Example 3.3

Disease mapping: from foundations to multidimensional modeling

Martinez-Beneito M.A. and Botella-Rocamora P.

This document reproduces the analysis made at Example 3.3 of the book: "Disease mapping: from foundations to multidimensional modeling" by Martinez-Beneito M.A. and Botella-Rocamora P., published by CRC press in 2019. You can watch the analysis made with full detail at this pdf document, or even execute it if you want with the material available at https://github.com/MigueBeneito/DMBook. Anyway, this pdf file should be enough for following most of the details of the analysis made for this example.

The statistical analysis below has been run in `R`, by additionally using the library `Rmarkdown`, so be sure that you have this software installed if you want to reproduce by yourself the content of this document. In that case we advise you to download first the annex material at https://github.com/MigueBeneito/DMBook, open with `Rstudio` the corresponding `.Rproj` file that you will find at the folder corresponding to this example and compile the corresponding `.Rmd` document. This will allow you to reproduce the whole statistical analysis below.

Libraries and data loading

```r
# Libraries loading
#-----------------
if (!require(R2WinBUGS)) {
    install.packages("R2WinBUGS")
    library(R2WinBUGS)
}
if (!require(pbugs)) {
    if (!require(devtools)) {
        install.packages("devtools")
        devtools::install_github("fisabio/pbugs")
    } else {
        install_github("fisabio/pbugs")
    }
}


# Data loading
#------------
load("../Data/OralCancerTimeTrends.RData")
```

`WinBUGS` call from `R`

```r
# WinBUGS syntax for the Bayesian logistic regression model
model.logist = function() {
    # Data likelihood
    for (i in 1:n) {
        O[i] ~ dbin(Prob[i], Pop[i])
        logit(Prob[i]) <- beta[1] + beta[2] * (year[i] - mean(year[]))
        rate[i] <- 1e+05 * Prob[i]
    }
    # Prior distributions
    beta[1] ~ dflat()
    beta[2] ~ dflat()

    # Probability of negative association (probabilities vs. years)
    P.beta2.lower.0 <- step(-beta[2])
}

# data preparation
year = 1987:2011

data = list(n = length(O), O = O, Pop = Pop, year = year)

# inits generating function
inits = function() {
    list(beta = rnorm(2, mean = 0, sd = 5))
}

# parameters to save
param = c("beta", "P.beta2.lower.0", "rate")


# WinBUGS call by means of bugs (R2WinBUGS library)
resul = bugs(data = data, inits = inits, parameters.to.save = param, n.iter = 3000,
    n.burnin = 300, model.file = model.logist, bugs.seed = 1)

# exploration of the structure of the bugs object
names(resul)
```

```
##  [1] "n.chains"        "n.iter"          "n.burnin"
##  [4] "n.thin"          "n.keep"          "n.sims"
##  [7] "sims.array"      "sims.list"       "sims.matrix"
## [10] "summary"         "mean"            "sd"
## [13] "median"          "root.short"      "long.short"
## [16] "dimension.short" "indexes.short"   "last.values"
## [19] "isDIC"           "DICbyR"          "pD"
## [22] "DIC"             "model.file"      "program"
```

```
# result summary
resul
```

```
## Inference for Bugs model at "C:/Users/Migue/AppData/Local/Temp/RtmpwZ5gVO/model7141ee63060.txt", fit
##  3 chains, each with 3000 iterations (first 300 discarded), n.thin = 8
##  n.sims = 1014 iterations saved
##                  mean  sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
## beta[1]          -9.5 0.0  -9.5  -9.5  -9.5  -9.5  -9.5    1  1000
## beta[2]           0.0 0.0   0.0   0.0   0.0   0.0   0.0    1   810
## P.beta2.lower.0   1.0 0.0   1.0   1.0   1.0   1.0   1.0    1     1
## rate[1]           9.1 0.3   8.5   8.9   9.0   9.3   9.7    1  1000
## rate[2]           8.9 0.3   8.4   8.7   8.9   9.1   9.5    1  1000
## rate[3]           8.8 0.2   8.3   8.6   8.8   8.9   9.3    1  1000
## rate[4]           8.6 0.2   8.2   8.5   8.6   8.8   9.1    1  1000
## rate[5]           8.5 0.2   8.1   8.4   8.5   8.6   8.9    1  1000
## rate[6]           8.4 0.2   8.0   8.2   8.4   8.5   8.7    1  1000
## rate[7]           8.2 0.2   7.9   8.1   8.2   8.3   8.6    1  1000
## rate[8]           8.1 0.2   7.8   8.0   8.1   8.2   8.4    1  1000
## rate[9]           8.0 0.1   7.7   7.9   8.0   8.1   8.2    1  1000
## rate[10]          7.8 0.1   7.6   7.8   7.8   7.9   8.1    1  1000
## rate[11]          7.7 0.1   7.5   7.6   7.7   7.8   8.0    1  1000
## rate[12]          7.6 0.1   7.3   7.5   7.6   7.7   7.8    1  1000
## rate[13]          7.5 0.1   7.2   7.4   7.5   7.6   7.7    1  1000
## rate[14]          7.4 0.1   7.1   7.3   7.4   7.4   7.6    1  1000
## rate[15]          7.2 0.1   7.0   7.1   7.2   7.3   7.5    1  1000
## rate[16]          7.1 0.1   6.8   7.0   7.1   7.2   7.4    1  1000
## rate[17]          7.0 0.2   6.7   6.9   7.0   7.1   7.3    1  1000
## rate[18]          6.9 0.2   6.6   6.8   6.9   7.0   7.3    1  1000
## rate[19]          6.8 0.2   6.4   6.7   6.8   6.9   7.2    1  1000
## rate[20]          6.7 0.2   6.3   6.6   6.7   6.8   7.1    1  1000
## rate[21]          6.6 0.2   6.2   6.4   6.6   6.7   7.0    1  1000
## deviance        173.0 1.9 171.1 171.7 172.4 173.7 178.6    1   540
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = Dbar-Dhat)
## pD = 1.9 and DIC = 174.9
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# detailed summary with 3 decimal digits
round(resul$summary, 3)
```

```
##                     mean    sd    2.5%     25%     50%     75%   97.5%
## beta[1]           -9.502 0.017  -9.535  -9.514  -9.502  -9.490  -9.469
## beta[2]           -0.016 0.003  -0.022  -0.018  -0.016  -0.014  -0.011
## P.beta2.lower.0    1.000 0.000   1.000   1.000   1.000   1.000   1.000
## rate[1]            9.064 0.296   8.533   8.855   9.044   9.264   9.665
## rate[2]            8.919 0.271   8.434   8.730   8.901   9.103   9.467
## rate[3]            8.776 0.247   8.329   8.609   8.762   8.941   9.270
## rate[4]            8.636 0.224   8.221   8.483   8.623   8.785   9.082
## rate[5]            8.498 0.203   8.122   8.356   8.490   8.633   8.907
## rate[6]            8.362 0.184   8.010   8.240   8.359   8.485   8.739
## rate[7]            8.229 0.168   7.898   8.118   8.226   8.335   8.575
## rate[8]            8.097 0.153   7.798   8.001   8.100   8.196   8.405
## rate[9]            7.968 0.141   7.696   7.877   7.970   8.062   8.249
## rate[10]           7.841 0.133   7.585   7.751   7.842   7.930   8.099
## rate[11]           7.716 0.128   7.464   7.625   7.714   7.803   7.961
## rate[12]           7.593 0.126   7.344   7.500   7.591   7.684   7.836
## rate[13]           7.472 0.128   7.227   7.381   7.473   7.563   7.717
## rate[14]           7.353 0.133   7.094   7.261   7.352   7.446   7.604
## rate[15]           7.236 0.140   6.976   7.141   7.237   7.331   7.500
## rate[16]           7.121 0.148   6.847   7.023   7.118   7.223   7.422
## rate[17]           7.008 0.158   6.710   6.903   6.999   7.114   7.338
## rate[18]           6.897 0.169   6.571   6.782   6.892   7.008   7.255
## rate[19]           6.787 0.180   6.440   6.669   6.780   6.902   7.164
## rate[20]           6.679 0.192   6.306   6.556   6.674   6.801   7.082
## rate[21]           6.573 0.204   6.176   6.446   6.570   6.699   7.001
## deviance         173.012 1.949 171.100 171.700 172.400 173.700 178.567
##                   Rhat n.eff
## beta[1]          1.001  1000
## beta[2]         1.002   810
## P.beta2.lower.0 1.000     1
## rate[1]         1.001  1000
## rate[2]         1.001  1000
## rate[3]         1.001  1000
## rate[4]         1.001  1000
## rate[5]         1.001  1000
## rate[6]         1.001  1000
## rate[7]         1.001  1000
## rate[8]         1.001  1000
## rate[9]         1.001  1000
## rate[10]        1.001  1000
## rate[11]        1.001  1000
## rate[12]        1.001  1000
## rate[13]        1.001  1000
## rate[14]        1.001  1000
## rate[15]        1.001  1000
## rate[16]        1.001  1000
## rate[17]        1.001  1000
## rate[18]        1.001  1000
## rate[19]        1.001  1000
## rate[20]        1.001  1000
## rate[21]        1.001  1000
## deviance        1.003   540
```

Computing time comparisons for bugs and pbugs

```
time.3000.bugs = system.time(bugs(data = data, inits = inits, parameters.to.save = param,
    n.iter = 3000, n.burnin = 300, model.file = model.logist, bugs.seed = 1))

time.30000.bugs = system.time(bugs(data = data, inits = inits, parameters.to.save = param,
    n.iter = 30000, n.burnin = 300, model.file = model.logist, bugs.seed = 1))

time.300000.bugs = system.time(bugs(data = data, inits = inits, parameters.to.save = param,
    n.iter = 3e+05, n.burnin = 300, model.file = model.logist, bugs.seed = 1))

time.3000000.bugs = system.time(bugs(data = data, inits = inits, parameters.to.save = param,
    n.iter = 3e+06, n.burnin = 300, model.file = model.logist, bugs.seed = 1))

time.3000.pbugs = system.time(pbugs(data = data, inits = inits, parameters.to.save = param,
    n.iter = 3000, n.burnin = 300, model.file = model.logist, bugs.seed = 1))

time.30000.pbugs = system.time(pbugs(data = data, inits = inits, parameters.to.save = param,
    n.iter = 30000, n.burnin = 300, model.file = model.logist, bugs.seed = 1))

time.300000.pbugs = system.time(pbugs(data = data, inits = inits, parameters.to.save = param,
    n.iter = 3e+05, n.burnin = 300, model.file = model.logist, bugs.seed = 1))

time.3000000.pbugs = system.time(pbugs(data = data, inits = inits, parameters.to.save = param,
    n.iter = 3e+06, n.burnin = 300, model.file = model.logist, bugs.seed = 1))

# Computing times bugs
```

```r
c(time.3000.bugs[3], time.30000.bugs[3], time.300000.bugs[3], time.3000000.bugs[3])
```

```
## elapsed elapsed elapsed elapsed
##    3.63    7.90   60.12  610.84
```

```r
# Computing times pbugs
c(time.3000.pbugs[3], time.30000.pbugs[3], time.300000.pbugs[3], time.3000000.pbugs[3])
```

```
## elapsed elapsed elapsed elapsed
##    4.04    5.52   27.20  245.28
```