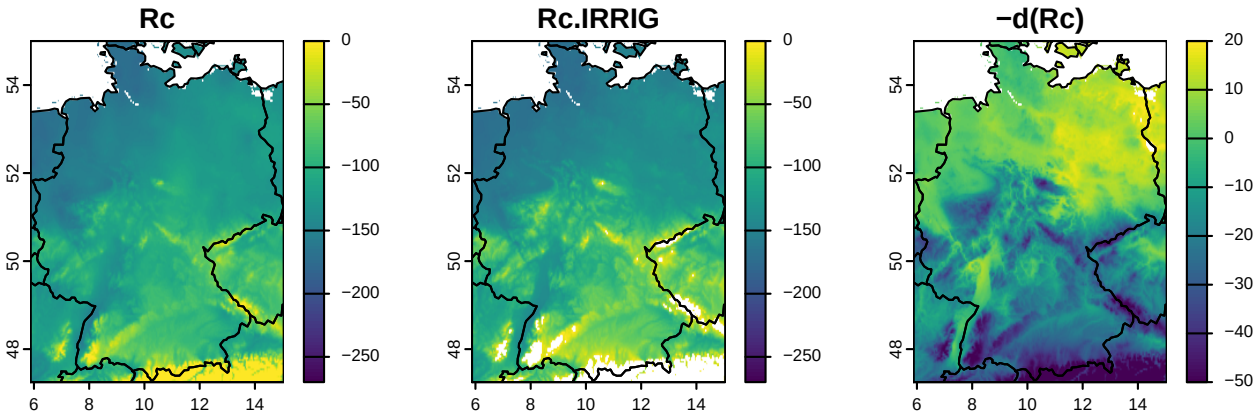
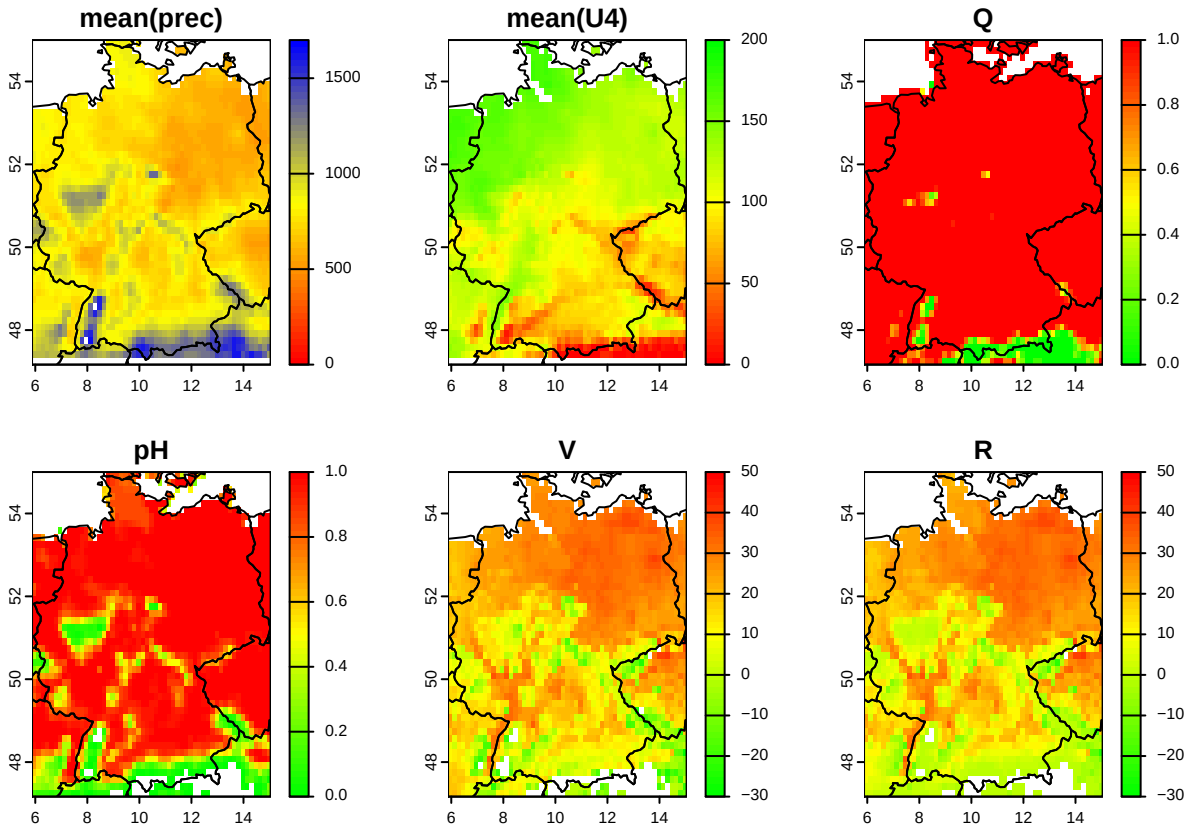


Figure 16: Corrected risk for utility. Left: no irrigation. Middle: irrigation as above. Right: decrease in corrected risk due to irrigation.

```
      col=map.pal("ryb"), range=c(0,1700))   ; plot( gadm5, add=T)
plot( mean(s_U4_DEU   ), main="mean(U4)"  , xaxt="n", yaxt="n",
      col=map.pal("ryg"), range=c(0, 200))   ; plot( gadm5, add=T)

plot( r_Q4_DEU , main="Q" , xaxt="n", yaxt="n",
      col=map.pal("gyr"), range=c(   0, 1) ) ; plot( gadm5, add=T)
plot( r_pH4_DEU, main="pH", xaxt='n',yaxt='n',
      col=map.pal("gyr"), range=c(   0, 1) ) ; plot( gadm5, add=T)
plot( r_V4_DEU , main="V" , xaxt='n',yaxt='n',
      col=map.pal("gyr"), range=c( -30,50) ) ; plot( gadm5, add=T)
plot( r_R4_DEU , main="R" , xaxt='n',yaxt='n',
      col=map.pal("gyr"), range=c( -30,50) ) ; plot( gadm5, add=T)
```


mean(prec)
mean(U4)
Q
pH
V
R

19 BDT

We denote the expectation for utility as $\overline{u}(a)$, and the action that maximises it as a^*.

$$\begin{aligned} \overline{u}(a) &= E[u(a)] \\ &= \sum p[x]\, u(a,x), \\ \overline{u}(a^*) &= \max_{a \in A} \overline{u}(a). \end{aligned} \tag{6}$$

19.1 Value of Information

Eq. (7) shows the maximum expectation for utility when accounting for specific, general and perfect information:

$$\begin{aligned} \overline{u}(a_y^*) &= \max_{a \in A} \sum p[x|y]\, u(a,x), \\ \overline{u}(a_Y^*) &= \sum_{y \in Y} p[y]\, \overline{u}(a_y^*) \\ &= \sum_{y \in Y} \max_{a \in A} \sum p[x]\, p[y|x]\, u(a,x), \\ \overline{u}(a_I^*) &= \sum p[x] \max_{a \in A}\, u(a,x). \end{aligned} \tag{7}$$

The "Value of Information" is defined as the degree to which informed expectation for utility exceeds its prior value:

$$\begin{aligned} VoI_{partial} &= \overline{u}(a_Y^*) - \overline{u}(a^*), \\ VoI_{perfect} &= \overline{u}(a_I^*) - \overline{u}(a^*). \end{aligned} \tag{8}$$

If the data y are indiscriminate, i.e. the likelihood function $p[y|x]$ is uniform, then $VoI_{partial}$ is zero:

$$\begin{aligned} \forall x \in X, \quad p[y_i|x] &= \frac{1}{n}; \quad i = 1..n \\ \implies \overline{u}(a_Y^*) &= \sum_{y \in Y} \max_{a \in A} \sum p[x]\, p[y|x]\, u(a,x) \\ &= n \times \max_{a \in A} \sum p[x]\, \frac{1}{n}\, u(a,x) \\ &= \max_{a \in A} \sum p[x]\, u(a,x) \\ &= \overline{u}(a^*) \\ \implies VoI_{partial} &= \overline{u}(a_Y^*) - \overline{u}(a^*) = 0. \end{aligned} \tag{9}$$

19.1.1 Example and exercise

If both a and x are binary, we have the following simple utility table:

	$x = x_1$	$x = x_2$	$\overline{u}(a_i) = \sum p[x_i]\, u(a_2, x_i)$
$a = a_1$	$u(a_1, x_1)$	$u(a_1, x_2)$	$\overline{u}(a_1)$
$a = a_2$	$u(a_2, x_1)$	$u(a_2, x_2)$	$\overline{u}(a_2)$

Let's make this concrete with the following example, where u is tree survival (%), and both x (beetles) and a (tree thinning) are binary. Here is the utility table:

Tree Survival	No beetles	Beetles	$\overline{u}(a_i)$
No thinning	100%	40%	$\overline{u}(a_1)$
50% thinning	50%	40%	$\overline{u}(a_2)$

EXERCISE

Assume that our only uncertainty is about the beetles for which $p[x = x_2 = Beetles]$ equals 0.7.

- (a) Complete the table by calculating $\overline{u}(a_1)$ and $\overline{u}(a_2)$.
- (b) So which action would you choose at that point: no thinning or 50% thinning?
- (c) Now assume you may decide to get more information first. You could measure a certain binary variable $y \in \{y_1, y_2\}$ for which $p[y_1|x_1] = p[y_2|x_2] = 0.6$. What is that information worth, i.e. what is $VoI_{partial}$ in this case? Express the answer in expected gain of tree survival (%).
- (d) What would be the value of perfect information about the beetles here, so what is $VoI_{perfect}$?

ANSWERS

- (a) $\overline{u}(a_1) = 0.3 \times 100 + 0.7 \times 10 = 37\%$, $\overline{u}(a_2) = 0.3 \times 50 + 0.7 \times 40 = 43\%$.
- (b) So utility is maximised by deciding to thin, i.e. action a_2.
- (c) $\overline{u}(a^*_Y) = \sum_{y \in Y} \max_{a \in A} \sum p[x]\, p[y|x]\, u(a, x) = \max(20.8, 20.2) + \max(16.2, 22.8) = 43.6$. So $VoI_{partial}$ is only 43.6 - 43 = 0.6%, reflecting the imprecision of the test where the two likelihoods $p[y_i|x_i]$ are both only 0.6.
- (d) $\overline{u}(a^*_I) = \sum p[x] \max_{a \in A}\, u(a, x) = 0.3 \times 100 + 0.7 \times 40 = 58\%$. So $VoI_{perfect}$ is 58 - 43 = 15%.

19.2 Graphical model for decision-making

The graphical model for BDT shown in Fig. 17 tells us that we have to provide probability distributions for the inputs x, θ and ϵ if we want to calculate $\overline{u}(a)$. It also tells us that the three inputs are independent from each other, so we can write:

$$\begin{aligned} \overline{u}(a) &= \sum_{x,\theta,\epsilon} p[x, \theta, \epsilon]\, u(a, x, \theta, \epsilon), \\ \texttt{where} \quad p[x, \theta, \epsilon] &= p[x]\, p[\theta]\, p[\epsilon]. \end{aligned} \tag{10}$$

Generally we shall have to do our calculations numerically by generating a large sample of input vectors and calculating the associated utility for each of them.

19.2.1 GM exercise

Start from a GM consisting of three binary variables: A -> B -> C. To keep notation brief, instead of writing "A=1" or "A=0", we shall write "A" resp. "a". So p[A] is the probability that the first node has the value 1, p[a] is its probability of being 0, etc.

Say that the prior for the first node is p[A]=0.9. The conditional probability distribution for the next node is { p[B|A] = 0.9, p[B|a] = 0.6 } and for C it is { p[C|B] = 0.9, p[C|b] = 0.2 }. This defines the prior probability distribution for the whole network (chain).

QUESTIONS:

1. What are the prior probabilities for B and C?

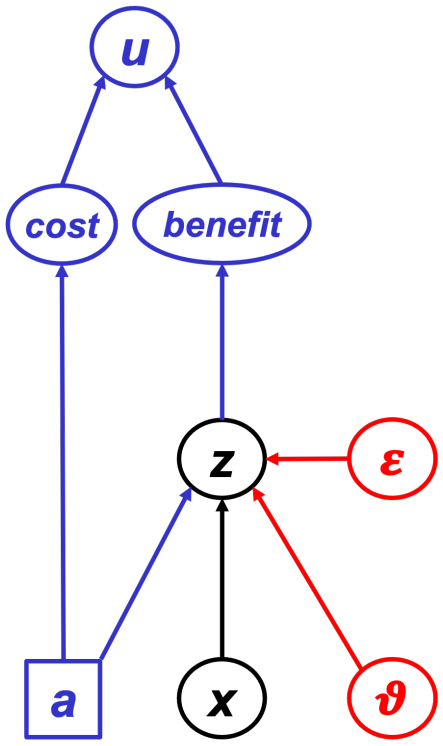


Figure 17: A graphical model for BDT.

We get information about one of the variables. What is then our posterior for the other two?

2. Say we learn that the first node has the value 0. What are p[B|a] and p[C|a]?
3. Say we learn instead that the second node has the value 0. What are p[A|b] and p[C|b]?
4. Say we learn instead that the third node has the value 0. What are p[A|c] and p[B|c]?

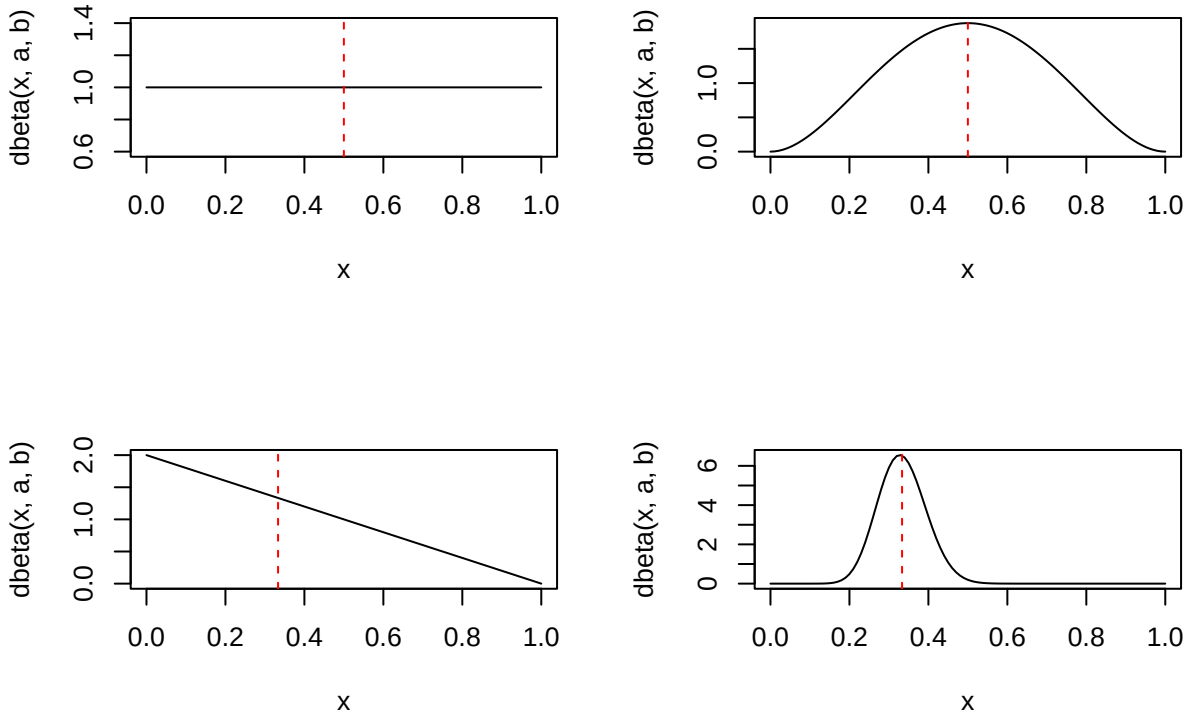
```
pA <- 0.9 ; pB_A <- 0.9 ; pB_a <- 0.6 ; pC_B <- 0.9 ; pC_b <- 0.2

# Analytical solutions using Law of Total Probability (LTP) and Bayes' Theorem (BT):
pB   <- pA*pB_A + (1-pA)*pB_a      # = 0.9 *0.9  + 0.1 *0.6 = 0.87        # LTP
pC   <- pB*pC_B + (1-pB)*pC_b      # = 0.87*0.9  + 0.13*0.2 = 0.809       # LTP
# pB_a <- 0.6                                                            # Prior
pC_a <- pB_a*pC_B + (1-pB_a)*pC_b # = 0.6 *0.9  + 0.4 *0.2 = 0.62        # LTP
pA_b <- pA*(1-pB_A) / (1-pB)       # = 0.9 *0.1  / 0.13     = 9/13        # BT
# pC_b <- 0.2                                                            # Prior
pB_c <- pB*(1-pC_B) / (1-pC)       # = 0.87*0.1  / 0.191    = 87/191      # BT
  pc_A <- pB_A*(1-pC_B) + (1-pB_A)*(1-pC_b) # = 0.9*0.1 + 0.1*0.8 = 0.17 # LTP
pA_c <- pA*pc_A     / (1-pC)       # = 0.9 *0.17 / 0.191    = 153/191     # BT
solA <- rbind( c(pB,pC), c(pB_a,pC_a), c(pA_b,pC_b), c(pA_c,pB_c) )

# Numerical solutions:
set.seed(1)
n    <- 1e6
A    <- rbinom( n, 1, pA )
B    <- rbinom( n, 1, A*pB_A + (1-A)*pB_a )
C    <- rbinom( n, 1, B*pC_B + (1-B)*pC_b )
```

```
  pB   <- sum(B==1) / n                ; pC   <- sum(C==1) / n
  pB_a <- sum(A==0 & B==1) / sum(A==0) ; pC_a <- sum(A==0 & C==1) / sum(A==0)
  pA_b <- sum(A==1 & B==0) / sum(B==0) ; pC_b <- sum(B==0 & C==1) / sum(B==0)
  pB_c <- sum(B==1 & C==0) / sum(C==0) ; pA_c <- sum(A==1 & C==0) / sum(C==0)
  solN <- rbind( c(pB,pC), c(pB_a,pC_a), c(pA_b,pC_b), c(pA_c,pB_c) )

  # Comparison:
  round( cbind( solA, solN ), 2 )
>      [,1] [,2] [,3] [,4]
> [1,] 0.87 0.81 0.87 0.81
> [2,] 0.60 0.62 0.60 0.62
> [3,] 0.69 0.20 0.69 0.20
> [4,] 0.80 0.46 0.80 0.46
```

20 APPENDIX: Additional R-code

20.1 Beta distribution (can be used as prior for scalar p[H] and Q)

```
par( mfrow=c(2,2) )
a <-  1 ; b <-  1 ; curve( dbeta(x,a,b) ) ; abline(v=a/(a+b),col="red",lty=2)
a <-  3 ; b <-  3 ; curve( dbeta(x,a,b) ) ; abline(v=a/(a+b),col="red",lty=2)
a <-  1 ; b <-  2 ; curve( dbeta(x,a,b) ) ; abline(v=a/(a+b),col="red",lty=2)
a <- 20 ; b <- 40 ; curve( dbeta(x,a,b) ) ; abline(v=a/(a+b),col="red",lty=2)
```



20.2 Dirichlet distribution (can be used as prior for vector p[H])

The mean vector of a Dirichlet distribution $Di_k[\alpha]$ is α/α_0 where $A = \sum_{i=1}^{k} \alpha_i$. Its covariance matrix consists of variances $\sigma_{ii} = (\alpha_i/\alpha_0)(1-\alpha_i/\alpha_0)/(\alpha_0+1)$, and covariances $\sigma_{ij} = -(\alpha_i/\alpha_0)(\alpha_j/\alpha_0)\,/\,(\alpha_0+1); \quad i \neq j)$.

Let's check that the mean and variance of a sample x generated using the `rdirch`-function from the `nimble`-package is close to the theoretical mean vector and covariance matrix of a $Di_4[a = (1,1,1,1)]$ distribution:

```
n_points <- 1e3 ; x0 <- matrix(NA, nrow=n_points, ncol=4 )
a0       <- rep(1,4)
for(i in 1:n_points) x0[i,] <- rdirch( 1, a=a0 )
```

We compare the mean and covariance matrix of the generated dataset to the theoretical expectation:

```
mS_Di <- function(a) {
  k <- length(a) ; A <- sum(a) ; m <- a/A
  S <- matrix(NA,nrow=k,ncol=k)
  for(r in 1:k){
```

```
      for(c in 1:k){ S[r,c] <- ((r==c)*a[r]/A - (a[r]/A)*(a[c]/A)) / (A+1) }
    }
    return( list( m=m, S=S ) )
  }

  colMeans(x0) ; cov(x0)
> [1] 0.2488705 0.2439417 0.2597409 0.2474468
>              [,1]         [,2]        [,3]        [,4]
> [1,]  0.036647880 -0.008270744 -0.01498236 -0.01339478
> [2,] -0.008270744  0.032590677 -0.01204849 -0.01227144
> [3,] -0.014982359 -0.012048493  0.03961870 -0.01258785
> [4,] -0.013394776 -0.012271439 -0.01258785  0.03825406
  mS_Di(a0)
> $m
> [1] 0.25 0.25 0.25 0.25
>
> $S
>         [,1]    [,2]    [,3]    [,4]
> [1,]  0.0375 -0.0125 -0.0125 -0.0125
> [2,] -0.0125  0.0375 -0.0125 -0.0125
> [3,] -0.0125 -0.0125  0.0375 -0.0125
> [4,] -0.0125 -0.0125 -0.0125  0.0375
```

The `rdirch`-function seems to work as intended.

20.3 Wishart distribution (used as prior for bivariate Gaussian precision matrix)

`rWishart(n,df,Sigma)` generates an array of n different matrices. The mean of the matrices is `df * Sigma`. So if we are after a sample of matrices varying around a matrix `sgm`, we can generate them by calling `rWishart` with `df = a` and `Sigma = sgm/a`:

```
  set.seed(1)
  N     <- 100 ; a <- 2 ; sgm <- toeplitz( (2:1)/2 )
  smpl1 <- rWishart( n=N, df=a, Sigma=sgm/a )
```

```
> sgm:
>      [,1] [,2]
> [1,]  1.0  0.5
> [2,]  0.5  1.0
>
> Two of the generated matrices:
> , , 1
>
>       [,1]  [,2]
> [1,] 0.155 0.384
> [2,] 0.384 2.098
>
> , , 2
>
>        [,1]   [,2]
> [1,]  0.836 -0.102
```

```
> [2,] -0.102  0.016
>
> The sample's mean matrix is:
>       [,1]  [,2]
> [1,] 1.054 0.561
> [2,] 0.561 1.007
```

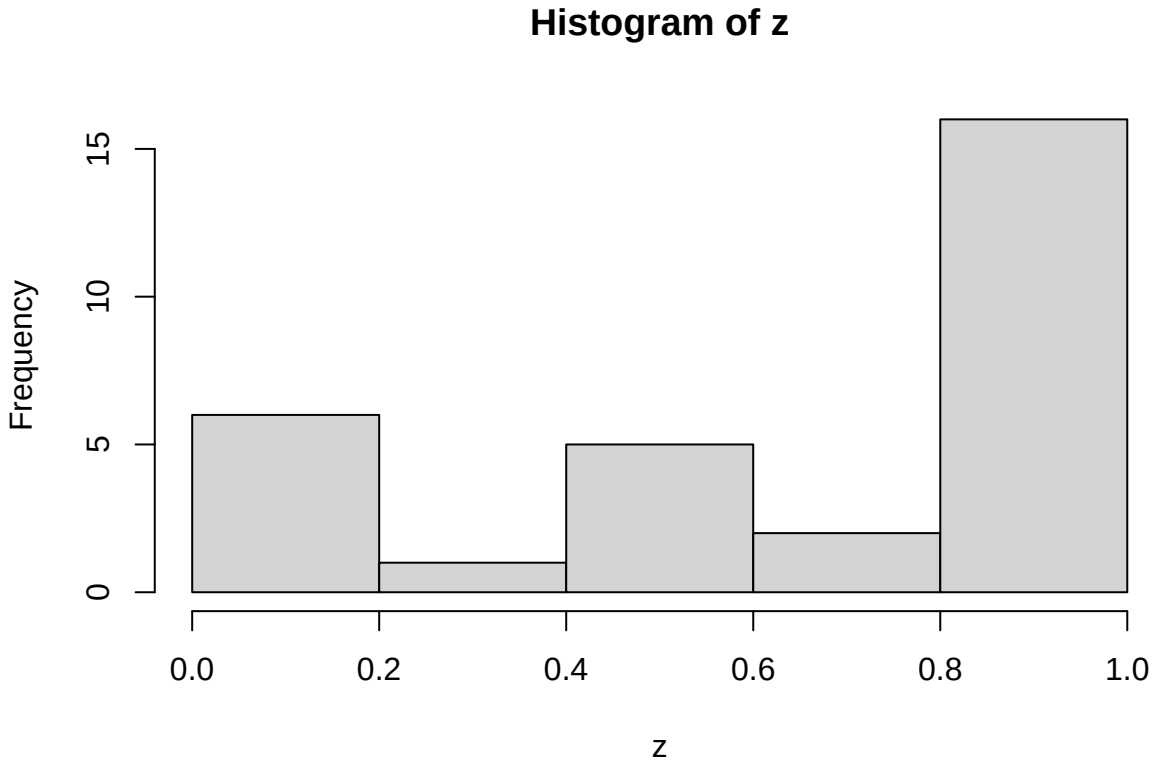
The matrices will be clustered more closely around `sgm` if we choose a higher value of `a`.

```
a     <- 100
smpl2 <- rWishart( n=N, df=a, Sigma=sgm/a )
```

```
>
> Two of the generated matrices:
> , , 1
>
>       [,1]  [,2]
> [1,] 0.886 0.635
> [2,] 0.635 1.123
>
> , , 2
>
>       [,1]  [,2]
> [1,] 1.227 0.689
> [2,] 0.689 1.262
>
> The sample's mean matrix is:
>      [,1]  [,2]
> [1,]  1.0 0.500
> [2,]  0.5 0.987
```

20.4 GLM

```
x1 <- rnorm( 30 )
x2 <- rnorm( 30, 0, 10 )
LP <- 1 + x1 + x2/5
z  <- 1 / ( 1 + exp(-LP) ) ; hist(z)
```




```
  glm(z ~ x1 + x2, family=binomial(link=logit) )
> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
>
> Call:  glm(formula = z ~ x1 + x2, family = binomial(link = logit))
>
> Coefficients:
> (Intercept)           x1           x2
>         1.0          1.0          0.2
>
> Degrees of Freedom: 29 Total (i.e. Null);  27 Residual
> Null Deviance:        19.25
> Residual Deviance: -3.516e-16     AIC: 16.69
```

20.5 Weather data

```
  library(geodata)
# https://ouicodedata.com//posts/worldclim-with-r/

  prec_GBR <- worldclim_country("GBR", "prec", "data/WC")
  prec_SCO <- crop( prec_GBR, ext_SCO )

  plot( prec_SCO )
  plot( prec_SCO[[1]] )
```

```
  library(easyclimate)
# https://cran.r-project.org/web/packages/easyclimate/easyclimate.pdf
```

```
  r  <- rast(prec_SCO[[1]])
  xy <- xyFromCell( r, 1:(492*708) )

  prcp <- get_daily_climate(
    coords           = xy[100000:100001,],
    climatic_var     = "Prcp",
    period           = 2000:2018,
    output           = "df",
    version          = 4,
    check_connection = TRUE
  )
```

```
  library(QBMS)
# https://r4photobiology.info/galleries/online-weather-data.html

  prec_TC_SCO <- get_terraclimate(
    lat = c( 54.5, 58.6 ),
    lon = c( -7.7, -1.8 ),
    from = "2000-01-01",
    to   = "2018-12-31",
    clim_vars = "ppt",
    month_mask = NULL,
    offline = FALSE,
    data_path = "./data/"
  )
```

```
  library(pastclim)
  # https://evolecolgroup.github.io/pastclim/articles/a0_pastclim_overview.html
  # https://evolecolgroup.github.io/pastclim/pastclim_cheatsheet.pdf
  # https://cran.r-project.org/web/packages/pastclim/vignettes/a0_pastclim_overview.html

  set_data_path("data/pastclim")

  get_available_datasets()

  help("Beyer2020")
  help("WorldClim_2.1")
  help("CHELSA_2.1")

  get_resolution(dataset = "Example")
  get_resolution(dataset = "WorldClim_2.1_ACCESS-CM2_ssp245_10m")
  get_resolution(dataset = "CHELSA_2.1_MRI-ESM2-0_ssp585_0.5m")

  get_vars_for_dataset(dataset = "Example")
  get_vars_for_dataset(dataset = "Beyer2020")
  get_vars_for_dataset(dataset = "WorldClim_2.1_ACCESS-CM2_ssp245_10m" )
  get_vars_for_dataset(dataset = "CHELSA_2.1_MRI-ESM2-0_ssp585_0.5m" )

  get_time_ce_steps   (dataset = "WorldClim_2.1_ACCESS-CM2_ssp245_10m" )
  get_time_ce_steps   (dataset = "CHELSA_2.1_MRI-ESM2-0_ssp585_0.5m" )

  download_dataset(dataset = "Beyer2020", bio_variables = "bio12")
  download_dataset(dataset = "WorldClim_2.1_ACCESS-CM2_ssp245_10m",
```

```
                  bio_variables = "bio12", annual=TRUE)
download_dataset(dataset = "CHELSA_2.1_MRI-ESM2-0_ssp585_0.5m",
                  bio_variables = "bio12", annual=TRUE)

get_downloaded_datasets()

clim <- region_series( time_bp = list(min=-20000,max=0),
  bio_variables = "bio12", dataset = "Example", ext = ext_SCO )
clim <- region_series( time_ce = c(2030,2050,2070,2090),
  bio_variables = "bio12", dataset = "WorldClim_2.1_ACCESS-CM2_ssp245_10m",
  ext = ext_SCO )
clim <- region_series( time_ce = c(2025,2055,2075),
  bio_variables = "bio12", dataset = "CHELSA_2.1_MRI-ESM2-0_ssp585_0.5m",
  ext = ext_SCO )

plot(clim$bio12)
rclim <- rast(clim)
plot(rclim)
```

See also:

- https://rpubs.com/dbebber/r_geospatial
- https://oceanhealthindex.org/news/raster_to_terra/
- https://r4photobiology.info/galleries/online-weather-data.html
- https://pjbartlein.github.io/REarthSysSci/netcdf_terra.html

References

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2013. *Bayesian Data Analysis*. 3 edition. Boca Raton: Chapman and Hall/CRC.

He, Qiaoning, Weimin Ju, Shengpei Dai, Wei He, Lian Song, Songhan Wang, Xinchuan Li, and Guangxiong Mao. 2021. "Drought Risk of Global Terrestrial Gross Primary Productivity Over the Last 40 Years Detected by a Remote Sensing-Driven Process Model." *Journal of Geophysical Research: Biogeosciences* 126 (6). https://doi.org/10.1029/2020JG005944.

Hochrainer-Stigler, Stefan. 2019. "Large Scale Extreme Risk Assessment Using Copulas: An Application to Drought Events Under Climate Change for Austria." *Computational Management Science* 16: 651–69.

Jaynes, Edwin T. 2003. *Probability Theory: The Logic of Science*. Cambridge University Press.

Kuhnert, Matthias, Jagadeesh Yeluripati, Pete Smith, Holger Hoffmann, Marcel Van Oijen, Julie Constantin, Rene Dechow, et al. 2017. "Impact Analysis of Climate Data Aggregation at Different Spatial Scales on Simulated Net Primary Productivity for Croplands." *European Journal of Agronomy* 88: 41–52.

Lindley, D. V., and A. F. M. Smith. 1972. "Bayes Estimates for the Linear Model." *J. Roy. Stat. Soc.* 34: 1–41.

Liu, Changfeng, Weiping Chen, Ying Hou, and Lingchao Ma. 2020. "A New Risk Probability Calculation Method for Urban Ecological Risk Assessment." *Environmental Research Letters* 15 (2): 024016. https://doi.org/10.1088/1748-9326/ab6667.

Mee, Robert W., and D. B. Owen. 1983. "A Simple Approximation for Bivariate Normal Probabilities." *Journal of Quality Technology* 15 (2): 72–75. https://doi.org/10.1080/00224065.1983.11978848.

Murphy, Kevin P. 2007. "Conjugate Bayesian Analysis of the Gaussian Distribution." *Def* 1 ($2\sigma 2$): 16.

Nandintsetseg, Banzragch, Bazartseren Boldgiv, Jinfeng Chang, Philippe Ciais, Enkhbaatar Davaanyam, Altangerel Batbold, Tserenpurev Bat-Oyun, and Nils Chr. Stenseth. 2021. "Risk and Vulnerability of Mongolian Grasslands Under Climate Change." *Environmental Research Letters* 16 (3): 034035. https://doi.org/10.1088/1748-9326/abdb5b.

Nandintsetseg, Banzragch, Jinfeng Chang, Omer L. Sen, Christopher P. O. Reyer, Kaman Kong, Omer Yetemen, Philippe Ciais, and Jamts Davaadalai. 2024. "Future Drought Risk and Adaptation of Pastoralism in Eurasian Rangelands." *Npj Climate and Atmospheric Science* 7 (1): 1–14. https://doi.org/10.1038/s41612-024-00624-2.

Oliphant, Travis E. 2006. "A Bayesian Perspective on Estimating Mean, Variance, and Standard-Deviation from Data."

Ren, Jinyuan, Xiaomeng Guo, Siqin Tong, Yuhai Bao, Gang Bao, and Xiaojun Huang. 2023. "Risk Posed to Vegetation Net Primary Productivity by Drought on the Mongolian Plateau." *Journal of Geographical Sciences* 33 (11): 2175–92. https://doi.org/10.1007/s11442-023-2171-1.

Salcedo-Sanz, Sancho, Jorge Pérez-Aracil, Guido Ascenso, Javier Del Ser, David Casillas-Pérez, Christopher Kadow, Dušan Fister, et al. 2023. "Analysis, Characterization, Prediction, and Attribution of Extreme Atmospheric Events with Machine Learning and Deep Learning Techniques: A Review." *Theoretical and Applied Climatology*, August. https://doi.org/10.1007/s00704-023-04571-5.

Van Oijen, M. 2019. "Tools for Landscape Science: Theory, Models and Data." In *Current Trends in Landscape Research*, edited by L. Müller and F. Eulenstein, 219–31. Springer.

Van Oijen, Marcel. 2020. *Bayesian Compendium*. Springer International Publishing. https://doi.org/10.1007/978-3-030-55897-0.

———. 2024. *Bayesian Compendium*. 2nd ed. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-66085-6.

Van Oijen, Marcel, and Mark Brewer. 2022. *Probabilistic Risk Analysis and Bayesian Decision Theory*. SpringerBriefs in Statistics. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-16333-3.

Van Oijen, Marcel, and Miguel A. Zavala. 2019. "Probabilistic Drought Risk Analysis for Even-Aged Forests." In *Climate Extremes and Their Implications for Impact and Risk Assessment*, edited by Jana Sillmann, Sebastian Sippel, and Simone Russo, 159–76. Elsevier.

Van Oijen, M., J. Balkovic, C. Beer, D. R. Cameron, P. Ciais, W. Cramer, T. Kato, et al. 2014. "Impact of Droughts on the Carbon Cycle in European Vegetation: A Probabilistic Risk Analysis Using Six Vegetation Models." *Biogeosciences* 11 (22): 6357–75. https://doi.org/10.5194/bg-11-6357-2014.

Van Oijen, M., C. Beer, W. Cramer, A. Rammig, M. Reichstein, S. Rolinski, and J.-F. Soussana. 2013. "A Novel Probabilistic Risk Analysis to Determine the Vulnerability of Ecosystems to Extreme Climatic Events." *Environmental Research Letters* 8 (1): 015032. https://doi.org/10.1088/1748-9326/8/1/015032.

Wheeler, Lorien, Jessie Dotson, Michael Aftosmis, Ashley Coates, Grégoire Chomette, and Donovan Mathias. 2024. "Risk Assessment for Asteroid Impact Threat Scenarios." *Acta Astronautica* 216 (March): 468–87. https://doi.org/10.1016/j.actaastro.2023.12.049.

Zhou, Lei, Shaoqiang Wang, Yonggang Chi, and Junbang Wang. 2018. "Drought Impacts on Vegetation Indices and Productivity of Terrestrial Ecosystems in Southwestern China During 2001–2012." *Chinese Geographical Science* 28 (5): 784–96. https://doi.org/10.1007/s11769-018-0967-1.