Bayesian Biopharmaceutical Applications using PROC MCMC and PROC BGLIMM

Fang Chen
SAS Institute Inc.
fangk.chen@sas.com

KOL Lecture, March 22, 2019

Acknowledgments go to my colleague, Amy Shi, who is the developer of PROC BGLIMM.

Outline

Bayesian Analysis in SAS/STAT$^{®}$Software

SAS/STAT software includes procedures mainly for statistical analysis and visualization. We take a number of routes in providing Bayesian capabilities in the software:

Bayesian Analysis in SAS/STAT®Software

SAS/STAT software includes procedures mainly for statistical analysis and visualization. We take a number of routes in providing Bayesian capabilities in the software:

- In model-specific procedures that provide both frequentists and Bayesian solutions:
  - PROC PHREG, PROC GENMOD, PROC LIFEREG, PROC FMM, etc.

Bayesian Analysis in SAS/STAT®Software

SAS/STAT software includes procedures mainly for statistical analysis and visualization. We take a number of routes in providing Bayesian capabilities in the software:

- In model-specific procedures that provide both frequentists and Bayesian solutions:
  - PROC PHREG, PROC GENMOD, PROC LIFEREG, PROC FMM, etc.
  - The BAYES statement
  - A set of frequently used prior distributions (noninformative, Jeffreys')

Bayesian Analysis in SAS/STAT®Software

SAS/STAT software includes procedures mainly for statistical analysis and visualization. We take a number of routes in providing Bayesian capabilities in the software:

- In model-specific procedures that provide both frequentists and Bayesian solutions:
  - PROC PHREG, PROC GENMOD, PROC LIFEREG, PROC FMM, etc.
  - The BAYES statement
  - A set of frequently used prior distributions (noninformative, Jeffreys')
- General simulation procedure
  - PROC MCMC

Bayesian Analysis in SAS/STAT®Software

SAS/STAT software includes procedures mainly for statistical analysis and visualization. We take a number of routes in providing Bayesian capabilities in the software:

- In model-specific procedures that provide both frequentists and Bayesian solutions:
  - PROC PHREG, PROC GENMOD, PROC LIFEREG, PROC FMM, etc.
  - The BAYES statement
  - A set of frequently used prior distributions (noninformative, Jeffreys')
- General simulation procedure
  - PROC MCMC
- Fully Bayesian procedures for a class of models:
  - PROC BGLIMM for generalized linear mixed models (GLMMs)
    - New in SAS/STAT 15.1 (9.4 TS1M6, the 6th maintenance release)

SAS 9.4

Release dates and versions of SAS 9.4:

Version	Release Date	STAT name
9.4	July 2013	STAT 12.3
9.4m1	December 2013	STAT 13.1
9.4m2	August 2014	STAT 13.2
9.4m3	July 2015	STAT 14.1
9.4m4	November 2016	STAT 14.2
9.4m5	September 2017	STAT 14.3
9.4m6	November 2018	STAT 15.1

Version Information

To find out your version:

```
proc product_status;
   run;
```

which produces something like:

```
...
For SAS/STAT ...
   Custom version information: 15.1
...
```

Outline

1. Software Overview
   - PROC MCMC
   - PROC BGLIMM
2. Applications
   - Power Prior: Kociba Case Study
   - Evaluation of a Basket Clinical Trial Design
   - Internal Release Limits

PROC MCMC is a General Sampling Procedure

- Your program represents how you would write the statistical model - it is similar to PROC NLMIXED

PROC MCMC is a General Sampling Procedure

- Your program represents how you would write the statistical model - it is similar to PROC NLMIXED
- You must specify all aspects of a statistical model: parameters, prior distributions, random effects, how random effects enter the model, likelihood function, and so on.

PROC MCMC is a General Sampling Procedure

- Your program represents how you would write the statistical model - it is similar to PROC NLMIXED
- You must specify all aspects of a statistical model: parameters, prior distributions, random effects, how random effects enter the model, likelihood function, and so on.
- Statements simplify the specification of your statistical model, provide coding convenience, and make the program readable.

PROC MCMC is a General Sampling Procedure

- Your program represents how you would write the statistical model - it is similar to PROC NLMIXED
- You must specify all aspects of a statistical model: parameters, prior distributions, random effects, how random effects enter the model, likelihood function, and so on.
- Statements simplify the specification of your statistical model, provide coding convenience, and make the program readable.
- Use DATA step programming statements in more complex scenarios where the standard distributions or functions are inadequate.

Generality of PROC MCMC

The MCMC procedure fits

- single-level or multilevel (hierarchical) models
- linear or nonlinear models, such as regression, survival, ordinal multinomial
- multivariate analysis, latent variable models, state space models, PK models
- missing data problems
- . . .

Generality of PROC MCMC

The MCMC procedure fits

- single-level or multilevel (hierarchical) models
- linear or nonlinear models, such as regression, survival, ordinal multinomial
- multivariate analysis, latent variable models, state space models, PK models
- missing data problems
- ...

In addition, PROC MCMC supports

- SAS DATA step programming language
- user-defined sampling algorithms, functions, distributions.
- prediction

Input Data Set

PROC MCMC takes in a SAS data set, which is a rectangular structure that has **variables** (columns) and **records** (rows).

```
  Name         Height     Weight

  Alfred        69.0       112.5
  Alice         56.5        84.0
  Barbara       65.3        98.0
  Carol         62.8       102.5
  Henry         63.5         .
  James         57.3        83.0
  Jane          59.8        84.5
  Janet         62.5       112.5
...
```

Missing values are coded as dots.

Syntax Reflects the Statistical Model

$$\begin{aligned}
\text{weight}_i &\sim \text{N}(\mu_i, \text{var} = \sigma^2), \quad i = 1, \ldots, n \\
\mu_i &= \beta_0 + \beta_1 \cdot \text{height}_i \\
\beta_0, \beta_1 &\sim \text{N}(0, \text{var} = 100) \\
\sigma^2 &\sim \text{iGamma}(\text{shape} = 2, \text{scale} = 2)
\end{aligned}$$

This is similar to what all general-purpose Bayesian software packages (BUGS, NIMBLE, Stan, etc) strive for.

```
proc mcmc data=class;
   parms b0 b1 s2;
   prior b0 b1 ~ normal(0, var=100);
   prior s2 ~ igamma(shape=2, scale=2);
   mu = b0 + b1 * height;
   model weight ~ normal(mu, var=s2);
   run;
```

Procedure Offers Modeling Flexibility

$$\begin{aligned}
\text{weight}_i &\sim \text{t}(\mu, \text{sd} = \sigma, \text{df} = 3) \quad i = 1, \ldots, n \\
\mu_i &= \beta_0 + \beta_1 \cdot \text{height}_i \\
\beta_0, \beta_1 &\sim \text{N}(0, \text{var} = 100) \\
\sigma &\sim \text{uniform}(0, 25)
\end{aligned}$$

```
proc mcmc data=class seed=1 nbi=5000 nmc=10000 outpost=regOut;
   parms b0 b1 sig;
   prior b0 b1 ~ normal(0, var=100);
   prior sig ~ uniform(0, 25);
   mu = b0 + b1 * height;
   model weight ~ t(mu, sd=sig, df=3);
   run;
```

DATA Step Language Offers More Flexibility

$$
\begin{aligned}
\text{weight}_i &\sim \text{N}(\mu_i, \text{var} = \sigma^2), \quad i = 1, \ldots, n \\
\mu_i &= \begin{cases} \alpha + \beta_1 \cdot \text{height}_i & \text{if height}_i < \theta \\ \alpha + \beta_2 \cdot \text{height}_i & \text{if height}_i \geq \theta \end{cases}
\end{aligned}
$$

```
proc mcmc data=class;
   parms b0 b1 b2 s2 theta;
   prior b: ~ normal(0, var=100);
   prior s2 ~ igamma(shape=2, scale=2);
   prior theta ~ uniform(0, 200);
   if height < theta then
      mu = b0 + b1 * height;
   else
      mu = b0 + b2 * height;
   model weight ~ normal(mu, var=s2);
   run;
```

Compare to BUGS

In WinBUGS, you see the entire data set and work with the matrix (do indexing explicitly, for example).

```
height[] weight[]
  69.0       112.5
  56.5        84.0
  65.3        98.0
...
  66.5       112.0
END
```

```
model
{
   for(i in 1:19) {
      mu[i] = b0 + b1 * height[i]
      weight[i] ~ dnorm(mu[i], tau)
   }
   b0 ~ dnorm(0, 0.1)
   b1 ~ dnorm(0, 0.1)
   tau ~ gamma(0.1, 0.1)
}
```

Compare to BUGS

In PROC MCMC, you work with variables (think one record at a time).

```
height     weight
  69.0      112.5
  56.5       84.0
  65.3       98.0
...
  66.5      112.0
```

```
prior b0 b1 ~ normal(0, prec=0.1);
prior tau ~ gamma(0.1, iscale=0.1);
mu = b0 + b1 * height;
model weight ~ dnorm(mu, prec=tau);
```

The variables `height` and `weight` are filled in with data set values as PROC MCMC processes the input data set.

The variable `mu` is calculated on the fly.

Looping Over the Data Set

At each iteration, PROC MCMC steps through the data set, record by record:

- resolves symbols and processes programming statements
- accumulates the loglikelihood

```
Obs    Height    Weight
  1      69.0     112.5
  2      56.5      84.0
  3      65.3      98.0
  ...
 19      66.5     112.0
```

```
proc mcmc data=input;
   prior;
   progm stmt;
   model ;
   run;
```

at the top of the data set

$$\log \pi(\theta|\boldsymbol{y}) = \log(f(y_1|\theta))$$

Looping Over the Data Set

At each iteration, PROC MCMC steps through the data set, record by record:

- resolves symbols and processes programming statements
- accumulates the loglikelihood

```
Obs     Height     Weight
  1       69.0      112.5
  2       56.5       84.0
  3       65.3       98.0
  ...
 19       66.5      112.0
```

```
proc mcmc data=input;
   prior;
  {progm stmt;
  {model ;
   run;
```

stepping through the data set

$$\log \pi(\theta|\mathbf{y}) = \log \pi(\theta|\mathbf{y}) + \log(f(y_2|\theta))$$

Looping Over the Data Set

At each iteration, PROC MCMC steps through the data set, record by record:

- resolves symbols and processes programming statements
- accumulates the loglikelihood

```
Obs    Height    Weight
  1      69.0     112.5
  2      56.5      84.0
  3      65.3      98.0
...
 19      66.5     112.0
```

```
proc mcmc data=input;
   prior;
   progm stmt;
   model  ;
   run;
```

stepping through the data set

$$\log \pi(\theta|\mathbf{y}) = \log \pi(\theta|\mathbf{y}) + \log(f(y_3|\theta))$$

Looping Over the Data Set

At each iteration, PROC MCMC steps through the data set, record by record:

- resolves symbols and processes programming statements
- accumulates the loglikelihood

Obs	Height	Weight
1	69.0	112.5
2	56.5	84.0
3	65.3	98.0
...		
19	66.5	112.0

```
proc mcmc data=input;
   prior;
   progm stmt;
   model;
   run;
```

at the last observation, the prior is included

$$\log \pi(\theta|\boldsymbol{y}) = \log(\pi(\theta)) + \textstyle\sum_{i=1}^{n} \log(f(y_i|\theta))$$

Sampling Algorithm Hierarchy

	Continuous Parameters	**Discrete Parameters**
Users First	User-Defined Samplers	
When Applicable	Conjugate Direct	Conjugate Direct Inverse CDF
All Others	**RWM** RWM-t HMC NUTS slice	**Discrete RWM** Geometric RWM

Algorithms are multithreaded for fast performance.

Programming Order Matters

PROC MCMC relies on SAS programming language, hence the order matters.

```
mu = beta0 + beta1 * x;
model y ~ normal(mu, var=s2);
```

is different from

```
model y ~ normal(mu, var=s2);
mu = beta0 + beta1 * x;
```

This means that you can reuse the same symbol in a program:

```
mu = beta0 + beta1 * x;
model y ~ normal(mu, var=s2);
mu = alpha0 + alpha2 * y;
model z ~ normal(mu, var=sz2);
```

or

```
if lambda ne 0 then
   z = (y**lambda - 1) / lambda;
else
   z = log(y);
model z ~ normal(mu, var=s2);
```

Minimize Redundant Computations

Most runtime is spent on executing programming statements over and over again, at each iteration for every observation.

Minimize Redundant Computations

Most runtime is spent on executing programming statements over and over again, at each iteration for every observation.

- constant terms, ignored after initialization.

```
BEGINCNST;
w = 3;
ENDCNST;
```

Minimize Redundant Computations

Most runtime is spent on executing programming statements over and over again, at each iteration for every observation.

- constant terms, ignored after initialization.

```
BEGINCNST;
w = 3;
ENDCNST;
```

- redundant computations not carried out for every record:

```
BEGINNODATA;
tau = 1/sigma2;
ENDNODATA;
```

Features Relevant to Pharma Applications

- Truncation and Censoring
- Non-standard Distributions
- Multivariate and Categorical Distributions
- Hierarchical Models
- Missing Data
- Posterior Prediction

You Can Specify Truncated Distributions

- Normalized distribution with bounds.

You Can Specify Truncated Distributions

- Normalized distribution with bounds.
- Univariate distributions support (optional) LOWER= and UPPER= bounds.

```
prior alpha ~ n(0, sd=10, lower=0);
prior b ~ expon(scale=100, lower=100, upper=2000);
```

You Can Specify Truncated Distributions

- Normalized distribution with bounds.
- Univariate distributions support (optional) LOWER= and UPPER= bounds.

```
prior alpha ~ n(0, sd=10, lower=0);
prior b ~ expon(scale=100, lower=100, upper=2000);
```

- The bounds can be (functions of) random variables:

```
prior beta  ~ n(0, sd=10, lower=alpha);
prior gamma ~ n(0, sd=10, lower=alpha * beta);
```

Or Work With Censored Data

- Unobserved (missing) data that we know lie within some bounds but can't observe them

Or Work With Censored Data

- Unobserved (missing) data that we know lie within some bounds but can't observe them
- Univariate distribution support CLOWER= and CUPPER= censoring option:

```
model y ~ normal(mu, sd=1, clower=cl, cupper=cr);
```

  Missing *y* values become parameters and sampled accordingly. The censoring indicators, *cl* and *cr*, can be missing (left-, right-, interval censoring).

Or Work With Censored Data

- Unobserved (missing) data that we know lie within some bounds but can't observe them
- Univariate distribution support CLOWER= and CUPPER= censoring option:

  ```
  model y ~ normal(mu, sd=1, clower=cl, cupper=cr);
  ```

  Missing *y* values become parameters and sampled accordingly. The censoring indicators, *cl* and *cr*, can be missing (left-, right-, interval censoring).
- You can also use the marginal approach to model censored data (see PROC MCMC documentation)

Non-Standard Distribution

You can specify non-standard distribution in PROC MCMC.

Non-Standard Distribution

You can specify non-standard distribution in PROC MCMC. Suppose that a distribution has the following density specification:

$$\begin{aligned} & \pi(p) \propto p^{-1}(1-p)^{-1} \\ \Rightarrow \quad & \log(\pi(p)) = -(log(p) + log(1-p)) \end{aligned}$$

Non-Standard Distribution

You can specify non-standard distribution in PROC MCMC. Suppose that a distribution has the following density specification:

$$\begin{aligned} & \pi(p) \propto p^{-1}(1-p)^{-1} \\ \Rightarrow \quad & \log(\pi(p)) = -(log(p) + log(1-p)) \end{aligned}$$

You use the GENERAL function in PROC MCMC to specify the prior distribution:

```
proc mcmc data=trials seed=17 nmc=20000 outpost=HalBin;
   parm p 0.5;
   lprior = -(log(p) + log(1-p));
   prior p ~ general(lprior, lower=0, upper=1);
   model event ~ binomial(n,p);
   run;
```

Direct Simulation

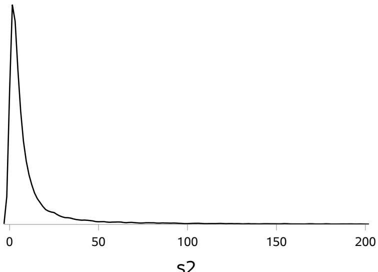
You can use PROC MCMC to draw samples from a joint distribution with marginal and conditional specifications (without data):

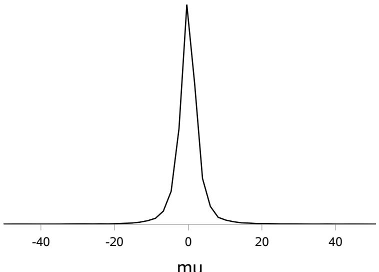
```
data a; run;   /* make an empty data set */
proc mcmc data=a seed=79467 nmc=20000 outpost=two_out;
   parm s2 mu;
   prior s2 ~ cauchy(0, 5, lower=0); ! $\sigma^2 \sim \pi(\sigma^2)$
   prior mu ~ n(0, var=s2);  ! $\mu \sim \pi(\mu|\sigma^2)$
   model general(0);
run;
```

Direct Simulation

You can use PROC MCMC to draw samples from a joint distribution with marginal and conditional specifications (without data):

```
data a; run;   /* make an empty data set */
proc mcmc data=a seed=79467 nmc=20000 outpost=two_out;
   parm s2 mu;
   prior s2 ~ cauchy(0, 5, lower=0); ! σ² ∼ π(σ²)
   prior mu ~ n(0, var=s2);  ! μ ∼ π(μ|σ²)
   model general(0);
run;
```





Multivariate or Categorical Distributions

PROC MCMC also supports the following distributions:

- `dirich`: Dirichlet
- `iwish`: inverse-Wishart
- `mvn`: multivariate normal
- `multinom`: multinomial
- `table`: categorical

Model Response Variables (Likelihood Function)

MODEL *dependent-variable-list* $\sim$ *distribution*;

specifies the likelihood function. The dependent variables can be

- data set variables

```
model y ~ normal(alpha, var=1);
```

Model Response Variables (Likelihood Function)

MODEL *dependent-variable-list* $\sim$ *distribution*;

specifies the likelihood function. The dependent variables can be

- data set variables

```
model y ~ normal(alpha, var=1);
```

- functions of data set variables

```
w = log(y);
model w ~ normal(alpha, var=1);
```

Model Response Variables (Likelihood Function)

MODEL *dependent-variable-list* $\sim$ *distribution*;

specifies the likelihood function. The dependent variables can be

- data set variables

```
model y ~ normal(alpha, var=1);
```

- functions of data set variables

```
w = log(y);
model w ~ normal(alpha, var=1);
```

You can specify multiple MODEL statements, one for a response variable:

```
model height ~ normal(mu, var=s2_h);
model weight ~ normal(b0 + b1 * height, var=s2_w);
```

Use RANDOM Statement for Random Effects

Specify a random-effects model is fairly straightforward:

```
proc mcmc data=schools nmc=5000 seed=2157;
   parm mu s2;
   prior mu ~ n(0, sd=1000);    ! $\mu \sim N(0, 1000)$
   parm s2g ~ normal(0, sd=5, lower=0); ! $\sigma^2 \sim half - normal$
   random theta ~ n(mu, var=s2g) subject=ID; ! $\theta_i \sim N(\mu, \sigma^2)$
   model y ~ normal(theta, sd=s2y); ! $y_i \sim N(\theta_i, \sigma_y^2)$
run;
```

You can specify complex multilevel random-effects models:

- multiple random effects
- nested or non-nested hierarchical models
- random-effects with non-normal prior
- nonlinear models
- various latent class models
- autoregressive or spatially-distributed random effects

You can specify complex multilevel random-effects models:

- multiple random effects
- nested or non-nested hierarchical models
- random-effects with non-normal prior
- nonlinear models
- various latent class models
- autoregressive or spatially-distributed random effects

For generalized linear mixed-effects models, PROC BGLIMM offers an easier alternative.

Missing Data

- Missing values are represented using a period (.) in SAS data sets.

Missing Data

- Missing values are represented using a period (.) in SAS data sets.
- Missing response values are modeled by default:

  ```
  model y ~ n(mu, var=1);
  ```

  Each missing y becomes a parameter and is sampled. This is equivalent to Missing at Random (MAR).

Missing Data

- Missing values are represented using a period (.) in SAS data sets.
- Missing response values are modeled by default:

```
model y ~ n(mu, var=1);
```

Each missing y becomes a parameter and is sampled. This is equivalent to Missing at Random (MAR).

- PROC MCMC supports partial missing:

```
array data[3] y1 y2 y3;
model data ~ mvn(mu, Sigma);
```

or

```
llike = f(y1, y2, y3);
model y1 y2 y3 ~ general(llike);
```

You can have partial missing in any of the response variables.

Various Missing Data Scenarios

- You can carry out a complete-case analysis

```
proc mcmc ... missing=CC;
```

PROC MCMC discards all records with missing values. This is equivalent to Missing Completely at Random (MCAR).

Various Missing Data Scenarios

- You can carry out a complete-case analysis

  ```
  proc mcmc ... missing=CC;
  ```

  PROC MCMC discards all records with missing values. This is equivalent to Missing Completely at Random (MCAR).
- You can also model Missing Not at Random (MNAR) data
  - selection model approach
  - pattern mixture approach

Various Missing Data Scenarios

- You can carry out a complete-case analysis

  ```
  proc mcmc ... missing=CC;
  ```

  PROC MCMC discards all records with missing values. This is equivalent to Missing Completely at Random (MCAR).
- You can also model Missing Not at Random (MNAR) data
  - selection model approach
  - pattern mixture approach
- Or an all-case analysis

  ```
  proc mcmc ... missing=AC;
  ```

  This gives you the control on how to handle the missing values directly.

Posterior Prediction

Sample $\mathbf{y}_{\text{pred}}$ from

$$\pi(\mathbf{y}_{\text{pred}}|\mathbf{y}) = \int \pi(\mathbf{y}_{\text{pred}}|\boldsymbol{\theta}, \mathbf{y})\pi(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$$

There are three ways to do that in PROC MCMC:

- In-procedure approach
- Missing data approach
- Use the `PREDDIST` statement

Outline

Model-Specific Bayesian Procedures

SAS/STAT has two such procedures:

- PROC BCHOICE: Bayesian discrete choice models
- PROC BGLIMM: Bayesian generalized linear mixed models

Both procedures use model-specific algorithms to draw samples from the joint posterior distribution.

Model-Specific Bayesian Procedures

SAS/STAT has two such procedures:

- PROC BCHOICE: Bayesian discrete choice models
- PROC BGLIMM: Bayesian generalized linear mixed models

Both procedures use model-specific algorithms to draw samples from the joint posterior distribution.

PROC BGLIMM was release in SAS/STAT 15.1.

Mixed Models

A mixed model (random-effects) is a model that contains fixed and random effects.

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon} \\ \boldsymbol{\gamma} &\sim N(\mathbf{0}, \mathbf{G}) \\ \boldsymbol{\epsilon} &\sim N(\mathbf{0}, \mathbf{R}) \end{aligned}$$

the parameter $\boldsymbol{\beta}$ is considered fixed and $\boldsymbol{\gamma}$ (random effects) are random.

Estimation (frequentist) is achieved by maximizing the marginal likelihood of the fixed-effects parameter while integrating out the random effects.

Mixed Modeling Procedures in SAS

- `PROC MIXED` fits linear mixed-effects models:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\gamma + \epsilon; \quad \gamma \sim N(\mathbf{0}, \mathbf{G}) \quad \epsilon \sim \mathbf{N}(\mathbf{0}, \mathbf{R})$$

- `PROC GLIMMIX` fits generalized linear mixed-effects models:

$$E[\mathbf{Y}|\gamma] = g^{-1}(\boldsymbol{\eta}) = g^{-1}(\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\gamma)$$

  where $\boldsymbol{\eta}$ is the linear predictor and $g^{-1}(\cdot)$ is the inverse link function
- `PROC NLMIXED` includes nonlinear capabilities:
  - $\mathbf{Y}$ relates to $\boldsymbol{\eta}$ via nonlinear transformation
  - the random effects enters the model nonlinearly

Bayesian Approach

The Bayesian paradigm ($\pi(\boldsymbol{\theta}|\mathbf{Y}) \propto \pi(\boldsymbol{\theta}) \cdot L(\mathbf{Y};\boldsymbol{\theta})$) fits the same class of models but treats every parameter, fixed effect or random effect, as random:

$$\begin{array}{rcll} \mathbf{Y} & = & \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon} & \text{same likelihood function} \\ \boldsymbol{\beta} & \sim & \pi(\boldsymbol{\beta}) & \\ \boldsymbol{\gamma} & \sim & N(\mathbf{0}, \mathbf{G}) & \text{same prior on RE} \\ \mathbf{G} & \sim & \pi(\mathbf{G}) & \\ \mathbf{R} & \sim & \pi(\mathbf{R}) & \end{array}$$

The Bayesian approach estimates the joint posterior of $\pi(\boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{R}, \mathbf{G}|\mathbf{Y}, \mathbf{X}, \mathbf{Z})$ and infers from the marginal posterior $\pi(\boldsymbol{\beta}|\mathbf{Y}, \mathbf{X}, \mathbf{Z})$.

Mixed Modeling Procedures

	Likelihood Function	RE Dist	Linear Predictor	Hierarchy
`MIXED`	Normal	Normal	$\mathbf{X}\beta + \mathbf{Z}\gamma$	Nested & Non-Nested
`GLIMMIX`	GLM	Normal	$\mathbf{X}\beta + \mathbf{Z}\gamma$	Nested & Non-Nested
`NLMIXED`	General	Normal	General	Nested

Nested	students within classes.
Non-Nested	students taking lessons from different teachers.

PROC MCMC

	Likelihood Function	RE Dist	Linear Predictor	Hierarchy
MIXED	Normal	Normal	$\mathbf{X}\beta + \mathbf{Z}\gamma$	Nested & Non-Nested
GLIMMIX	GLM	Normal	$\mathbf{X}\beta + \mathbf{Z}\gamma$	Nested & Non-Nested
NLMIXED	General	Normal	General	Nested
MCMC	General	General	General	Nested & Non-Nested

PROC MCMC offers flexibility.

PROC BGLIMM

	Likelihood Function	RE Dist	Linear Predictor	Hierarchy
`MIXED`	Normal	Normal	$\mathbf{X}\beta + \mathbf{Z}\gamma$	Nested & Non-Nested
`GLIMMIX`	GLM	Normal	$\mathbf{X}\beta + \mathbf{Z}\gamma$	Nested & Non-Nested
`NLMIXED`	General	Normal	General	Nested
`BGLIMM`	GLM	Normal	$\mathbf{X}\beta + \mathbf{Z}\gamma$	Nested & Non-Nested

PROC BGLIMM fits a smaller class of models but with much ease.

PROC BGLIMM Shares Similar Syntax to PROC MIXED/PROC GLIMMIX

If you are somewhat familiar with PROC MIXED and PROC GLIMMIX, transition to PROC BGLIMM is not difficult.

PROC BGLIMM Shares Similar Syntax to PROC MIXED/PROC GLIMMIX

If you are somewhat familiar with PROC MIXED and PROC GLIMMIX, transition to PROC BGLIMM is not difficult.
You have the usual suspects in

- MODEL Statement: specifies the response (y), fixed effects (x), likelihood function (dist=), and link function (link=)

PROC BGLIMM Shares Similar Syntax to PROC MIXED/PROC GLIMMIX

If you are somewhat familiar with PROC MIXED and PROC GLIMMIX, transition to PROC BGLIMM is not difficult.
You have the usual suspects in

- MODEL Statement: specifies the response (y), fixed effects (x), likelihood function (dist=), and link function (link=)
- RANDOM Statement: specifies the random effects and the G-side variance/covariance structure

PROC BGLIMM Shares Similar Syntax to PROC MIXED/PROC GLIMMIX

If you are somewhat familiar with PROC MIXED and PROC GLIMMIX, transition to PROC BGLIMM is not difficult.
You have the usual suspects in

- MODEL Statement: specifies the response (y), fixed effects (x), likelihood function (dist=), and link function (link=)
- RANDOM Statement: specifies the random effects and the G-side variance/covariance structure
- REPEATED Statement: specifies the R-side residual var/cov structure

PROC BGLIMM Shares Similar Syntax to PROC MIXED/PROC GLIMMIX

If you are somewhat familiar with PROC MIXED and PROC GLIMMIX, transition to PROC BGLIMM is not difficult.
You have the usual suspects in

- MODEL Statement: specifies the response (y), fixed effects (x), likelihood function (dist=), and link function (link=)
- RANDOM Statement: specifies the random effects and the G-side variance/covariance structure
- REPEATED Statement: specifies the R-side residual var/cov structure
- CLASS Statement (not supported in PROC MCMC)

PROC BGLIMM Shares Similar Syntax to PROC MIXED/PROC GLIMMIX

If you are somewhat familiar with PROC MIXED and PROC GLIMMIX, transition to PROC BGLIMM is not difficult.
You have the usual suspects in

- MODEL Statement: specifies the response (y), fixed effects (x), likelihood function (dist=), and link function (link=)
- RANDOM Statement: specifies the random effects and the G-side variance/covariance structure
- REPEATED Statement: specifies the R-side residual var/cov structure
- CLASS Statement (not supported in PROC MCMC)
- ESTIMATE Statement

Procedure Details: Syntax

PROC BGLIMM Statement '

This statement includes these commonly used options:

DATA=	names the input data set
DIC	computes the deviance information criterion
NBI=	specifies the number of burn-in iterations
NMC=	specifies the number of iterations, excluding the burn-ins
OUTPOST=	names the output data set to contain posterior samples
SEED=	specifies the random seed for simulation
STATS=	controls posterior statistics

Procedure Details: Syntax

MODEL ***response*** **=** ***fixed-effects*** **< / model-options>;**

This statement specifies the response and fixed-effects parameters. You can also use this statement to specify the response distribution via the DIST= option and to specify the link function $g(\cdot)$ via the LINK= option.

Some other useful options follow:

- NOINT excludes the fixed-effects intercept from the model.
- OFFSET= specifies the offset variable.
- COEFFPRIOR= specifies the prior of the fixed-effects coefficients.
- SCALEPRIOR= specifies the prior of the scale parameter.

Simple Linear Regression with Class Variable

```
proc bglimm data=Sashelp.Class nmc=10000 thin=2
   seed=436792 outpost=Classout;
   class sex;
   model Weight = Height Age Sex / cprior=normal(var=1e6);
run;
```

The CPRIOR= option specifies the prior distribution for the coefficient prior (β's).

Simple Linear Regression with Class Variable

```
proc bglimm data=Sashelp.Class nmc=10000 thin=2
   seed=436792 outpost=Classout;
   class sex;
   model Weight = Height Age Sex / cprior=normal(var=1e6);
run;
```

The CPRIOR= option specifies the prior distribution for the coefficient prior (β's).

There are default priors for all parameters.

Procedure Details: Syntax

Built-In Resposne Distributions:

DIST= Option Value	Distribution Function
BINARY	Binary
BINOMIAL	Binary or binomial
EXPONENTIAL | EXPO	Exponential
GAMMA | GAM	Gamma
GEOMETRIC | GEOM	Geometric
INVGAUSS | IG	Inverse Gaussian
NEGBINOMIAL | NEGBIN | NB	Negative binomial
NORMAL | GAUSSIAN | GAUSS	Normal
POISSON | POI	Poisson

Procedure Details: Syntax

Default and Commonly Used Link Functions:

Distributions	**Default Link Function**	**Other Commonly Used Link Functions**
BINARY	Logit	Probit, comp log-log, log-log
BINOMIAL	Logit	Probit, comp log-log, log-log
EXPONENTIAL	Log	Reciprocal
GAMMA	Log	Reciprocal
GEOMETRIC	Log	
INVGAUSS	Reciprocal square	
NEGBINOMIAL	Log	
NORMAL	Identity	Log
POISSON	Log	

Procedure Details: Syntax

RANDOM ***random-effects*** < / **options**>;

Defines the **Z** design matrix for the random effects, γ, and the covariance structure of the **G** matrix.

- SUBJECT= option to identify the subjects for the random effects and thus to set up the blocks of **G**. A set of random effects is estimated for each subject level.
- GROUP= option to identify groups by which to vary the covariance parameters; each new level of the grouping effect produces a new set of covariance parameters
- TYPE= option to define the covariance structure of **G**.
- You can specify multiple RANDOM statements.

Logistic Random-Effects Model

Example program:

```
proc bglimm data=MultiCenter nmc=10000 seed=976352;
   class Center Group;
   model SideEffect/N = Group / noint;
   random int / subject = Center;
run;
```

Logistic Random-Effects Model

Example program:

```
proc bglimm data=MultiCenter nmc=10000 seed=976352;
   class Center Group;
   model SideEffect/N = Group / noint;
   random int / subject = Center;
run;
```

Recall that the mixed model setup in BGLIMM follows the standard convention:

$$E[Y|\boldsymbol{\beta},\boldsymbol{\gamma}] = g^{-1}(\boldsymbol{\eta}) = g^{-1}(\mathbf{X}\boldsymbol{\beta}+\mathbf{Z}\boldsymbol{\gamma})$$

Logistic Random-Effects Model

Example program:

```
proc bglimm data=MultiCenter nmc=10000 seed=976352;
   class Center Group;
   model SideEffect/N = Group / noint;
   random int / subject = Center;
run;
```

Recall that the mixed model setup in BGLIMM follows the standard convention:

$$E[Y|\boldsymbol{\beta},\boldsymbol{\gamma}] = g^{-1}(\boldsymbol{\eta}) = g^{-1}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma})$$

The random effects are assumed normally distributed:

$$\gamma_i \sim \; N(0, \mathbf{G_i})$$

Multiple RANDOM Statements

You can add multiple random effects to the model:

```
proc bglimm data=a;
   class Analyst Run Plate  conc;
   model log_assay = Analyst conc ;
   random int / subject=run(analyst)
     covprior=uniform(lower=0, upper=2) s;
   random int / subject=plate(run*analyst)
    covprior=halfnormal(var=4) s;
run;
```

Multiple RANDOM Statements

You can add multiple random effects to the model:

```
proc bglimm data=a;
   class Analyst Run Plate  conc;
   model log_assay = Analyst conc ;
   random int / subject=run(analyst)
     covprior=uniform(lower=0, upper=2) s;
   random int / subject=plate(run*analyst)
    covprior=halfnormal(var=4) s;
run;
```

The random effects can be nested or nonnested.

Multiple RANDOM Statements

You can add multiple random effects to the model:

```
proc bglimm data=a;
   class Analyst Run Plate  conc;
   model log_assay = Analyst conc ;
   random int / subject=run(analyst)
     covprior=uniform(lower=0, upper=2) s;
   random int / subject=plate(run*analyst)
    covprior=halfnormal(var=4) s;
run;
```

The random effects can be nested or nonnested.

The COVPRIOR= option provides choices on the prior distribution of the *G*-sided variance/covariance parameter.

Procedure Details: Syntax

Types of covariance structures:

Structure	**Description**
ANTE(1)	Antedependence
AR(1)	Autoregressive(1)
ARH(1)	Heterogeneous AR(1)
ARMA(1,1)	ARMA(1,1)
CS	Compound symmetry
CSH	Heterogeneous compound symmetry
FA(1)	Factor analytic
HF	Huynh-Feldt
TOEP	Toeplitz
TOEP(q)	Banded Toeplitz
TOEPH	Banded heterogeneous Toeplitz
UN	Unstructured
UN(q)	Banded unstructured
VC	Variance components

Procedure Details: Syntax

REPEATED *repeated-effect* < / options>;

Secifies the **R** matrix in the model.

- A *repeated-effect* is required to define the proper location of the repeated responses. The levels of the *repeated-effect* must be different for each observation within a subject.
- SUBJECT= option to set up the blocks of **R**.
- GROUP= option to identify groups by which to vary the covariance parameters; each new level of the grouping effect produces a new set of covariance parameters.
- TYPE= option to define the covariance structure.
- You can specify only one REPEATED statement.

Repeated Measures Model

The REPEATED statement models balanced/unbalanced repeated measurements data:

```
proc bglimm data=Fev nmc=10000 seed=44672057
      outpost=FevOut;
   class Drug Patient Hour;
   model FEV = BaseVal Drug Hour;
   random int / subject=Patient;
   repeated Hour / subject=Patient(Drug) type=un;
run;
```

Repeated Measures Model

The REPEATED statement models balanced/unbalanced repeated measurements data:

```
proc bglimm data=Fev nmc=10000 seed=44672057
      outpost=FevOut;
   class Drug Patient Hour;
   model FEV = BaseVal Drug Hour;
   random int / subject=Patient;
   repeated Hour / subject=Patient(Drug) type=un;
run;
```

Only the MVN likelihood is supported in this release.

Model Heterogeneity

The GROUP= option models different covariance types for different groups:

```
   proc bglimm data=pr seed=475193 outpost=pr_out;
      class Person Gender Time;
      model Distance = Age|Gender;
      repeated Time / type=un subject=Person group=Gender;
   run;
```

Misc Features

PROC BGLIMM models missing response variable by default.

Misc Features

PROC BGLIMM models missing response variable by default.

- This corresponds to Missing at Random (MAR).

Misc Features

PROC BGLIMM models missing response variable by default.

- This corresponds to Missing at Random (MAR).

You can use PROC BGLIMM for prediction via the missing data approach. Much in the same way as PROC GLIMMIX does it.

Misc Features

PROC BGLIMM models missing response variable by default.

- This corresponds to Missing at Random (MAR).

You can use PROC BGLIMM for prediction via the missing data approach. Much in the same way as PROC GLIMMIX does it.

The procedure supports a suite of prior distributions for β, G and R parameters, in addition to many different types of covariance structures (TYPE=).

Misc Features

PROC BGLIMM models missing response variable by default.

- This corresponds to Missing at Random (MAR).

You can use PROC BGLIMM for prediction via the missing data approach. Much in the same way as PROC GLIMMIX does it.

The procedure supports a suite of prior distributions for β, G and R parameters, in addition to many different types of covariance structures (TYPE=).

The procedure uses model-specific sampling algorithms (more efficient than PROC MCMC), and they are threaded for performance.

Outline

A Case Study on the Benchmark Approach in Toxicology

- The benchmark approach is a useful tool in toxicology.

A Case Study on the Benchmark Approach in Toxicology

- The benchmark approach is a useful tool in toxicology.
- The benchmark dose (BMD) is defined as the dose of an environmental toxicant that corresponds to a prescribed change in response compared with the background response level.

A Case Study on the Benchmark Approach in Toxicology

- The benchmark approach is a useful tool in toxicology.
- The benchmark dose (BMD) is defined as the dose of an environmental toxicant that corresponds to a prescribed change in response compared with the background response level.
- The toxicological data comprises n binomial responses $\boldsymbol{y} = (y_1, \ldots, y_n)$ with $y_i \sim b(n_i, p_i)$, where n_i is the number of animals tested at dose level x_i and p_i is the probability that an animal gives an adverse response at dose level x_i,

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)}, \quad i = 1, \ldots, n.$$

The Two Benchmark Studies in Toxicology

- The Kociba study (Kociba et al. 1978) is a lifetime feeding study of both female and male Sprague Dawley rats, with 50 rats tested in each group at doses of 0, 1, 10, and 100 ng/kg/day. Inferences derived from the Kociba study have been widely used as the basis for risk assessments for 2,3,7,8-tetrachlorodibenzodioxin (TCDD).

The Two Benchmark Studies in Toxicology

- The Kociba study (Kociba et al. 1978) is a lifetime feeding study of both female and male Sprague Dawley rats, with 50 rats tested in each group at doses of 0, 1, 10, and 100 ng/kg/day. Inferences derived from the Kociba study have been widely used as the basis for risk assessments for 2,3,7,8-tetrachlorodibenzodioxin (TCDD).
- The National Toxicology Program (NTP) study (National Toxicology Program 1982) is a study in which groups of 50 male rats, 50 female rats, and 50 male mice received TCDD as a suspension in 9:1 corn oil-acetone by gavage twice each week to achieve doses of 0, 10, 50, or 500 ng/kg/week for two years.

The Two Benchmark Studies in Toxicology

- The Kociba study (Kociba et al. 1978) is a lifetime feeding study of both female and male Sprague Dawley rats, with 50 rats tested in each group at doses of 0, 1, 10, and 100 ng/kg/day. Inferences derived from the Kociba study have been widely used as the basis for risk assessments for 2,3,7,8-tetrachlorodibenzodioxin (TCDD).
- The National Toxicology Program (NTP) study (National Toxicology Program 1982) is a study in which groups of 50 male rats, 50 female rats, and 50 male mice received TCDD as a suspension in 9:1 corn oil-acetone by gavage twice each week to achieve doses of 0, 10, 50, or 500 ng/kg/week for two years.
- In this analysis, we treat the Kociba study as the historical data and the NTP study the current data.

Benchmark Data Summary and Parameter Estimates

Study	TCDD(ng/kg/day) and Response				Estimates	
Kociba	Control (or 0)	1	10	100	β_0 (SD)	β_1 (SD)
	9/86	3/50	18/50	34/48	-1.785 (0.210)	0.028 (0.004)
NTP	Control (or 0)	1.4	7.1	71	β_0 (SD)	β_1 (SD)
	5/75	1/49	3/50	12/49	-3.030 (0.366)	0.026 (0.007)

Datasets

```
data KOCIBA;
   input y n dose;
datalines;
 9 86   0
 3 50   1
18 50  10
34 48 100
;
```

```
data NTP;
   input y n dose;
datalines;
 5 75   0
 1 49 1.4
 3 50 7.1
12 49  71
;
```

y	response
n	number of patients
dose	dosage

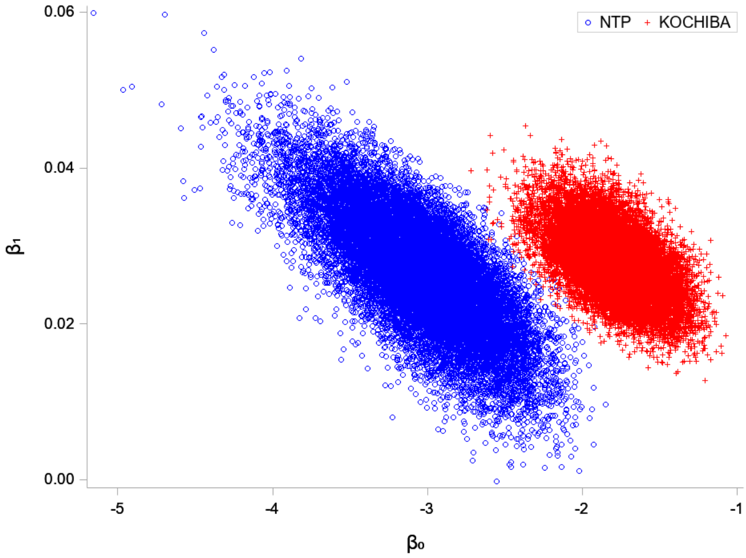
Logistic Regression with Flat Prior

```
proc mcmc data=kociba nmc=50000 seed=70273
   propcov=quanew outpost=kociba_flat;
   parm b0 0 b1 0;
   prior b: ~ general(0);
   p = logistic(b0 + b1 * dose);
   model y ~ binomial(n, p);
run;
```

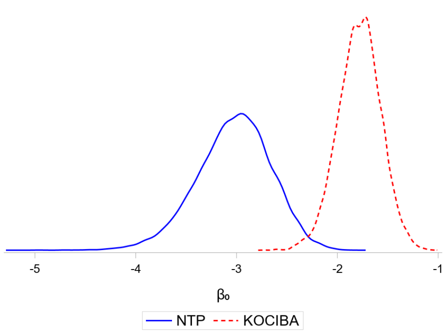
general(0) : flat prior on β_0 and β_1

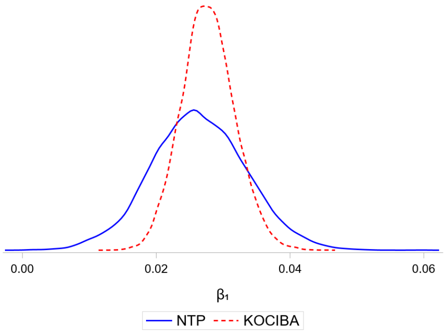
logistic : $p = \frac{\exp(\mu)}{1+\exp(\mu)}$

Joint Posterior Distributions from Two Separate Analysis

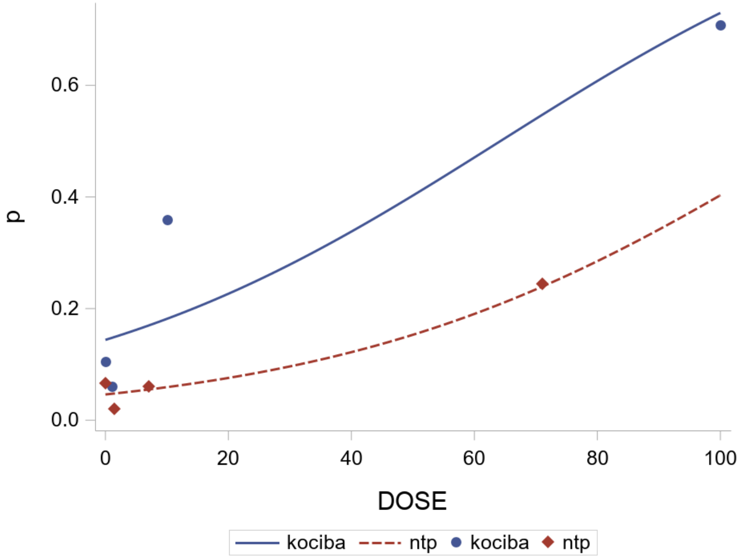


Marginal Posterior Densities of β_0 and β_1





Prediction Curves from the Noninformative Analysis



Power Prior using PROC MCMC

There are two ways to fit a power prior using PROC MCMC:

Power Prior using PROC MCMC

There are two ways to fit a power prior using PROC MCMC:

- Combined Approach
  - Form a larger data set and put a weight (a_0) on each observation

Power Prior using PROC MCMC

There are two ways to fit a power prior using PROC MCMC:

- Combined Approach
  - Form a larger data set and put a weight (a_0) on each observation
- Conventional Approach
  - Use the historical data to construct the power prior
  - Use the current data for the (binomial) likelihood function

Power Prior using PROC MCMC

There are two ways to fit a power prior using PROC MCMC:

- Combined Approach
  - Form a larger data set and put a weight (a_0) on each observation
- Conventional Approach
  - Use the historical data to construct the power prior
  - Use the current data for the (binomial) likelihood function

Each has its pros and cons.

Combined Approach

First recognize that the posterior distribution can be rewritten as:

$$p(\theta|D^*, a_0) \propto \prod_{i=1}^{n+n_0} f_i(y_i|\theta, x_i) \cdot \pi_0(\theta)$$

$$\text{where } f_i = \begin{cases} f(y_i|\theta, x_i) & \text{for each } i \text{ in the current data set} \\ f(y_{0,i}|\theta, x_{0,i})^{a_0} & \text{for each } i \text{ in the historical data set} \end{cases}$$

Combined Approach

First recognize that the posterior distribution can be rewritten as:

$$p(\theta|D^*, a_0) \propto \prod_{i=1}^{n+n_0} f_i(y_i|\theta, x_i) \cdot \pi_0(\theta)$$

$$\text{where } f_i = \begin{cases} f(y_i|\theta, x_i) & \text{for each } i \text{ in the current data set} \\ f(y_{0,i}|\theta, x_{0,i})^{a_0} & \text{for each } i \text{ in the historical data set} \end{cases}$$

You can create a combined data set and assign separate likelihood functions to different observations.

Combine Data Sets

You first combine both data sets:

```
data combined;
   format group $8.;
   set kociba(in=i) ntp;
   if i then group = "pilot";
   else group = "current";
   run;
```

Combine Data Sets

You first combine both data sets:

```
data combined;
   format group $8.;
   set kociba(in=i) ntp;
   if i then group = "pilot";
   else group = "current";
   run;
```

⇒

```
  y   n   dose   group
  9  86    0.0  pilot
  3  50    1.0  pilot
 18  50   10.0  pilot
 34  48  100.0  pilot
  5  75    0.0  current
  1  49    1.4  current
  3  50    7.1  current
 12  49   71.0  current
```

Binomial Model: Power Prior

For each observation in the new `combined` data set, the likelihood function is either:

- a binomial (if `group == 'current'`) or
- a weighted binomial (if `group == 'pilot'`)

Binomial Model: Power Prior

For each observation in the new `combined` data set, the likelihood function is either:

- a binomial (if `group == 'current'`) or
- a weighted binomial (if `group == 'pilot'`)

```
%let a0=0.3;
proc mcmc data=combined nmc=50000 seed=70273
   propcov=quanew outpost=ntp_power;
   parm b0 0 b1 0;
   prior b: ~ general(0);
   p = logistic(b0 + b1 * dose);
   llike = logpdf("binomial", y, p, n);
   if group eq "pilot" then
      llike = &a0 * llike;
   model y ~ general(llike);
   run;
```

Alternatively, you can put the weight a_0 in the `combined` data set:

```
data combined;                               y   n   dose    a0
   set kociba(in=i) ntp;                     9  86    0.0   0.3
   if i then a0 = 0.3;                       3  50    1.0   0.3
   else a0 = 1;                             ...
                                             5  75    0.0   1.0
                                             1  49    1.4   1.0
                                            ...
```

Alternatively, you can put the weight a_0 in the combined data set:

```
data combined;                      y   n  dose    a0
   set kociba(in=i) ntp;            9  86   0.0   0.3
   if i then a0 = 0.3;              3  50   1.0   0.3
   else a0 = 1;                   ...
                                    5  75   0.0   1.0
                                    1  49   1.4   1.0
                                  ...

proc mcmc data=combined ...;
   parm b0 0 b1 0;
   prior b: ~ general(0);
   p = logistic(b0 + b1 * dose);
   llike = a0 * logpdf("binomial", y, p, n);
   model y ~ general(llike);
run;
```

This produces the same posterior estimates.

Notes on Combined Approach

The combined approach in fitting power prior is intuitive and easy to implement;

Notes on Combined Approach

The combined approach in fitting power prior is intuitive and easy to implement;

The setup is generic to many model specifications, as long as the conditional independence assumption (e.g. in the likelihood function) holds.

Notes on Combined Approach

The combined approach in fitting power prior is intuitive and easy to implement;

The setup is generic to many model specifications, as long as the conditional independence assumption (e.g. in the likelihood function) holds.

There are some issues with this approach:

- DIC calculation, which should only depend on D, not D_0, cannot be correctly calculated within the procedure. Post-simulation calculation (use DATA step for example) can be tedious.

Notes on Combined Approach

The combined approach in fitting power prior is intuitive and easy to implement;

The setup is generic to many model specifications, as long as the conditional independence assumption (e.g. in the likelihood function) holds.

There are some issues with this approach:

- DIC calculation, which should only depend on D, not D_0, cannot be correctly calculated within the procedure. Post-simulation calculation (use DATA step for example) can be tedious.
- Cannot be extended to normalized power prior due to an integral calculation

Notes on Combined Approach

The combined approach in fitting power prior is intuitive and easy to implement;

The setup is generic to many model specifications, as long as the conditional independence assumption (e.g. in the likelihood function) holds.

There are some issues with this approach:

- DIC calculation, which should only depend on D, not D_0, cannot be correctly calculated within the procedure. Post-simulation calculation (use DATA step for example) can be tedious.
- Cannot be extended to normalized power prior due to an integral calculation

Will discuss these issues in later slides.

Conventional Approach in Fitting Power Prior

This approach specifies the power prior in its original form $\pi(\boldsymbol{\theta}|D_0, a_0) \propto L(\boldsymbol{\theta}|D_0)^{a_0}\pi_0(\boldsymbol{\theta})$, which depends on the pilot (`KOCIBA`) data set.

Conventional Approach in Fitting Power Prior

This approach specifies the power prior in its original form $\pi(\boldsymbol{\theta}|D_0, a_0) \propto L(\boldsymbol{\theta}|D_0)^{a_0}\pi_0(\boldsymbol{\theta})$, which depends on the pilot (`KOCIBA`) data set.

- Use `read_array` function to store the `KOCIBA` data set in an array
- Use `DO-loop` to compute the power prior
- Use the `general` function to specify the non-standard prior distribution

```
%let a0=0.3;
proc mcmc data=ntp ...;              ! use the current data set
   array pdata[1] / nosymbols;       ! array to store the pilot data set
   begincnst;
   rc = read_array("kociba", pdata); !  save kociba data in pdata
   nobs = dim(pdata, 1);
   endcnst;
```

```
%let a0=0.3;
proc mcmc data=ntp ...;             ! use the current data set
   array pdata[1] / nosymbols;      ! array to store the pilot data set
   begincnst;
   rc = read_array("kociba", pdata); !  save kociba data in pdata
   nobs = dim(pdata, 1);
   endcnst;

   parm b0 0 b1 0;
   beginprior;
   lp = 0;
   do j = 1 to nobs;                    ! loop through the pilot data
      p = logistic(b0 + b1 * pdata[j,3]);
      lp = lp+logpdf("binomial", pdata[j,1],p,pdata[j,2]); ! log(L(θ; D0))
      end;
   lp = &a0 * lp;                  ! a0 · log(L(θ; D0))
   prior b0 b1 ~ general(lp);
   endprior;
```

```
%let a0=0.3;
proc mcmc data=ntp ...;              ! use the current data set
   array pdata[1] / nosymbols;       ! array to store the pilot data set
   begincnst;
   rc = read_array("kociba", pdata); !  save kociba data in pdata
   nobs = dim(pdata, 1);
   endcnst;

   parm b0 0 b1 0;
   beginprior;
   lp = 0;
   do j = 1 to nobs;                     ! loop through the pilot data
      p = logistic(b0 + b1 * pdata[j,3]);
      lp = lp+logpdf("binomial", pdata[j,1],p,pdata[j,2]); ! log(L(θ; D_0))
      end;
   lp = &a0 * lp;                  ! a_0 · log(L(θ; D_0))
   prior b0 b1 ~ general(lp);
   endprior;

   p = logistic(b0 + b1 * dose);
   model y ~ binomial(n, p);
   run;
```

Notes on Conventional Approach

This approach is requires more coding:

- The objective function needs to be coded at two places:
    - once in the MODEL statement (`NTP`), the looping of observations is implicit
    - once in the prior construction (`KOCIBA`), the looping of observations is explicit

Notes on Conventional Approach

This approach is requires more coding:

- The objective function needs to be coded at two places:
    - once in the MODEL statement (`NTP`), the looping of observations is implicit
    - once in the prior construction (`KOCIBA`), the looping of observations is explicit
- More susceptible to coding errors

Notes on Conventional Approach

This approach is requires more coding:

- The objective function needs to be coded at two places:
  - once in the MODEL statement (`NTP`), the looping of observations is implicit
  - once in the prior construction (`KOCIBA`), the looping of observations is explicit
- More susceptible to coding errors

But this approach makes extensions easier.

Notes on Conventional Approach

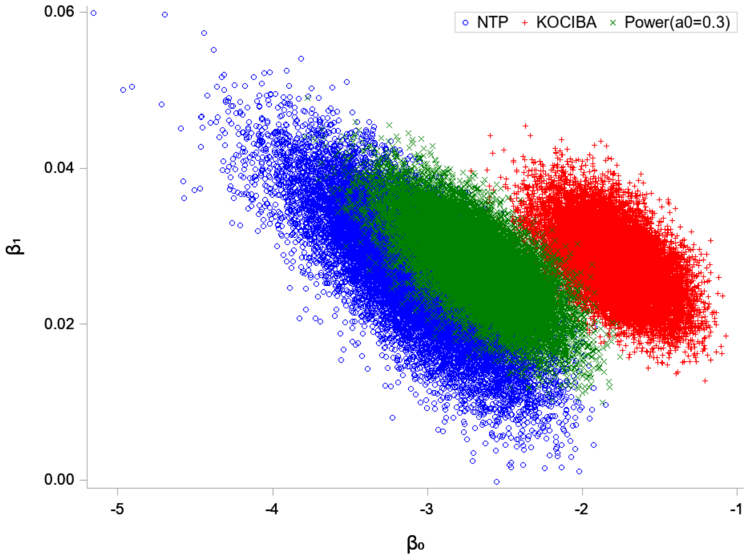
This approach is requires more coding:

- The objective function needs to be coded at two places:
  - once in the MODEL statement (`NTP`), the looping of observations is implicit
  - once in the prior construction (`KOCIBA`), the looping of observations is explicit
- More susceptible to coding errors

But this approach makes extensions easier.

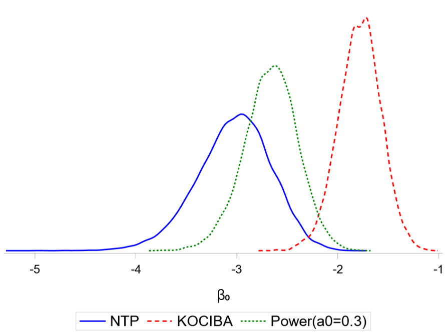
Use either approaches, depending on what you want to do.

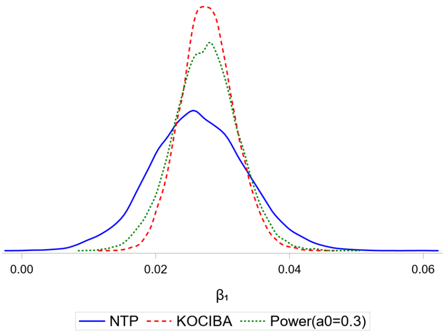
Power Prior with $a_0 = 0.3$



Compare to page 81

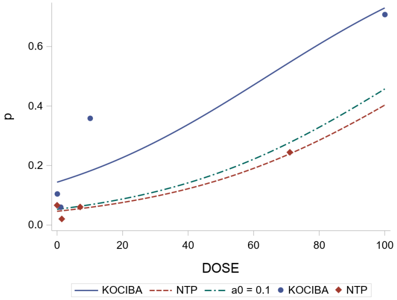
Marginal Posterior Comparisons



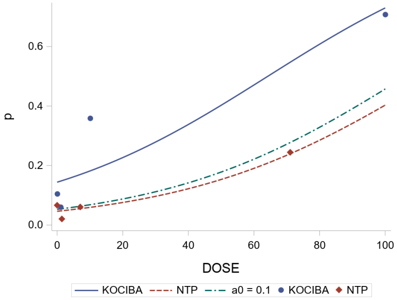


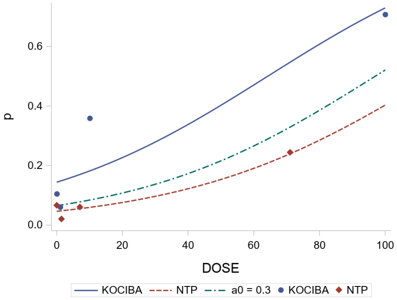
Compare to page 82

Fitted Curve Comparisons

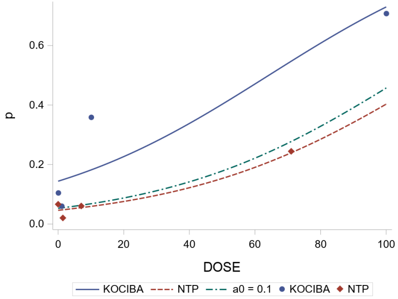


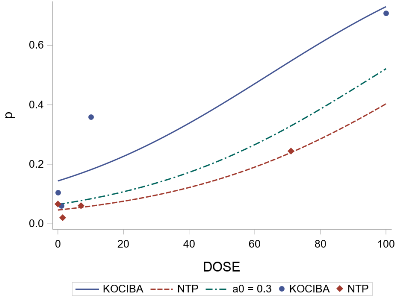
Fitted Curve Comparisons

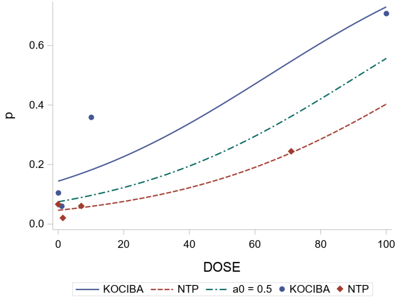




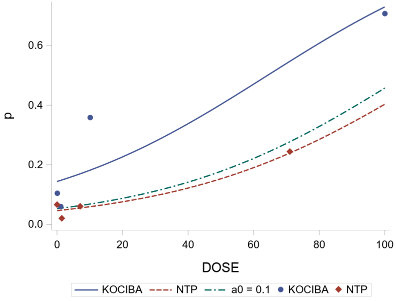
Fitted Curve Comparisons

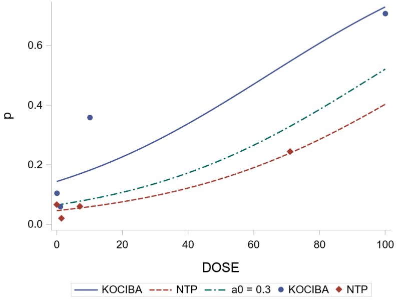


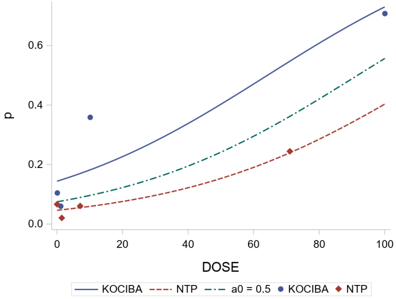


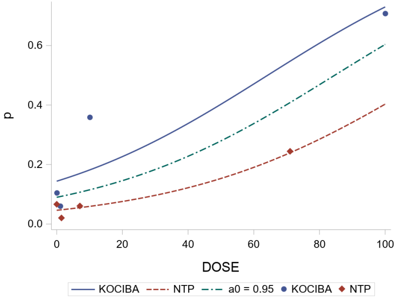


Fitted Curve Comparisons









An Immediate Question

How to choose an "optimal" value of a_0?

An Immediate Question

How to choose an "optimal" value of a_0?

- Model comparison:
  - Deviance Information Criterion (DIC)
  - Penalized Likelihood-type Criterion (PLC)
  - Marginal Likelihood Criterion (MLC)
  - Logarithm of the Pseudo-Marginal Likelihood Criterion (LPML)

An Immediate Question

How to choose an "optimal" value of a_0?

- Model comparison:
  - Deviance Information Criterion (DIC)
  - Penalized Likelihood-type Criterion (PLC)
  - Marginal Likelihood Criterion (MLC)
  - Logarithm of the Pseudo-Marginal Likelihood Criterion (LPML)
- Treat a_0 as a parameter and let the data inform:
  - Normalized power prior

An Immediate Question

How to choose an "optimal" value of a_0?

- Model comparison:
  - Deviance Information Criterion (DIC)
  - Penalized Likelihood-type Criterion (PLC)
  - Marginal Likelihood Criterion (MLC)
  - Logarithm of the Pseudo-Marginal Likelihood Criterion (LPML)
- Treat a_0 as a parameter and let the data inform:
  - Normalized power prior

Here we cover DIC and normalized power prior.

Deviance Information Criterion

- DIC (Spiegelhalter et al., 2002, *JRSSB*, 64:583) is a Bayesian alternative to AIC and BIC, a model assessment and selection tool.
- The criterion can be applied to non-nested models and models that have non-iid data.
- A smaller DIC indicates a better fit to the data.

Deviance Information Criterion (DIC)

$$\mathrm{DIC} = \overline{D(\boldsymbol{\theta})} + p_D = D(\bar{\boldsymbol{\theta}}) + 2p_D$$

where

- $D(\boldsymbol{\theta}) = 2\left(\log(f(\mathbf{y})) - \log(p(\mathbf{y}|\boldsymbol{\theta}))\right)$ is the deviance where
  - $p(\mathbf{y}|\boldsymbol{\theta})$ is the likelihood function
  - $f(\mathbf{y})$ is a constant term that is not calculated
- $\overline{D(\boldsymbol{\theta})}$ is posterior mean of the deviance, approximated by $\frac{1}{n}\sum_{t=1}^{n} D(\boldsymbol{\theta}^t)$. The expected deviation measures how well the model fits the data.
- $D(\bar{\boldsymbol{\theta}})$ is the deviance evaluated at $\bar{\boldsymbol{\theta}}$, equal to $-2\log(p(\mathbf{y}|\bar{\boldsymbol{\theta}}))$. It is the deviance evaluated at your "best" posterior estimate.
- p_D is the effective number of parameters.

DIC Computation

PROC MCMC supports a DIC option, which computes the DIC value:

```
   proc mcmc data=NTP ... DIC;
```

DIC Computation

PROC MCMC supports a DIC option, which computes the DIC value:

```
   proc mcmc data=NTP ... DIC;
```

For $a_0 = 0.3$:

```
             Deviance Information Criterion


Dbar (posterior mean of deviance)                      17.950
Dmean (deviance evaluated at posterior mean)           16.622
pD (effective number of parameters)                     1.329
DIC (smaller is better)                                19.279
```

Compare DIC Values with Different a_0

You run parallel analysis over a grid of a_0 values, choose an a_0 that produces the lowest DIC value.

Compare DIC Values with Different a_0

You run parallel analysis over a grid of a_0 values, choose an a_0 that produces the lowest DIC value.

Good time for BY group processing.

Compare DIC Values with Different a_0

You run parallel analysis over a grid of a_0 values, choose an a_0 that produces the lowest DIC value.

Good time for BY group processing.

```
data NTP_by;
   set NTP;
   do a0 = 0.05, 0.15, 0 to 1 by 0.1;
      output;
      end;
   run;

proc sort data=ntp_by;
   by a0;
   run;
```

Compare DIC Values with Different a_0

You run parallel analysis over a grid of a_0 values, choose an a_0 that produces the lowest DIC value.

Good time for BY group processing.

```
data NTP_by;
   set NTP;
   do a0 = 0.05, 0.15, 0 to 1 by 0.1;
      output;
      end;
   run;

proc sort data=ntp_by;
   by a0;
   run;
```

$\Rightarrow$

```
  y   n dose   a0
  5  75  0.0 0.00
  1  49  1.4 0.00
  3  50  7.1 0.00
 12  49 71.0 0.00
...
  5  75  0.0 0.10
  1  49  1.4 0.10
  3  50  7.1 0.10
 12  49 71.0 0.10
...
  5  75  0.0 1.00
  1  49  1.4 1.00
  3  50  7.1 1.00
 12  49 71.0 1.00
```

DIC Computation using PROC MCMC

```
ods output dic=ntp_dic;          ! save DIC results to a data set
proc mcmc data=ntp_by ... dic;
   by a0;      ! 13 simulations are performed
```

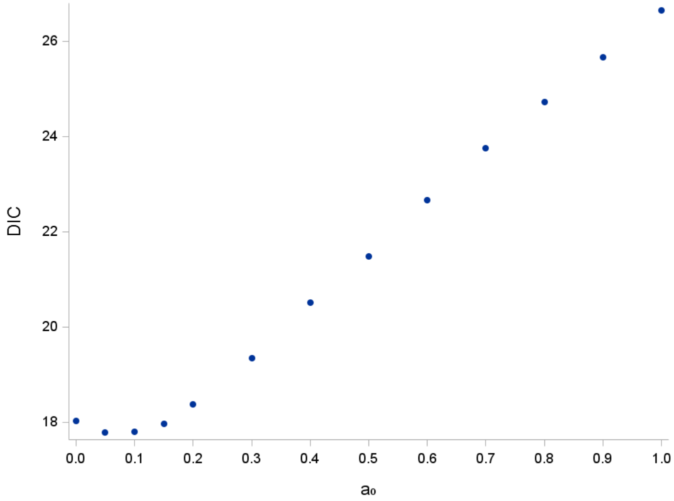
DIC Computation using PROC MCMC

```
ods output dic=ntp_dic;          ! save DIC results to a data set
proc mcmc data=ntp_by ... dic;
   by a0;      ! 13 simulations are performed

   array pdata[1] / nosymbols;
   begincnst;
   rc = read_array("kociba", pdata); ! must read in KOCIBA
   nobs = dim(pdata, 1);             ! data set separately
   endcnst;
```

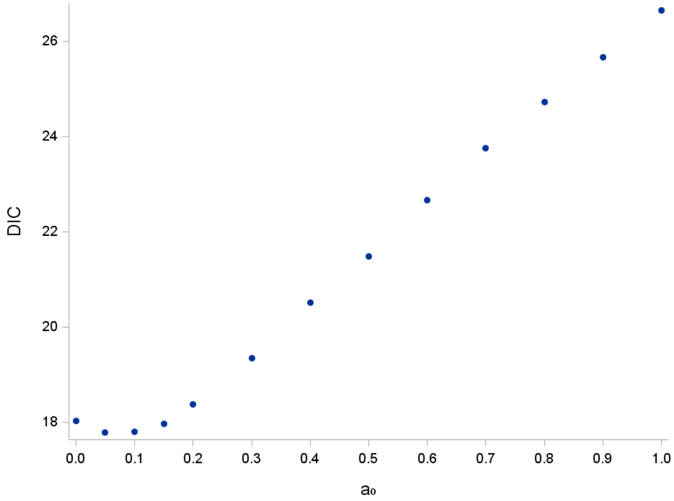
DIC Computation using PROC MCMC

```
ods output dic=ntp_dic;          ! save DIC results to a data set
proc mcmc data=ntp_by ... dic;
   by a0;      ! 13 simulations are performed

   array pdata[1] / nosymbols;
   begincnst;
   rc = read_array("kociba", pdata); ! must read in KOCIBA
   nobs = dim(pdata, 1);             ! data set separately
   endcnst;
   ...
   lp = a0 * lp; ! for each BY group, a different $a_0$ value is used.
   prior b0 b1 ~ general(lp);

   p = logistic(b0 + b1 * dose);
   model y ~ binomial(n, p);
   run;
```

DIC Values vs a_0



DIC Values vs a_0



This suggests a small value of a_0 (0.05 or 0.1) is preferred.

Variability in DIC

There are two sources of variability in DIC computation:

- distributional variability (data)
- sampling variability (monte carlo)

Variability in DIC

There are two sources of variability in DIC computation:

- distributional variability (data)
- sampling variability (monte carlo)

The data variability is difficult to get a handle on - requires repeats of the data, perhaps using bootstrap.

Variability in DIC

There are two sources of variability in DIC computation:

- distributional variability (data)
- sampling variability (monte carlo)

The data variability is difficult to get a handle on - requires repeats of the data, perhaps using bootstrap. But not always realistic.

Variability in DIC

There are two sources of variability in DIC computation:

- distributional variability (data)
- sampling variability (monte carlo)

The data variability is difficult to get a handle on - requires repeats of the data, perhaps using bootstrap. But not always realistic.

The sampling variability can be accessed by repeating the simulation many times (another BY variable) and compare the distributions of the DIC.

```
   by a0 rep;
```

Variability in DIC

There are two sources of variability in DIC computation:

- distributional variability (data)
- sampling variability (monte carlo)

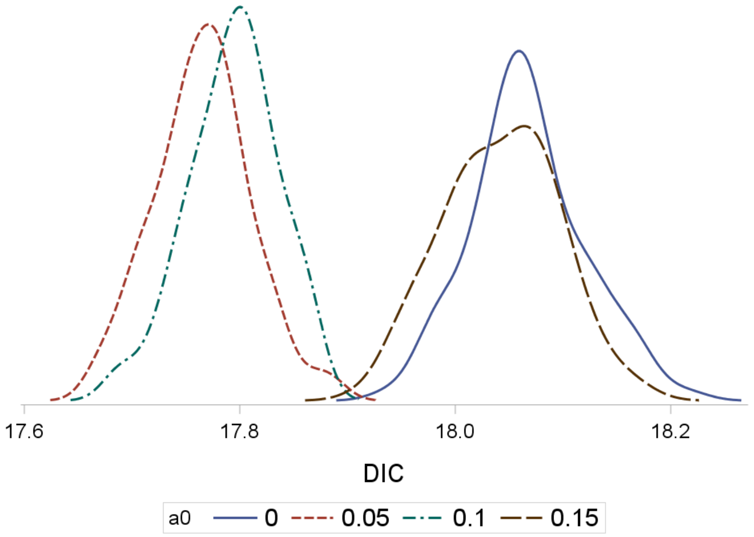
The data variability is difficult to get a handle on - requires repeats of the data, perhaps using bootstrap. But not always realistic.

The sampling variability can be accessed by repeating the simulation many times (another BY variable) and compare the distributions of the DIC.

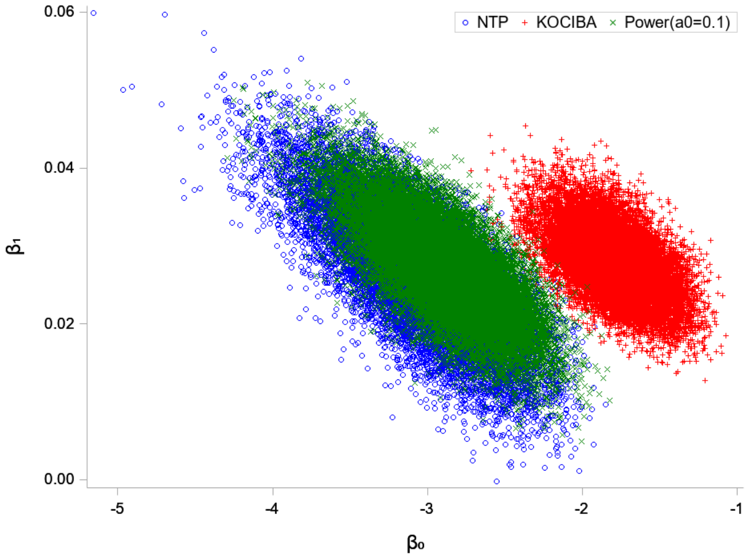
```
   by a0 rep;
```

This takes sometime to run, about five minutes (100 repeats per a_0, NMC=50,000).

Monte Carlo Variability

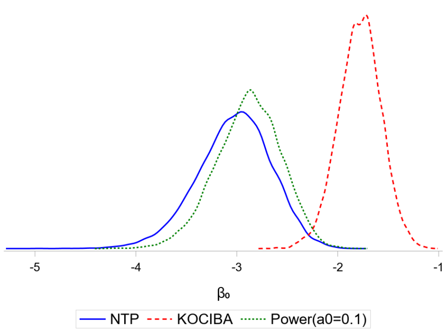


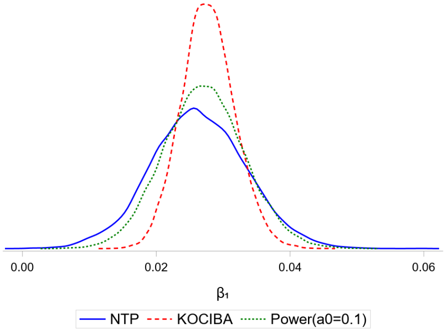
Power Prior with $a_0 = 0.1$



Compare to page 69

Marginal Posterior Comparisons





Compare to page 70

Prior on a_0

Placing a hyper prior, $\pi(a_0)$, on the weight parameter is not as straightforward as it seems.

Prior on a_0

Placing a hyper prior, $\pi(a_0)$, on the weight parameter is not as straightforward as it seems. Multiplying the unnormalized power prior, $L(\beta; D_0)^{a_0} \cdot \pi_0(\beta)$, and $\pi(a_0)$ does not lead to the right joint prior:

Prior on a_0

Placing a hyper prior, $\pi(a_0)$, on the weight parameter is not as straightforward as it seems. Multiplying the unnormalized power prior, $L(\beta; D_0)^{a_0} \cdot \pi_0(\beta)$, and $\pi(a_0)$ does not lead to the right joint prior:

$$p(\beta, a_0|D_0) \quad \propto \quad p(\beta|D_0, a_0) \cdot \pi(a_0)$$

Prior on a_0

Placing a hyper prior, $\pi(a_0)$, on the weight parameter is not as straightforward as it seems. Multiplying the unnormalized power prior, $L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})$, and $\pi(a_0)$ does not lead to the right joint prior:

$$\begin{aligned} p(\boldsymbol{\beta}, a_0 | D_0) &\propto p(\boldsymbol{\beta} | D_0, a_0) \cdot \pi(a_0) \\ &= \frac{L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})}{\int L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) d\boldsymbol{\beta}} \cdot \pi_0(a_0) \end{aligned}$$

Prior on a_0

Placing a hyper prior, $\pi(a_0)$, on the weight parameter is not as straightforward as it seems. Multiplying the unnormalized power prior, $L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})$, and $\pi(a_0)$ does not lead to the right joint prior:

$$\begin{aligned} p(\boldsymbol{\beta}, a_0 | D_0) &\propto p(\boldsymbol{\beta} | D_0, a_0) \cdot \pi(a_0) \\ &= \frac{L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})}{\int L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) d\boldsymbol{\beta}} \cdot \pi_0(a_0) \\ &= \frac{1}{C(a_0)} \cdot L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) \cdot \pi_0(a_0) \end{aligned}$$

Prior on a_0

Placing a hyper prior, $\pi(a_0)$, on the weight parameter is not as straightforward as it seems. Multiplying the unnormalized power prior, $L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})$, and $\pi(a_0)$ does not lead to the right joint prior:

$$\begin{aligned}
p(\boldsymbol{\beta}, a_0 | D_0) &\propto p(\boldsymbol{\beta} | D_0, a_0) \cdot \pi(a_0) \\
&= \frac{L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})}{\int L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) d\boldsymbol{\beta}} \cdot \pi_0(a_0) \\
&= \frac{1}{C(a_0)} \cdot L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) \cdot \pi_0(a_0) \\
&\not\propto L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) \cdot \pi_0(a_0)
\end{aligned}$$

Prior on a_0

Placing a hyper prior, $\pi(a_0)$, on the weight parameter is not as straightforward as it seems. Multiplying the unnormalized power prior, $L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})$, and $\pi(a_0)$ does not lead to the right joint prior:

$$\begin{aligned} p(\boldsymbol{\beta}, a_0 | D_0) &\propto p(\boldsymbol{\beta} | D_0, a_0) \cdot \pi(a_0) \\ &= \frac{L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})}{\int L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) d\boldsymbol{\beta}} \cdot \pi_0(a_0) \\ &= \frac{1}{C(a_0)} \cdot L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) \cdot \pi_0(a_0) \\ &\not\propto L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) \cdot \pi_0(a_0) \end{aligned}$$

The normalizing constant $C(a_0)$ requires integration.

Prior on a_0

Placing a hyper prior, $\pi(a_0)$, on the weight parameter is not as straightforward as it seems. Multiplying the unnormalized power prior, $L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})$, and $\pi(a_0)$ does not lead to the right joint prior:

$$\begin{aligned} p(\boldsymbol{\beta}, a_0 | D_0) &\propto p(\boldsymbol{\beta} | D_0, a_0) \cdot \pi(a_0) \\ &= \frac{L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})}{\int L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) d\boldsymbol{\beta}} \cdot \pi_0(a_0) \\ &= \frac{1}{C(a_0)} \cdot L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) \cdot \pi_0(a_0) \\ &\not\propto L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta}) \cdot \pi_0(a_0) \end{aligned}$$

The normalizing constant $C(a_0)$ requires integration.

For more information on the normalized Power Prior, see Neuenschwander, Branson, and Spiegelhalter (2009, *Statisti. Med.* 28:3562)

Numerical Integral Function

To compute the normalizing constant, you need an integral function (DATA step is doable, but it is complicated).

Numerical Integral Function

To compute the normalizing constant, you need an integral function (DATA step is doable, but it is complicated).

PROC MCMC supports a native `CALL QUAD` subroutine that computes the integral of a user-specific function.

Numerical Integral Function

To compute the normalizing constant, you need an integral function (DATA step is doable, but it is complicated).

PROC MCMC supports a native `CALL QUAD` subroutine that computes the integral of a user-specific function.

```
   CALL QUAD("ObjFun", Res, LLimit, ULimit <, args>);
```

Numerical Integral Function

To compute the normalizing constant, you need an integral function (DATA step is doable, but it is complicated).

PROC MCMC supports a native `CALL QUAD` subroutine that computes the integral of a user-specific function.

```
  CALL QUAD("ObjFun", Res, LLimit, ULimit <, args>);
```

ObjFun	:	name of an integrand function (defined using PROC FCMP)
Res	:	result
Limit	:	lower and upper limits (of the w.r.t. parameters)
arg	:	arguments to the ObjFun (e.g. data set variables, parameters)

Numerical Integral Function

To compute the normalizing constant, you need an integral function (DATA step is doable, but it is complicated).

PROC MCMC supports a native `CALL QUAD` subroutine that computes the integral of a user-specific function.

```
  CALL QUAD("ObjFun", Res, LLimit, ULimit <, args>);
```

ObjFun	name of an integrand function (defined using PROC FCMP)
Res	result
Limit	lower and upper limits (of the w.r.t. parameters)
arg	arguments to the ObjFun (e.g. data set variables, parameters)

The first four arguments are location specific. The w.r.t. parameter(s) is specified in the definition of the `ObjFun` function.

Define Objective Function

The objective function (e.g. $L(\boldsymbol{\beta}; D_0)^{a_0} \cdot \pi_0(\boldsymbol{\beta})$) is defined using PROC FCMP:

Define Objective Function

The objective function (e.g. $L(\beta; D_0)^{a_0} \cdot \pi_0(\beta)$) is defined using PROC FCMP:

```
PROC FCMP outlib=sasuser.funcs.power;
   SUBROUTINE ObjFun(parm, obj, vars);
   OUTARGS obj;
   obj = f(parm, vars ...);
   endsub;
run;
```

outlib	location to store the objective function
parm	w.r.t. parameters (e.g. β)
obj	integrand (e.g. $C(a_0)$), must be declared as an OUTARGS
vars	variables needed to construct the integrand

Integration over D_0

Integral of sums is not the sum of integrals!

Integration over D_0

Integral of sums is not the sum of integrals!

The integral must be computed using the entire historical data set (D_0).

Integration over D_0

Integral of sums is not the sum of integrals!

The integral must be computed using the entire historical data set (D_0).

You cannot use the combined dataset approach to compute integral by observation.

Integration over D_0

Integral of sums is not the sum of integrals!

The integral must be computed using the entire historical data set (D_0).

You cannot use the combined dataset approach to compute integral by observation.

The object function must be written using the `pdata` array.

Specifying $[L(\beta_0, \beta_1 | D_0)]^{a_0}$ in PROC FCMP

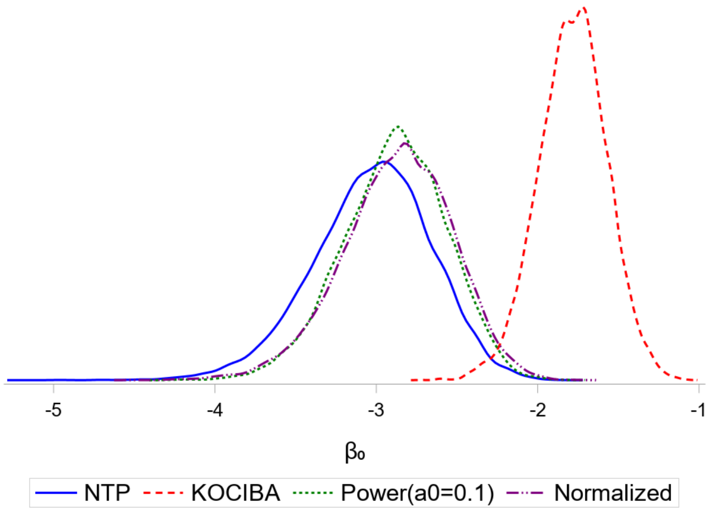
```
proc fcmp outlib=sasuser.funcs.power;
   subroutine bPower(beta[*], den, pdata[*,*], a0); !integration w.r.t. β
   outargs den;
   nobs = dim(pdata, 1);
   lp = 0;
   do j = 1 to nobs;
      p = logistic(beta[1] + beta[2] * pdata[j,3]);
      lp = lp + logpdf("binomial", pdata[j,1], p, pdata[j,2]);
      end;
   den = exp(a0 * lp);        ! [L(β0, β1|D0)]^a0
   endsub;
run;
```

The OUTLIB= option specifies the library that stores the objective function.

Fitting Normalized Power Prior in PROC MCMC

```
options cmplib=sasuser.funcs;
proc mcmc data=ntp ...
   ...
   array beta[2] b0 b1;
   array lower[2] -100 -100;   ! integration lower bound
   array upper[2]  100  100;   ! integration upper bound

   prior a0 ~ uniform(0, 1);    ! a0 is a parameter
   beginprior;
   lp = 0;
   do j = 1 to nobs;
      p = logistic(beta[1] + beta[2] * pdata[j,3]);
      lp = lp + logpdf("binomial", pdata[j,1], p, pdata[j,2]);
      end;
   CALL QUAD('bPower', C, lower, upper, pdata, a0); ! C = C(a0)
   lp = -log(C) + a0 * lp;
   endprior;
   prior b0 b1 ~ general(lp);
   ...
   run;
```

Normalized Power Prior is Similar to $a_0 = 0.1$



Selection of a_0

- On one hand, the normalized power prior provides an automated approach in selecting a_0.

Selection of a_0

- On one hand, the normalized power prior provides an automated approach in selecting a_0.
- But it is quite computationally intensive

Selection of a_0

- On one hand, the normalized power prior provides an automated approach in selecting a_0.
- But it is quite computationally intensive
    - Numerical integral can be costly to compute, and the problem gets worse as the dimension of the model gets to be larger.

Selection of a_0

- On one hand, the normalized power prior provides an automated approach in selecting a_0.
- But it is quite computationally intensive
  - Numerical integral can be costly to compute, and the problem gets worse as the dimension of the model gets to be larger.
  - In addition, normalized power prior requires coding the likelihood function at three places: MODEL statement, PRIOR statement, and in the (integral) objective function.

Selection of a_0

- On one hand, the normalized power prior provides an automated approach in selecting a_0.
- But it is quite computationally intensive
    - Numerical integral can be costly to compute, and the problem gets worse as the dimension of the model gets to be larger.
    - In addition, normalized power prior requires coding the likelihood function at three places: MODEL statement, PRIOR statement, and in the (integral) objective function.
    - This is prone to coding errors, and can be difficult in maintaining production code

Selection of a_0

- On one hand, the normalized power prior provides an automated approach in selecting a_0.
- But it is quite computationally intensive
  - Numerical integral can be costly to compute, and the problem gets worse as the dimension of the model gets to be larger.
  - In addition, normalized power prior requires coding the likelihood function at three places: MODEL statement, PRIOR statement, and in the (integral) objective function.
  - This is prone to coding errors, and can be difficult in maintaining production code
- Alternatively, grid-based search over DIC can be effective.

Selection of a_0

- On one hand, the normalized power prior provides an automated approach in selecting a_0.
- But it is quite computationally intensive
  - Numerical integral can be costly to compute, and the problem gets worse as the dimension of the model gets to be larger.
  - In addition, normalized power prior requires coding the likelihood function at three places: MODEL statement, PRIOR statement, and in the (integral) objective function.
  - This is prone to coding errors, and can be difficult in maintaining production code
- Alternatively, grid-based search over DIC can be effective. Although the plug-in method does not account for the uncertainty in a_0, often the difference is relatively minor.

Outline

Evaluation of a Basket Clinical Trial Design

The goal is to evaluate the drug response and select cohorts for further study.

- A basket adaptive design enrolls patients across cohorts
- Evaluate the performance at an interim analysis for each cohort to either continue enroll, or stop for efficacy or futility

In this example, there are 10 cohorts with 80 patients, and different cohorts have different enrollment rates. The endpoint is clinical response rate:

$$H_0 : \theta = 10\% \quad vs \quad H_1 : \theta = 35\%$$

Thanks to Frank Liu (Merck) for his help with this example.

Hierarchical Models

A logistic random-effects model is used to fit all the cohorts patients. For $j = 1, \cdots, 10$:

$$\begin{aligned} y_j &\sim \text{binomial}(n_j, \theta_j) \\ \theta_j &= \frac{\exp(\mu_j)}{(1 + \exp(\mu_j))} \\ \mu_j &\sim \text{normal}(\mu, \tau) \\ \mu &\sim \text{normal}(0, \text{prec} = 0.001) \\ \tau &\sim \text{gamma}(0.01, \text{iscale} = 0.01) \end{aligned}$$

The decision criteria are

1. stop for futility if $P(\theta_j > 0.225) < 0.05$
2. stop for efficacy if $P(\theta_j > 0.225) > 0.85$
3. otherwise, continue enrollment (in adaptive design)

Simulation Details

Draw number of cohort patients from a multinomial distribution with ntotal $= 80$, with analysis carried out in cohorts that have $n_j > 5$:

$$(n_1, n_2, \cdots, n_{10}) \sim \text{Multi}(p_1, p_2, \cdots, p_{10}), \quad \sum_i p_i = 1$$

where the allocation probabilities are set to be

$$p_1 = \cdots = p_6 = 0.14, \quad p_7 = \cdots = p_{10} = 0.04$$

and consider three scenarios of true response rates:

1. $\theta_j = 0.35$ for all cohorts (strong alternative)
2. $\theta_j = 0.1$ for all cohorts (strong null)
3. $\theta_1 = \cdots = \theta_4 = 0.35; \quad \theta_5 = \cdots = \theta_{10} = 0.1$

Simulate Cohort Patients Data

```
data Alloc;
   array p[10] (0.14 0.14 0.14 0.14 0.14 0.14 0.04 0.04 0.04 0.04);   ! Multinomial probability vector
   array theta[3, 10] (0.35 0.35 0.35 0.35 0.35 0.35 0.35 0.35 0.35 0.35, ! three true response rates
                       0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10,
                       0.35 0.35 0.35 0.35 0.10 0.10 0.10 0.10 0.10 0.10);
   array n[10]; array y[10];
   call streaminit(12467);
   do RespRate = 1 to 3;     !   Do-loop over three scenarios
      do Rep = 1 to 5000;    !   5000 Repeats
         do i = 1 to 10;  n[i] = 0; end;
         do i = 1 to 80;                  ! Draw Multinomial Samples, ntotal=80
            j = rand("table", of p[*]);   ! The table RNG draws an index
            n[j]+1;                       ! Increase count of according to that index
            end;
         do i = 1 to 10;
            y[i] = .;
            if (n[i] > 5) then                 ! Only draw y if the number of patients is greater than 5
               y[i] = rand("binomial", theta[RespRate, i], n[i]);  !   Draw Responses according to θ
            end;
         output;
         end;
      end;
   drop p: w theta: i j;
   run;
```

The RespRate and Rep variables become the BY variables.

Simulated Data Set

```
                         n                    y     r
 n  n  n  n  n  n n n  n 1  y y y y y y y y y 1  R  e
 1  2  3  4  5  6 7 8  9 0  1 2 3 4 5 6 7 8 9 0  R  p

13 17 15 12 11  8 1 1  1 1  6 8 3 2 1 4 . . . .  1  1
11 13  9 17 11  9 3 2  2 3  4 8 3 6 6 4 . . . .  1  2
...
 4 15 11 11 19  6 2 5  3 4  . 0 0 1 1 0 . . . .  2  1
 7  8  5 18 12 11 4 8  3 4  0 3 . 2 1 1 . 1 . .  2  2
...
11 11  7 12 18 14 1 2  2 2  4 5 3 3 1 3 . . . .  3  1
14 16 13  8 10  6 3 4  4 2  4 4 4 3 2 2 . . . .  3  2
...
```

Simulated Data Set

```
                         n                    y     r
 n  n  n  n  n  n n n  n 1  y y y y y y y y y 1  R  e
 1  2  3  4  5  6 7 8  9 0  1 2 3 4 5 6 7 8 9 0  R  p

13 17 15 12 11  8 1 1  1 1  6 8 3 2 1 4 . . . .  1  1
11 13  9 17 11  9 3 2  2 3  4 8 3 6 6 4 . . . .  1  2
...
 4 15 11 11 19  6 2 5  3 4  . 0 0 1 1 0 . . . .  2  1
 7  8  5 18 12 11 4 8  3 4  0 3 . 2 1 1 . 1 . .  2  2
...
11 11  7 12 18 14 1 2  2 2  4 5 3 3 1 3 . . . .  3  1
14 16 13  8 10  6 3 4  4 2  4 4 4 3 2 2 . . . .  3  2
...
```

Note that $y = 0$ is different from y=missing (.).

Simulated Data Set

```
                         n                    y     r
 n  n  n  n  n  n n n  n 1  y y y y y y y y y 1  R  e
 1  2  3  4  5  6 7 8  9 0  1 2 3 4 5 6 7 8 9 0  R  p

13 17 15 12 11  8 1 1  1 1  6 8 3 2 1 4 . . . .  1  1
11 13  9 17 11  9 3 2  2 3  4 8 3 6 6 4 . . . .  1  2
...
 4 15 11 11 19  6 2 5  3 4  . 0 0 1 1 0 . . . .  2  1
 7  8  5 18 12 11 4 8  3 4  0 3 . 2 1 1 . 1 . .  2  2
...
11 11  7 12 18 14 1 2  2 2  4 5 3 3 1 3 . . . .  3  1
14 16 13  8 10  6 3 4  4 2  4 4 4 3 2 2 . . . .  3  2
...
```

Note that $y = 0$ is different from y=missing (.). Simulation should include observations with $y = 0$ (groups with enough enrollment) but not $y = .$ (groups don't have enough enrollment, hence not part of the trial).

Input Data Set to PROC MCMC

```
Resp
Rate      rep      k     n     y
  1        1       1    13     6
  1        1       2    17     8
  1        1       3    15     3
  1        1       4    12     2
  1        1       5    11     1
  1        1       6     8     4
  1        1       7     1     .
  1        1       8     1     .
  1        1       9     1     .
  1        1      10     1     .
  1        2       1    11     4
  1        2       2    13     8
  1        2       3     9     3
  1        2       4    17     6
  1        2       5    11     6
  1        2       6     9     4
  1        2       7     3     .
```

Fitting Hierarchical Model in PROC MCMC

Each simulated data set is fitted using a binomial random-effects model:

```
proc mcmc data=alloc ... missing=cc;
   by RespRate Rep;
   parm mu tau;
   prior mu ~ normal(0, prec=0.001);
   prior tau ~ gamma(shape=0.01, iscale=0.01);
   random u ~ normal(mu, prec=tau) subject=k;
   model y ~ binomial(n, logistic(u));
   run;
```

Fitting Hierarchical Model in PROC MCMC

Each simulated data set is fitted using a binomial random-effects model:

```
proc mcmc data=alloc ... missing=cc;
   by RespRate Rep;
   parm mu tau;
   prior mu ~ normal(0, prec=0.001);
   prior tau ~ gamma(shape=0.01, iscale=0.01);
   random u ~ normal(mu, prec=tau) subject=k;
   model y ~ binomial(n, logistic(u));
   run;
```

- The `missing=cc` option discard observations with missing response (y's). Fitting models of different sizes in each BY group.

Fitting Hierarchical Model in PROC MCMC

Each simulated data set is fitted using a binomial random-effects model:

```
proc mcmc data=alloc ... missing=cc;
   by RespRate Rep;
   parm mu tau;
   prior mu ~ normal(0, prec=0.001);
   prior tau ~ gamma(shape=0.01, iscale=0.01);
   random u ~ normal(mu, prec=tau) subject=k;
   model y ~ binomial(n, logistic(u));
   run;
```

- The missing=cc option discard observations with missing response (y's). Fitting models of different sizes in each BY group.
- There are a total of 3×5000 of BY groups.

Fitting Hierarchical Model in PROC MCMC

Each simulated data set is fitted using a binomial random-effects model:

```
proc mcmc data=alloc ... missing=cc;
   by RespRate Rep;
   parm mu tau;
   prior mu ~ normal(0, prec=0.001);
   prior tau ~ gamma(shape=0.01, iscale=0.01);
   random u ~ normal(mu, prec=tau) subject=k;
   model y ~ binomial(n, logistic(u));
   run;
```

- The `missing=cc` option discard observations with missing response (y's). Fitting models of different sizes in each BY group.
- There are a total of 3×5000 of BY groups.
- Each BY group can have potentially different number of parameters (in random effects).

Fit the Same Model using PROC BGLIMM

```
proc bglimm data=alloc outpost=out seed=720517 nmc=20000
   stats=none diag=none plots=none missing=cc;
   by RespRate rep;
   class k;
   model y/n = / dist=binomial link=logit;
   random int / subject=k covprior=igamma(shape=0.01 scale=0.01);
run;
```

Fit the Same Model using PROC BGLIMM

```
proc bglimm data=alloc outpost=out seed=720517 nmc=20000
   stats=none diag=none plots=none missing=cc;
   by RespRate rep;
   class k;
   model y/n = / dist=binomial link=logit;
   random int / subject=k covprior=igamma(shape=0.01 scale=0.01);
run;
```

- The model is the same: a binomial random-effects logistic regression

Fit the Same Model using PROC BGLIMM

```
proc bglimm data=alloc outpost=out seed=720517 nmc=20000
   stats=none diag=none plots=none missing=cc;
   by RespRate rep;
   class k;
   model y/n = / dist=binomial link=logit;
   random int / subject=k covprior=igamma(shape=0.01 scale=0.01);
run;
```

- The model is the same: a binomial random-effects logistic regression
- The `k`-level random intercepts enter the regressor linearly

Fit the Same Model using PROC BGLIMM

```
proc bglimm data=alloc outpost=out seed=720517 nmc=20000
   stats=none diag=none plots=none missing=cc;
   by RespRate rep;
   class k;
   model y/n = / dist=binomial link=logit;
   random int / subject=k covprior=igamma(shape=0.01 scale=0.01);
run;
```

- The model is the same: a binomial random-effects logistic regression
- The `k`-level random intercepts enter the regressor linearly
- The `COVPRIOR=` option specifies the prior for the shrinkage parameter (of the random effects)

Fit the Same Model using PROC BGLIMM

```
proc bglimm data=alloc outpost=out seed=720517 nmc=20000
   stats=none diag=none plots=none missing=cc;
   by RespRate rep;
   class k;
   model y/n = / dist=binomial link=logit;
   random int / subject=k covprior=igamma(shape=0.01 scale=0.01);
run;
```

- The model is the same: a binomial random-effects logistic regression
- The k-level random intercepts enter the regressor linearly
- The COVPRIOR= option specifies the prior for the shrinkage parameter (of the random effects)
- The generate 300 million posterior samples. The rest is counting.

The Rest is Counting

For each posterior sample of u_j, you compute if `logistic(u) > 0.225` (result in 0 or 1):

```
                                                    u
  r    u      u      u      u      u      u   u u u _ p p p p p p
R e    _      _      _      _      _      _   _ _ _ 1 T T T T T T
R p    1      2      3      4      5      6   7 8 9 0 1 2 3 4 5 6


1 1 -0.286 -1.441 -0.472 -1.819 -2.489  0.124 . . . . 1 0 1 0 0 1
1 1 -0.286  0.632 -0.472 -1.819 -3.054  0.124 . . . . 1 1 1 0 0 1
1 1  0.472 -0.342 -0.472 -1.819 -3.054 -1.041 . . . . 1 1 1 0 0 1
1 1  0.472 -0.342 -0.472 -2.042 -1.277 -1.041 . . . . 1 1 1 0 0 1
1 1 -1.158  0.337 -0.472 -2.042 -1.277 -1.041 . . . . 1 1 1 0 0 1
1 1 -0.558  0.337 -0.472 -2.042 -1.455 -0.385 . . . . 1 1 1 0 0 1
1 1 -0.558  0.337 -0.472 -2.042 -1.455 -0.862 . . . . 1 1 1 0 0 1
1 1 -0.558  0.337 -1.483 -2.287 -0.054 -0.862 . . . . 1 1 0 0 1 1
1 1 -0.558 -0.105 -1.483 -2.287 -0.054  0.742 . . . . 1 1 0 0 1 1
1 1 -0.915 -0.105 -1.483 -2.287 -0.487 -0.755 . . . . 1 1 0 0 1 1
...
```

You Monte carlo over the 20,000 zero-one indicator variables (per `repeat`) to estimate the probabilities:

```
Resp
Rate rep  pT1   pT2   pT3   pT4   pT5   pT6   pT7   pT8   pT9  pT1

   1    1 0.96  0.98  0.66  0.62  0.52  0.93     .     .     .
   1    2 0.99     1  0.98  0.99     1  0.99     .     .     .
   1    3 0.95  0.99  0.99  0.91  0.97  0.99     .     .     .  0.9
   1    4 0.93  0.93  0.81  0.82  0.72  0.83     .     .     .
   1    5 0.67     .  0.55  0.58  0.56  0.59     .     .  0.66
...
```

Check for Futility and Efficacy

Now we compare the posterior probabilities with the decision criteria (0.05 for futility and 0.85 for efficacy), and get another bunch of zero-one indicator variables.

Check for Futility and Efficacy

Now we compare the posterior probabilities with the decision criteria (0.05 for futility and 0.85 for efficacy), and get another bunch of zero-one indicator variables.

Again, average over these indicator variables (5000 `repeats`) get us the estimates of the probabilities of trial reach one of the three decisions:

- Early stop for futility
- Early stop for Efficacy
- Trial is inconclusive

Probability Early Stopping for Futility

grp	coh1	coh2	coh3	coh4	coh5	coh6	coh7	coh8	coh9	coh10
1	.0002	.0000	.0004	.0002	.0000	.0004	.0000	.0000	.0000	.0000
2	.6637	.6790	.6819	.6764	.6737	.6669	.5934	.6315	.6553	.6163
3	.0072	.0066	.0059	.0054	.0859	.0862	.0432	.0479	.0451	.0421

Probability of Early Stopping for Efficacy

grp	coh1	coh2	coh3	coh4	coh5	coh6	coh7	coh8	coh9	coh10
1	.7002	.6895	.6916	.6918	.6912	.6856	.7033	.6764	.6679	.6190
2	.0002	.0004	.0008	.0004	.0002	.0002	.0000	.0000	.0000	.0020
3	.3336	.3258	.3286	.3275	.0258	.0247	.0247	.0077	.0132	.0165

Probability of Trial is Inconclusive:

grp	coh1	coh2	coh3	coh4	coh5	coh6	coh7	coh8	coh9	coh10
1	.2974	.3105	.3076	.3080	.3092	.3132	.2986	.3198	.3283	.3810
2	.3373	.3196	.3156	.3240	.3244	.3321	.3860	.3566	.3527	.3796
3	.6625	.6709	.6652	.6651	.8859	.8922	.9300	.9387	.9380	.9304

Adaptive Randomization

- **Algorithm for AR Design**
  - **Step 1. Early Loser:** If the probability that treatment arm k is the best falls below some prespecified probability p_L, i.e., if

    $$P(\theta_k > \theta_{j \neq k} | \text{Data}) < p_L,$$

    then arm k is declared a loser and suspended. Normally, we take $p_L \leq 0.10$.
  - **Step 2. Early Winner:** If the probability that treatment arm k is the best exceeds some prespecified probability p_U, i.e., if

    $$P(\theta_k > \theta_{j \neq k} | \text{Data}) > p_U,$$

    then arm k is declared the winner and the trial is stopped early. We typically take p_U fairly large. In a two-arm trial we would take $p_U = 1 - p_L$.

Adaptive Randomization

- **Step 3. Final winner:** If, after all patients have been evaluated, the probability that treatment arm k is best exceeds some prespecified probability, p_U^*, i.e., if

$$P(\theta_k > \theta_{j \neq k} | \text{Data}) > p_U^*,$$

then arm k is declared the winner. If no treatment arm can meet this criterion, the AR program does not make a final selection. One typically sets $p_U^* < p_U$ (say, between 0.70 and 0.90) to increase the chance of obtaining a final winner.

Adaptive Randomization

- **Step 4. Futility:** If the probability that treatment arm k is better than some prespecified minimally tolerable response rate, θ_{min}, falls below some prespecified probability p_L^*, i.e., if

$$P(\theta_k > \theta_{min} | \text{Data}) < p_L^*,$$

then arm k is declared futile and will not accrue more patients. This rule applies only in efficacy trials. We take $p_L^* \leq 0.10$, Once an arm is declared futile, it cannot be re-activated.

Adaptive Randomization

- As each new patient enters the trial, the randomization probability is updated. Assuming a trial with m arms, the probability of arm k being assigned next is
$$\frac{P(\theta_k = \mathrm{max}_j \theta_j | \mathrm{Data})^c}{\sum_{i=1}^{m} P(\theta_i = \mathrm{max}_j \theta_j | \mathrm{Data})^c},$$
where $c \geq 0$.
- $c = 0$ corresponds to equal randomization. Typically, c is chosen to be some significant fraction of the sample size, such as $c = n/2N$, where N is the maximum number of patients and n is the number of currently enrolled patients.
- In general, values of c near 1 and no bigger than 2 are recommended.

Adaptive Randomization

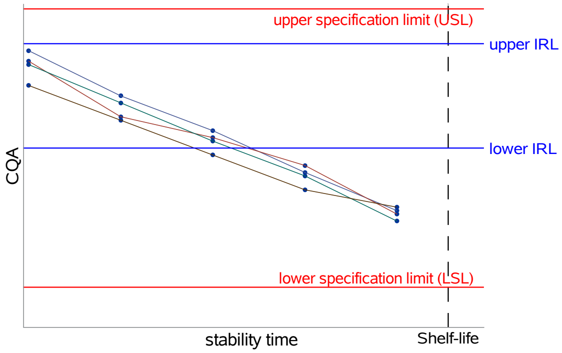
- As each new patient enters the trial, the randomization probability is updated. Assuming a trial with m arms, the probability of arm k being assigned next is
$$\frac{P(\theta_k = \text{max}_j \theta_j | \text{Data})^c}{\sum_{i=1}^{m} P(\theta_i = \text{max}_j \theta_j | \text{Data})^c},$$
where $c \geq 0$.
- $c = 0$ corresponds to equal randomization. Typically, c is chosen to be some significant fraction of the sample size, such as $c = n/2N$, where N is the maximum number of patients and n is the number of currently enrolled patients.
- In general, values of c near 1 and no bigger than 2 are recommended.

This can't be done using BY group and one must write a macro do-loop to carry out the simulation.

Outline

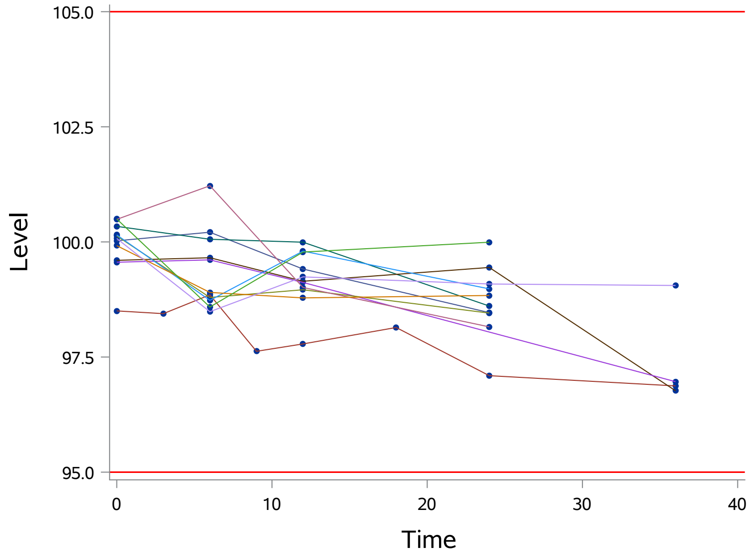
Internal Release Limits

- drug stability: the capacity of a drug to remain within limits before expiry date (shelf-life)
- Internal Release Limits: a window which guarantees with a defined level of confidence that a batch remains within specifications throughout its shelf-life

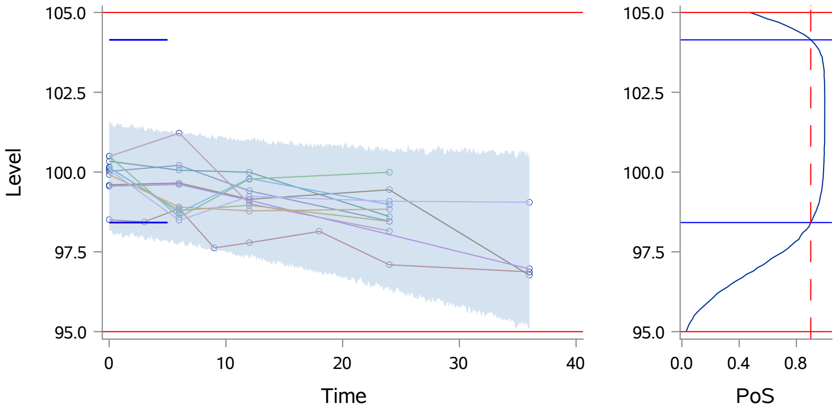


Thanks to Laurent Natalis (PharmaLex) for the data and help with this example.

Data



You Want to Get



where the blue bars are the lower and upper IRLs.

What are Internal Release Limits

From a modeling perspective, we want to find an interval, $(IRL_{lower}, IRL_{upper})$, such that, when the initial measurement (at time 0, $y_{t=0}$) falls within this interval, then

What are Internal Release Limits

From a modeling perspective, we want to find an interval, $(IRL_{lower}, IRL_{upper})$, such that, when the initial measurement (at time 0, $y_{t=0}$) falls within this interval, then

$$\Pr(\{y_{t=1}, \cdots, y_{t=SL}\} \in (LL, UL)) > 95\%$$

What are Internal Release Limits

From a modeling perspective, we want to find an interval, $(IRL_{lower}, IRL_{upper})$, such that, when the initial measurement (at time 0, $y_{t=0}$) falls within this interval, then

$$\Pr(\{y_{t=1}, \cdots, y_{t=SL}\} \in (LL, UL)) > 95\%$$

If we consider a monotone linear model (with negative slope), it is sufficient to find the interval based on the last measurement point:

$$\Pr(y_{t=SL} \in (LL, UL)) > 95\%$$

What are Internal Release Limits

From a modeling perspective, we want to find an interval, $(IRL_{lower}, IRL_{upper})$, such that, when the initial measurement (at time 0, $y_{t=0}$) falls within this interval, then

$$\Pr(\{y_{t=1}, \cdots, y_{t=SL}\} \in (LL, UL)) > 95\%$$

If we consider a monotone linear model (with negative slope), it is sufficient to find the interval based on the last measurement point:

$$\Pr(y_{t=SL} \in (LL, UL)) > 95\%$$

Because we don't know what the true value is at $t = 0$ (measurement errors), we find the interval based on both end points:

$$\Pr(\{y_{t=0}, y_{t=SL}\} \in (LL, UL)) > 95\%$$

Steps in Estimating IRLs

1. Fit a model to the data (random intercept/random slope model in this case)

Steps in Estimating IRLs

1. Fit a model to the data (random intercept/random slope model in this case)
2. Compute predictive distributions at different time points (blue band)

Steps in Estimating IRLs

1. Fit a model to the data (random intercept/random slope model in this case)
2. Compute predictive distributions at different time points (blue band)
3. Probability of Success (PoS) curve

Steps in Estimating IRLs

1. Fit a model to the data (random intercept/random slope model in this case)
2. Compute predictive distributions at different time points (blue band)
3. Probability of Success (PoS) curve
   - Estimate the joint (predictive) distribution of $y_{t=0}$ and $y_{t=SL}$

Steps in Estimating IRLs

1. Fit a model to the data (random intercept/random slope model in this case)
2. Compute predictive distributions at different time points (blue band)
3. Probability of Success (PoS) curve
   - Estimate the joint (predictive) distribution of $y_{t=0}$ and $y_{t=SL}$
   - Compute the Probability of Success (PoS) of $y_{t=SL} \in [LL, UL]$, given different values of $y_{t=0}$, estimate a curve

Steps in Estimating IRLs

1. Fit a model to the data (random intercept/random slope model in this case)
2. Compute predictive distributions at different time points (blue band)
3. Probability of Success (PoS) curve
   - Estimate the joint (predictive) distribution of $y_{t=0}$ and $y_{t=SL}$
   - Compute the Probability of Success (PoS) of $y_{t=SL} \in [LL, UL]$, given different values of $y_{t=0}$, estimate a curve
4. Estimate $(IRL_{lower}, IRL_{upper})$

Part of the Data Set

```
Batch      TIME       LEVEL


V2_0        12       99.411
V2_0        24       98.464
V2_0         6      100.210
V2_1        12       97.785
V2_1        18       98.142
V2_1         3       98.442
V2_1         6       98.850
V2_1         9       97.625
V2_2         6       99.656
...
```

There are 11 batches and 50 observations (unbalanced).

Step 1: Random-Effects Model

Random intercept and random slope model:

$$\begin{aligned}
y_{ij} &\sim \mathrm{N}(\mu_{ij}, \sigma_y^2) \\
\mu_{ij} &= \gamma_{0,j} + \gamma_{t,j} \cdot TIME_{ij} \\
\gamma_{0,j} &\sim \mathrm{N}(\beta_0, \sigma_{\gamma_0}^2) \\
\gamma_{t,j} &\sim \mathrm{N}(\beta_t, \sigma_{\gamma_t}^2) \\
\sigma_y^2, \sigma_{\gamma_0}^2, \sigma_{\gamma_t}^2 &\sim \text{half-Cauchy} \\
\beta_0, \beta_t &\sim \mathrm{N}(0, 10^6)
\end{aligned}$$

where i and j represent the i-th measurement in the j-th batch.

Fitting Random-Effects Model using PROC MCMC

```
proc mcmc data=irl nmc=10000 nbi=1000 seed=107561
      outpost=irlOut alg=nuts;
   parm b0 bT;
   parms s2g0 s2gT s2y / slice;
   prior b: ~ n(0, sd=1e6);
   prior s2: ~ cauchy(0, 1, lower=0);
   random g0 ~ n(0, var=s2g0) subject=batch;
   random gT ~ n(0, var=s2gT) subject=batch;
   mu = b0 + bt * time + g0 + gT * time;
   model level ~ normal(mu, var=s2y);
   run;
```

Fitting Random-Effects Model using PROC MCMC

```
proc mcmc data=irl nmc=10000 nbi=1000 seed=107561
      outpost=irlOut alg=nuts;
   parm b0 bT;
   parms s2g0 s2gT s2y / slice;
   prior b: ~ n(0, sd=1e6);
   prior s2: ~ cauchy(0, 1, lower=0);
   random g0 ~ n(0, var=s2g0) subject=batch;
   random gT ~ n(0, var=s2gT) subject=batch;
   mu = b0 + bt * time + g0 + gT * time;
   model level ~ normal(mu, var=s2y);
   run;
```

The posterior mean estimate of β_t is negative, an overall declining slope.

Step 2: Posterior Prediction

The posterior predictive distribution is the distribution of unobserved observations (prediction), conditional on the observed data.

Step 2: Posterior Prediction

The posterior predictive distribution is the distribution of unobserved observations (prediction), conditional on the observed data.

$$\begin{aligned}
\pi(\mathbf{y}_{\text{pred}}|\mathbf{y}) &= \int \pi(\mathbf{y}_{\text{pred}}, \boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta} \\
&= \int \pi(\mathbf{y}_{\text{pred}}|\boldsymbol{\theta}, \mathbf{y}) \pi(\boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta} \\
&= \int \pi(\mathbf{y}_{\text{pred}}|\boldsymbol{\theta}) \pi(\boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta}
\end{aligned}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \boldsymbol{\gamma}\}$, fixed and random effects.

Step 2: Posterior Prediction

The posterior predictive distribution is the distribution of unobserved observations (prediction), conditional on the observed data.

$$\begin{aligned}\pi(\mathbf{y}_{\text{pred}}|\mathbf{y}) &= \int \pi(\mathbf{y}_{\text{pred}}, \boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta} \\ &= \int \pi(\mathbf{y}_{\text{pred}}|\boldsymbol{\theta}, \mathbf{y})\pi(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta} \\ &= \int \pi(\mathbf{y}_{\text{pred}}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}\end{aligned}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \boldsymbol{\gamma}\}$, fixed and random effects.

This distribution does not depend on parameters - all uncertainties are integrated out, including those from the random effects.

Step 2: Posterior Prediction

We want to make predictions on a hypothetical new batch over different time points.

Step 2: Posterior Prediction

We want to make predictions on a hypothetical new batch over different time points.

For every time point (TIME=36, for example), you use the posterior samples (OUTPOST= data set, $k = 1, \cdots, 10000$) to draw the random effects and the predicted values:

1. draw $\gamma_{0,k} \sim N(\beta_{0,k}, \sigma^2_{\gamma_0,k})$
2. draw $\gamma_{t.k} \sim N(\beta_{t,k}, \sigma^2_{\gamma_t,k})$

Step 2: Posterior Prediction

We want to make predictions on a hypothetical new batch over different time points.

For every time point (TIME=36, for example), you use the posterior samples (OUTPOST= data set, $k = 1, \cdots, 10000$) to draw the random effects and the predicted values:

1. draw $\gamma_{0,k} \sim N(\beta_{0,k}, \sigma^2_{\gamma_0,k})$
2. draw $\gamma_{t.k} \sim N(\beta_{t,k}, \sigma^2_{\gamma_t,k})$
3. compute over all mean: $\mu = \gamma_{0,k} + \gamma_{t,k} \cdot TIME$
4. draw $y_k \sim N(\mu, \sigma^2_{y,k})$

Step 2: Posterior Prediction

We want to make predictions on a hypothetical new batch over different time points.

For every time point (TIME=36, for example), you use the posterior samples (OUTPOST= data set, $k = 1, \cdots, 10000$) to draw the random effects and the predicted values:

1. draw $\gamma_{0,k} \sim N(\beta_{0,k}, \sigma^2_{\gamma_0,k})$
2. draw $\gamma_{t.k} \sim N(\beta_{t,k}, \sigma^2_{\gamma_t,k})$
3. compute over all mean: $\mu = \gamma_{0,k} + \gamma_{t,k} \cdot TIME$
4. draw $y_k \sim N(\mu, \sigma^2_{y,k})$

Repeat the process for all time points.

SAS Code Drawing from Posterior Predictive Distribution

```
/* set up a fine grid */
data pred;
   do time = 0 to 36 by 0.1;
      output;
      end;
   run;

/* count the length */
data _null_;
   set pred nobs=nobs;
   call symputx('n', nobs);
   stop;
run;
```

SAS Code Drawing from Posterior Predictive Distribution

```
/* set up a fine grid */
data pred;
   do time = 0 to 36 by 0.1;
      output;
      end;
   run;

/* count the length */
data _null_;
   set pred nobs=nobs;
   call symputx('n', nobs);
   stop;
run;
```

```
data irlPred;
  set irlOut; /* posterior samples */
  array newY[&n];
  call streaminit(112071);
  do j = 1 to &n;
    set pred point=j; /* pred data set */
    g0    = rand("normal", b0, sqrt(s2g0));
    gT    = rand("normal", bt, sqrt(s2gT));
    muY   = g0 + gT * time;
    newY[j] = rand("normal", muY, sqrt(s2y))
    end;
  output;
  keep newY:;
run;
```

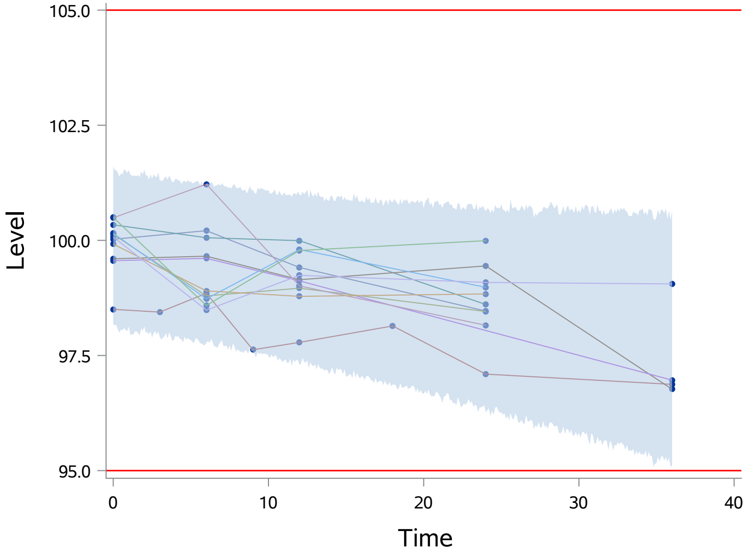
SAS Code Drawing from Posterior Predictive Distribution

```
/* set up a fine grid */
data pred;
   do time = 0 to 36 by 0.1;
      output;
      end;
   run;

/* count the length */
data _null_;
   set pred nobs=nobs;
   call symputx('n', nobs);
   stop;
run;
```

```
data irlPred;
  set irlOut; /* posterior samples */
  array newY[&n];
  call streaminit(112071);
  do j = 1 to &n;
    set pred point=j; /* pred data set */
    g0   = rand("normal", b0, sqrt(s2g0));
    gT   = rand("normal", bt, sqrt(s2gT));
    muY  = g0 + gT * time;
    newY[j] = rand("normal", muY, sqrt(s2y))
    end;
  output;
  keep newY:;
run;
```

You can use the PREDDIST statement in PROC MCMC to do posterior prediction.

Prediction Band



Some Observations

- The prediction band is highly sensitive to the prior choice on shrinkage parameter ($\sigma^2_{\gamma_t}$).

Some Observations

- The prediction band is highly sensitive to the prior choice on shrinkage parameter ($\sigma^2_{\gamma_t}$).
  - Possibly an indication that there is not enough information in this data set to estimate that variance.

Some Observations

- The prediction band is highly sensitive to the prior choice on shrinkage parameter ($\sigma^2_{\gamma_t}$).
    - Possibly an indication that there is not enough information in this data set to estimate that variance.
    - There are three levels of variability here: σ^2_y, $\sigma^2_{\gamma_0}$, and $\sigma^2_{\gamma_t}$. Can be just one too many.

Some Observations

- The prediction band is highly sensitive to the prior choice on shrinkage parameter ($\sigma^2_{\gamma_t}$).
    - Possibly an indication that there is not enough information in this data set to estimate that variance.
    - There are three levels of variability here: σ^2_y, $\sigma^2_{\gamma_0}$, and $\sigma^2_{\gamma_t}$. Can be just one too many.
- You can also consider using a truncated prior on β_t:

$$\beta_t \sim \mathrm{N}(0, 10^6) \cdot I_{(\beta_t < 0)}$$

which ensures a negative slope.

Some Observations

- The prediction band is highly sensitive to the prior choice on shrinkage parameter ($\sigma^2_{\gamma_t}$).
    - Possibly an indication that there is not enough information in this data set to estimate that variance.
    - There are three levels of variability here: σ^2_y, $\sigma^2_{\gamma_0}$, and $\sigma^2_{\gamma_t}$. Can be just one too many.
- You can also consider using a truncated prior on β_t:

$$\beta_t \sim \mathrm{N}(0, 10^6) \cdot I_{(\beta_t < 0)}$$

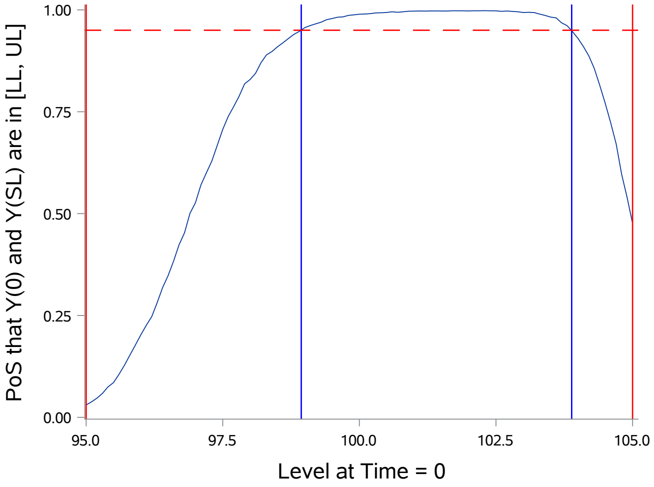
which ensures a negative slope.
Specification using PROC MCMC:

```
   prior bT ~ n(0, sd=1e6, upper=0);
```

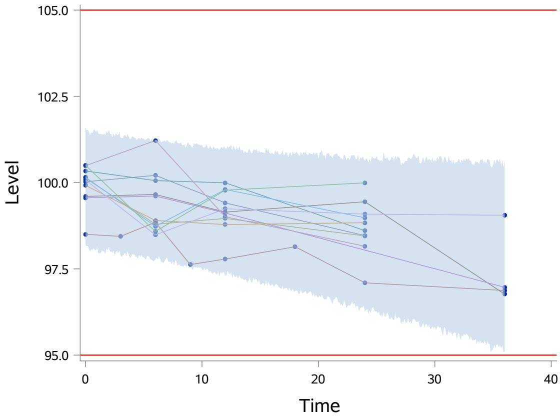
Not much impact on the posterior distribution in this example.

Step 3: Estimate PoS Curve

Next we want to estimate Probability of Success (PoS):

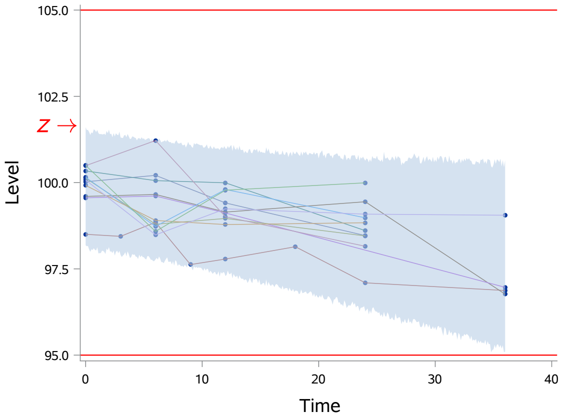


Step 3: Estimate PoS Curve



Step 3: Estimate PoS Curve

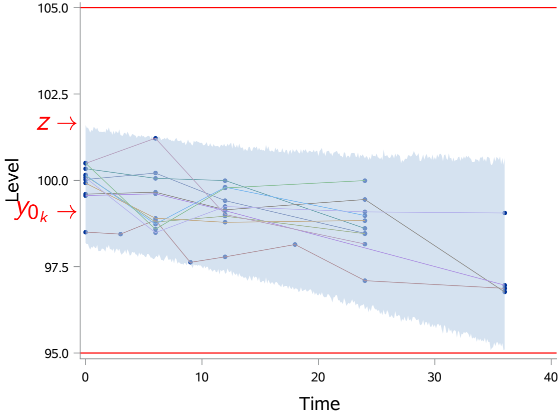
(1) Suppose that a true value at time 0 is z.



Step 3: Estimate PoS Curve

(1) Suppose that a true value at time 0 is z.

(1a) draw $y_{0_k} \sim \mathrm{N}(0, \sigma^2_{y_k})$, an "observed" value conditional on a posterior sample.

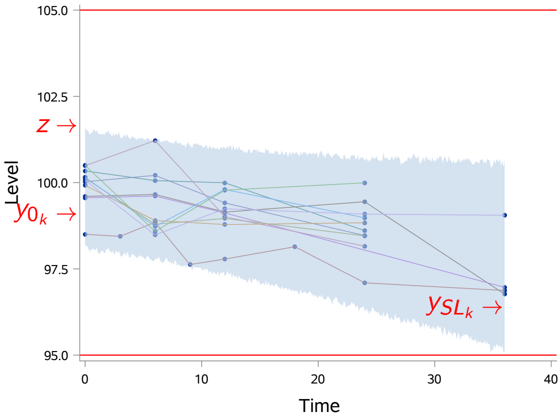


Step 3: Estimate PoS Curve

(1) Suppose that a true value at time 0 is z.

(1a) draw $y_{0_k} \sim \mathrm{N}(0, \sigma^2_{y_k})$, an "observed" value conditional on a posterior sample.

(1b) predict y_{SL_k}, in or out of [LL, UL]



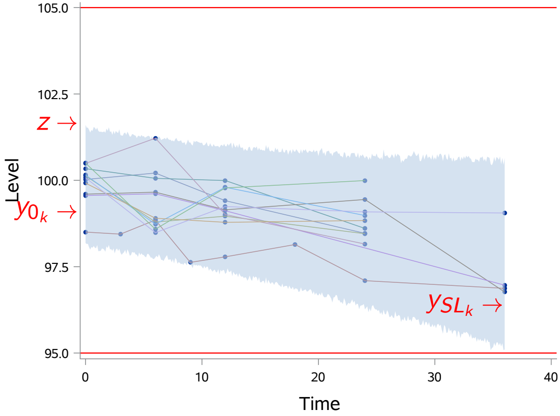
Step 3: Estimate PoS Curve

(1) Suppose that a true value at time 0 is z.

(1a) draw $y_{0_k} \sim \mathrm{N}(0, \sigma^2_{y_k})$, an "observed" value conditional on a posterior sample.

(1b) predict y_{SL_k}, in or out of [LL, UL]

Repeat (1a/1b) using all posterior samples to get $\Pr(y_0, y_{SL} \in [LL, UL])$.



Step 3: Estimate PoS Curve

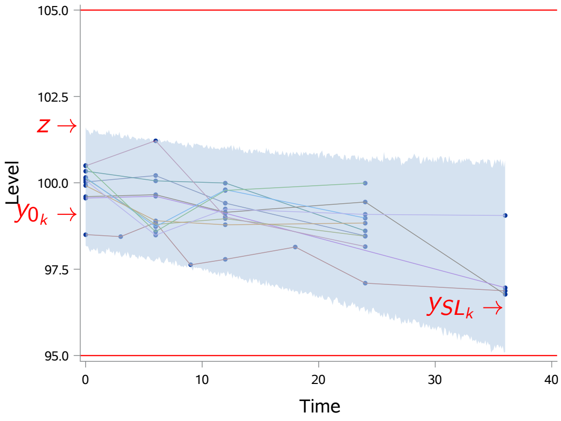
(1) Suppose that a true value at time 0 is z.

(1a) draw $y_{0_k} \sim \mathrm{N}(0, \sigma^2_{y_k})$, an "observed" value conditional on a posterior sample.

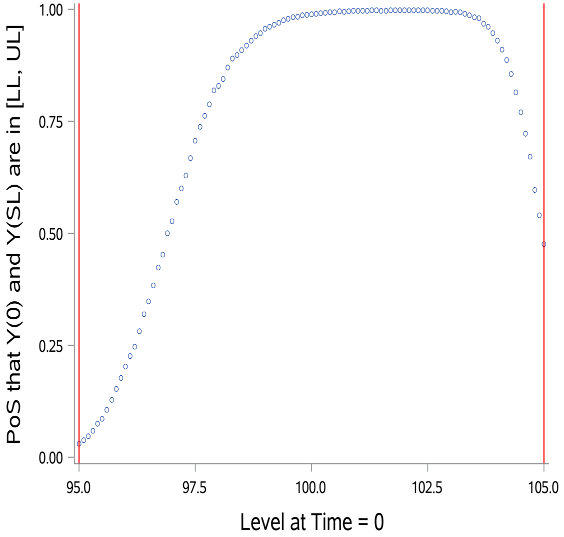
(1b) predict y_{SL_k}, in or out of [LL, UL]

Repeat (1a/1b) using all posterior samples to get $\Pr(y_0, y_{SL} \in [LL, UL])$.

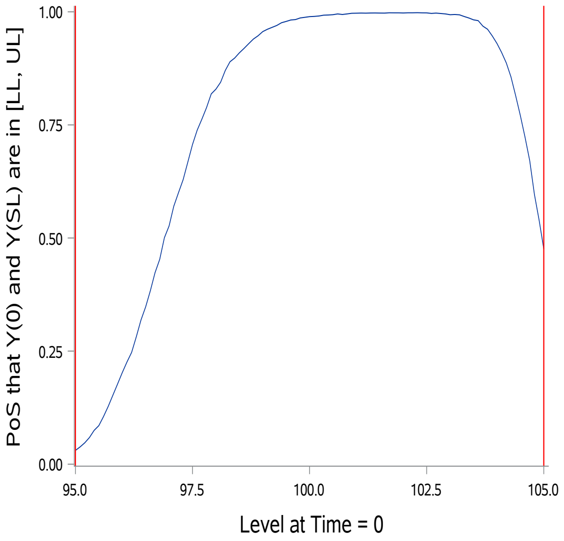
Repeat this process over a grid values of z.



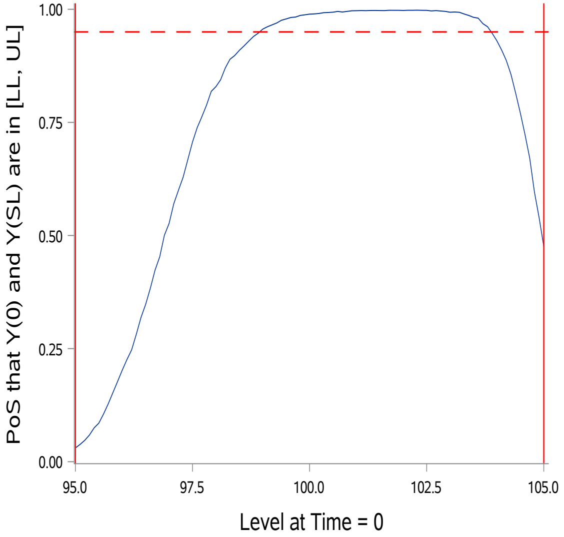
You get the following scatter plot of z vs PoS:



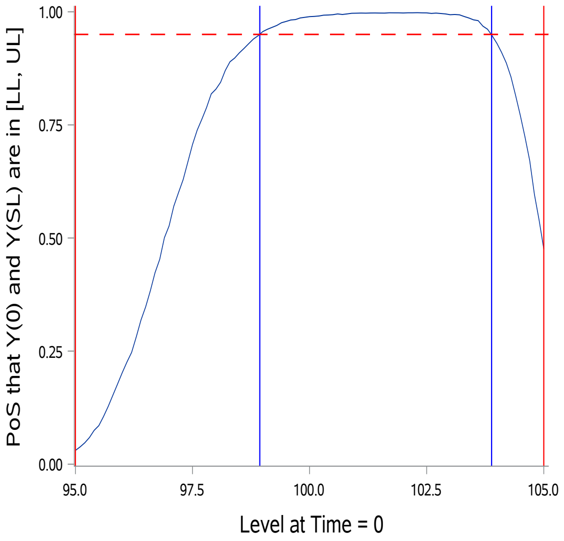
Fit a spline to get a curve



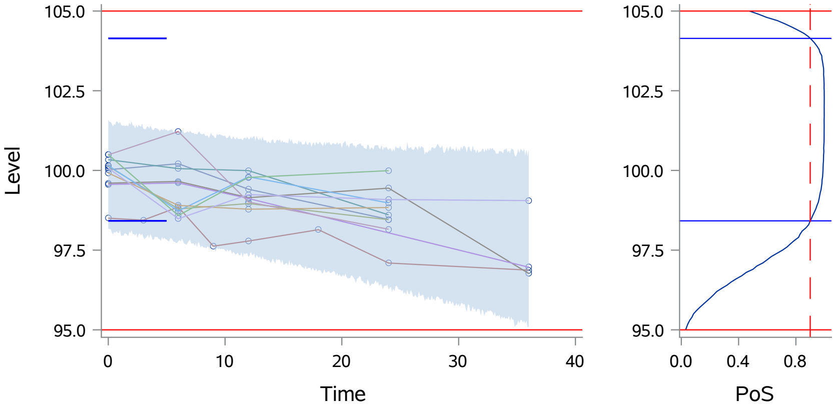
Find the intercept points with the 95% line



You get IRL_{lower} and IRL_{upper}.



End Result



SAS Program

```
data oz;
   call streaminit(10701);
   do z = &ll to &ul by 0.1; /* loop over a find grid */
      PoS = 0;
      do i = 1 to nobs;
         set irlOut nobs=nobs point=i; /* OUTPOST= data set */
         y_0 = rand("normal", z, sqrt(s2y));
         mn = y_0 + rand("normal", bt, sqrt(s2gt)) * 36;
         y_SL = rand("normal", mn, sqrt(s2y));
         success = (y_0 < &UL and y_0 > &LL) and
            (y_dSL < &UL and y_dSL > &LL);
         PoS = PoS + success/nobs;
         end;
      output;
      end;
   stop;
   keep z PoS;
run;
```

You can use PROC SGPLOT to fit a spline to the data:

```
proc sgplot noautolegend data=oz;
   ods output sgplot=sg;
   pbspline y=PoS x=z / nomarkers maxpoints=5000;
run;
```

You can use PROC SGPLOT to fit a spline to the data:

```
proc sgplot noautolegend data=oz;
   ods output sgplot=sg;
   pbspline y=PoS x=z / nomarkers maxpoints=5000;
run;
```

The rest of the program is fairly straightforward.

Thoughts on Prediction

The model used here is a simple random-effects model.

Thoughts on Prediction

The model used here is a simple random-effects model.

Technically, the observed values at y_0 do not impact the distribution of y at TIME=DL – they are conditionally independent.

Thoughts on Prediction

The model used here is a simple random-effects model.

Technically, the observed values at y_0 do not impact the distribution of y at TIME=DL – they are conditionally independent.

The prediction replaces the random intercept mean (β_0) with the value of y_0:

$$\begin{aligned} y_{ij} &\sim \mathrm{N}(\mu_{ij}, \sigma_y^2) \\ \mu_{ij} &= \gamma_{0,j} + \gamma_{t,j} \cdot TIME_{ij} \\ \gamma_{0,j} &\sim \mathrm{N}(\beta_0, \sigma_{\gamma 0}^2) \end{aligned}$$

Thoughts on Prediction

The model used here is a simple random-effects model.

Technically, the observed values at y_0 do not impact the distribution of y at TIME=DL – they are conditionally independent.

The prediction replaces the random intercept mean (β_0) with the value of y_0:

$$\begin{aligned} y_{ij} &\sim \mathrm{N}(\mu_{ij}, \sigma^2_y) \\ \mu_{ij} &= \gamma_{0,j} + \gamma_{t,j} \cdot TIME_{ij} \\ \gamma_{0,j} &\sim \mathrm{N}(\beta_0, \sigma^2_{\gamma 0}) \end{aligned}$$

At TIME=0, y_0 is a reasonable value to use as a plug in for β_0:

```
y_0 = rand("normal", z, sqrt(s2y));
mn = y_0 + rand("normal", bt, sqrt(s2gt)) * 36;
y_SL = rand("normal", mn, sqrt(s2y));
```

Thoughts on Prediction

The model used here is a simple random-effects model.

Technically, the observed values at y_0 do not impact the distribution of y at TIME=DL – they are conditionally independent.

The prediction replaces the random intercept mean (β_0) with the value of y_0:

$$\begin{aligned} y_{ij} &\sim \mathrm{N}(\mu_{ij}, \sigma^2_y) \\ \mu_{ij} &= \gamma_{0,j} + \gamma_{t,j} \cdot TIME_{ij} \\ \gamma_{0,j} &\sim \mathrm{N}(\beta_0, \sigma^2_{\gamma 0}) \end{aligned}$$

At TIME=0, y_0 is a reasonable value to use as a plug in for β_0:

```
y_0 = rand("normal", z, sqrt(s2y));
mn = y_0 + rand("normal", bt, sqrt(s2gt)) * 36;
y_SL = rand("normal", mn, sqrt(s2y));
```

This gives a "pseudo-"conditional prediction model for y at TIME=SL.

Alternatives

An alternative is to fit a repeated measurements model, which models y_0 and any y_t jointly.

Alternatives

An alternative is to fit a repeated measurements model, which models y_0 and any y_t jointly.

The data are unbalanced:

```
    y0      y3       y6       y9    y12     y18     y24     y36   Batch

100.02       .   100.21       .   99.41       .   98.46       .   V2_0
 98.50   98.44    98.85   97.62   97.78   98.14   97.09   96.87   V2_1
100.33       .   100.05       .   99.99       .   98.60       .   V2_10
 99.60       .    99.65       .   99.14       .   99.44   96.76   V2_2
...
```

Alternatives

An alternative is to fit a repeated measurements model, which models y_0 and any y_t jointly.

The data are unbalanced:

```
  y0       y3       y6       y9     y12      y18      y24      y36   Batch

100.02      .    100.21      .     99.41      .     98.46      .      V2_0
 98.50   98.44    98.85    97.62   97.78    98.14   97.09    96.87   V2_1
100.33      .    100.05      .     99.99      .     98.60      .      V2_10
 99.60      .     99.65      .     99.14      .     99.44    96.76   V2_2
...
```

While you can use PROC MCMC to fit this type of data, it is much easier to do so with PROC BGLIMM.

Unbalanced Repeated Measurements Model in PROC BGLIMM

```
proc bglimm data=irl;
   class time batch;
   model level = / noint;
   random int / subject = batch;
   repeated int /subject = time type=un;
run;
```

Random intercept model (11 batches) with repeats in time (8 time points).

Finishing Thoughts

- These development represents part of SAS' continuing effort to add Bayesian capability to the software.

Finishing Thoughts

- These development represents part of SAS' continuing effort to add Bayesian capability to the software.
- We are interested in
  - establishing knowledge base and providing how-tos to our users who are interested in Bayesian design and clinical trials
  - identifying key areas where we can make improvements and enhancements to our procedures, e.g. PROC MCMC, PROC BGLIMM, etc

Finishing Thoughts

- These development represents part of SAS' continuing effort to add Bayesian capability to the software.
- We are interested in
  - establishing knowledge base and providing how-tos to our users who are interested in Bayesian design and clinical trials
  - identifying key areas where we can make improvements and enhancements to our procedures, e.g. PROC MCMC, PROC BGLIMM, etc
- Give it a try in SAS University Edition, which is free to anyone who wishes to learn
  - Base SAS, SAS/STAT, SAS/IML, and part of SAS/ETS
  - Most recent release
  - `www.sas.com/en_us/software/university-edition.html`