

Large Scale Simulations on a Linux Cluster

David A. James (Modeling & Simulation)
David Ohlssen (Clinical Information Sciences)

Novartis

March 10, 2008



Outline

- ▶ Study characteristics and the need for a futility rule
- ▶ Large scale simulations on a high-performance computing (HPC) environment
- ▶ Lessons learned

Background

- ▶ Study characteristics and end points:
 - ▶ Three arms: Placebo, Low and High doses
 - ▶ Event type 1 counts (T1) at N months
 - ▶ Event type 2 counts (T2) at N months
- ▶ Need for a futility rule for the low-dose arm

A Bayesian methodology was chosen among various alternative, but significant computational challenges were identified

Our approach

- ▶ Develop a Bayesian model for bivariate over-dispersed Poisson counts
- ▶ Define a futility rule in terms of Poisson rates
- ▶ Conduct simulations to investigate the operating characteristics of the futility rule
- ▶ Use distributed computing to address computational challenges

Rules for futility

At the time of the interim analysis, stop the trial if a “good” outcome is “unlikely”

- ▶ A good outcome could be

$$\{T1/Placebo < \tau\} \text{ and } \{T2/Placebo < \tau\}$$

say, for $\tau = 0.50$

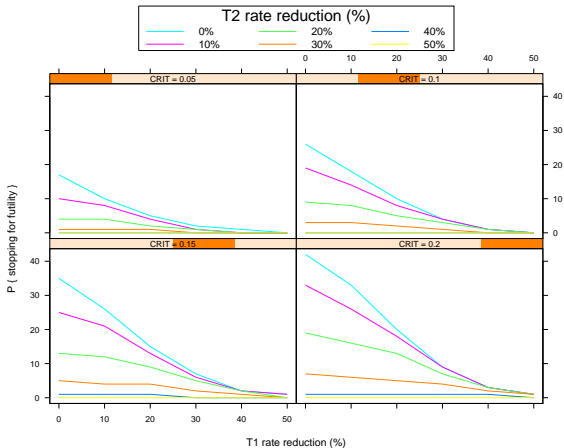
- ▶ An unlikely event could be

$$P\{T1/Placebo < \tau\} < CRIT$$

$$P\{T2/Placebo < \tau\} < CRIT$$

say, for $CRIT = 0.10$

Characteristics of a futility rule



A Bayesian model

Let Y_{ijk} denote the count for the i 'th patient, j 'th outcome (T1 at baseline, T1 at N months, T2), and k 'th treatment (placebo, treatment k):

$$Y_{ijk} \sim \text{Poisson}(\lambda_{ijk})$$
$$\log(\lambda_{ijk}) = \mu_{ijk} + \eta_{ij}$$

where the random effects $(\eta_{i1}, \eta_{i2}, \eta_{i3}) \sim N_3(0, \Sigma)$.
The treatment-to-Placebo ratios of rates are

$$R_1 = \mu_{22}/\mu_{21} \quad \text{for T1}$$
$$R_2 = \mu_{32}/\mu_{31} \quad \text{for T2}$$

Data is generated in R, estimation is MCMC-based in WinBUGS

A Bayesian model (WinBUGS)

```
for (i in 1:3){
  for (j in 1:2){
    mu1[i,j] ~ dnorm(0,0.0001)
    mu[i,j] <- exp(mu1[i,j])
  }
}
for (j in 1:N) {
  RE[j,1:3] ~ dnorm(zero[1:3],prec[1:3,1:3])
  log.lambda[j,1] <- mu1[1,TRT[j]] + RE[j,1]
  lambda[j,1] <- exp(log.lambda[j,1])
  Y1[j] ~ dpois(lambda[j,1])
  ...
}
for (j in 1:RatiosL) {
  RatiosMuT1[j] <- step( Ratios[j] - mu[2,2]/mu[2,1] )
  RatiosMuT2[j] <- step( Ratios[j] - mu[3,2]/mu[3,1] )
}
```

Simulation scenarios and timings

Estimate the probability of stopping for futility:

$$P\{T1/Placebo < \tau\} < CRIT \text{ and } P\{T2/Placebo < \tau\} < CRIT$$

- ▶ At interim analysis (N months) with n subjects per arm
- ▶ 36 “true” T1/Placebo, T2/Placebo ratios (1.0, 0.9, ..., 0.5)
- ▶ τ values of 1.0, 0.9, ..., 0.5

Each scenario simulates 2500 trials for a total of 90000 trials

Each simulated trial takes about 1.5 to 2 minutes

Entire simulation study requires about 3 to 4 CPU months

Grid computing: Divide and conquer

- ▶ Large number of computers
- ▶ Connected by a fast dedicated network
- ▶ Software for job submission, scheduling, security, etc.
- ▶ Full cluster with about 200 dual-CPU computing nodes (Linux/Irix)
- ▶ Nodes partitioned according to “queues”

Simulation requirements and challenges

- ▶ Very easy to parallelized
- ▶ WinBUGS, R, R2WinBUGS
- ▶ WinBUGS high-performance computing limitations:
 - ▶ Windows-only application
 - ▶ Very rudimentary scripting facilities
 - ▶ Undocumented single-instance limitations
 - ▶ GUI always displayed

Solutions:

- ▶ The cluster provides WINE (windows emulator)
- ▶ Setup multiple WINE installations on each compute node
- ▶ Connect WinBUGS' GUI to a virtual windowing manager (Xvfm)

Simulations

- ▶ Split over 50 compute nodes (2 jobs/node)
- ▶ Main computations done over a weekend
- ▶ Additional computations were required to post-process a large number of output files
- ▶ The entire study was compressed to about 4 weeks

Additional futility rules were requested based on zero-count probabilities

Grid distributed computing allowed us to respond in a timely manner

See handout for full computing details

Lessons learned

- ▶ “Tools matter”
- ▶ Yet a lot of our tools are inadequate for large scale computing
- ▶ Some of us need to educate ourselves on modern high-performance computing
- ▶ Some application may not be amenable for large scale computing (e.g., SAS appears to be extremely expensive to deploy on clusters)

Acknowledgments

- ▶ Chris Harwell
- ▶ José Pinheiro
- ▶ Jouni Kerman
- ▶ Heinz Schmidli