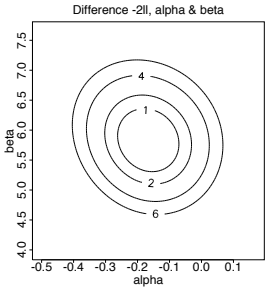
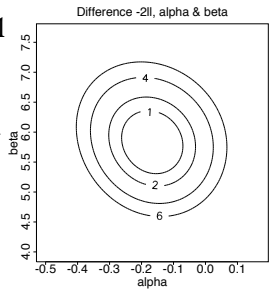
a bit on
Markov Chain Monte Carlo (MCMC)
using the Constant Method example from
Bock & Jones, 1968

The goal of (Bayesian) MCMC estimation is to compute the mean [as opposed to the (frequentist) ML mode] of the posterior (likelihood for an uninformative prior), as an estimate of the unknown parameter(s).



For this example, MCMC methods are not necessary. The R code that created the graphic below was based on the computation of 10,000 values of the likelihood in a 100 x 100 grid.

2-dimensional rectangular quadrature could be used to compute the mean.



Using 2-dimensional rectangular quadrature ("grid integration") to compute the mean:

The likelihood for the observations is

$$L = \prod_{j=1}^{n} \frac{N_{jc}!}{r_{jc}!(N_{jc} - r_{jc})!} P_{jc}^{r_{jc}} Q_{jc}^{(N_{jc} - r_{jc})}$$

where

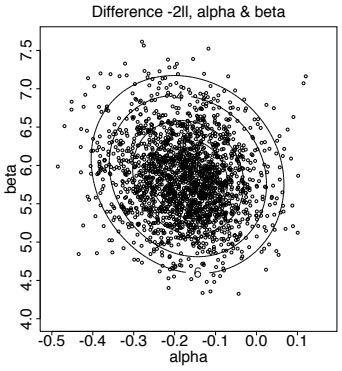
$$P_{jc} = \Phi(\alpha + \beta x_{jc})$$

So for a grid of points

$$\hat{\boldsymbol{\mu}}_{\alpha,\beta} = \frac{\sum_Q L_{\alpha_q,\beta_q} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}_q}{\sum_Q L_{\alpha_q,\beta_q}}$$

(Like the computation of the IRT EAP; this can be followed up with a parallel computation for SD)

MCMC computes the mean using a random sample of points drawn from the posterior (likelihood here).



Unlike direct quadrature, this will work for many more parameters. The trick is to get random draws from the likelihood.

Ways to draw random numbers from "unfamiliar" densities (may) use Gibbs sampling (Johnson & Albert, p. 58ff):

1) Partition the parameter set to produce a list of conditional densities:

$$\begin{gathered} p_1(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_r, \text{data}) \\ p_2(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1, \boldsymbol{\theta}_3, \ldots, \boldsymbol{\theta}_r, \text{data}) \\ \vdots \\ p_r(\boldsymbol{\theta}_r|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{r-1}, \text{data}) \end{gathered}$$

(Note that theta may represent an individual parameter or a set of parameters.)

Then, for the jth iteration:

1. Generate $\boldsymbol{\theta}_1^{(j)}$ from $p_1(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2^{(j-1)}, \ldots, \boldsymbol{\theta}_r^{(j-1)}, \text{data})$
2. Generate $\boldsymbol{\theta}_2^{(j)}$ from $p_2(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1^{(j)}, \boldsymbol{\theta}_3^{(j-1)}, \ldots, \boldsymbol{\theta}_r^{(j-1)}, \text{data})$

   $\vdots$

r. Generate $\boldsymbol{\theta}_r^{(j)}$ from $p_r(\boldsymbol{\theta}_r|\boldsymbol{\theta}_1^{(j)}, \ldots, \boldsymbol{\theta}_{r-1}^{(j)}, \text{data})$

At the end of each iteration one has a random draw from the (intractable, full) distribution. The draws are autocorrelated, but that is handled with "thinning" or some statistical adjustment.

The idea is to make the conditional distributions "easy to draw from" (aka, what's already programmed).

There are many ways to do this (subdivide) for any problem.

For today, only "Data Augmentation" (DA) Gibbs:

The trick here is to augment the data with random draws of Z (what Bock & Jones call Υ) in a two-step Gibbs sampler.

For DA Gibbs for the probit model as per Johnson & Albert (section 3.3), re-write the model:

$$P_i = \Phi(\mathbf{x}_i'\boldsymbol{\beta})$$

$$\mathbf{x}_i' = \begin{bmatrix} 1 & x_i \end{bmatrix} \qquad \boldsymbol{\beta} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

$$Z_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$\text{response}_i = \begin{cases} 0 & \text{if} \quad Z_i \leq 0 \\ 1 & \text{if} \quad Z_i > 0 \end{cases}$$

DA Gibbs for the probit model:

1. Draw Zs from truncated normal distributions:

$$Z_i|\boldsymbol{\beta}, \text{response}_i \propto \begin{cases} \phi(z : \mathbf{x}_i'\boldsymbol{\beta}, 1)I(z \leq 0) & \text{if} \quad \text{response}_i = 0 \\ \phi(z : \mathbf{x}_i'\boldsymbol{\beta}, 1)I(z > 0) & \text{if} \quad \text{response}_i = 1 \end{cases}$$

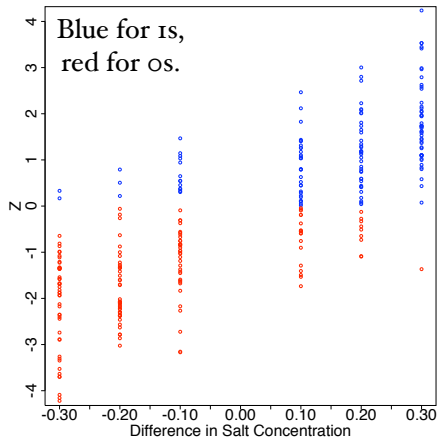
2. Draw $\boldsymbol{\beta}$s from a bivariate normal distribution:

$$\boldsymbol{\beta}|\mathbf{Z} \sim N(\hat{\boldsymbol{\beta}}, \mathbf{X'X}^{-1})$$

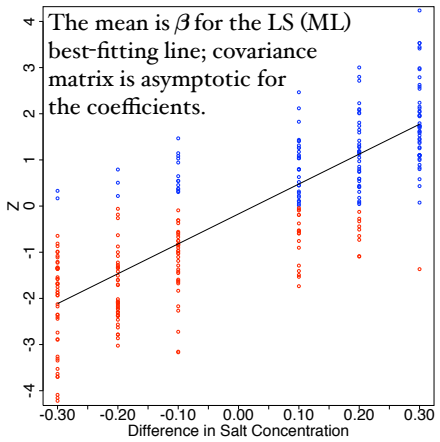
where

$$\hat{\boldsymbol{\beta}} = \mathbf{X'X}^{-1}\mathbf{X'Z}$$

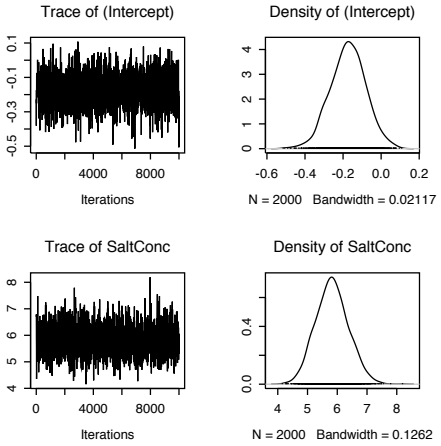
1. Draw Zs from truncated normal distributions:



2. Draw $\boldsymbol{\beta}$s from a bivariate normal distribution:



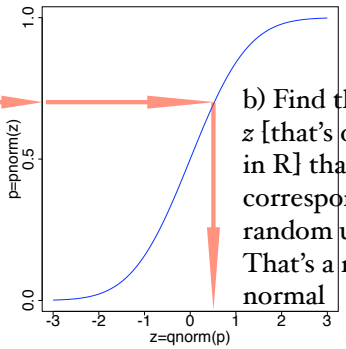
Standard output graphics for MCMC:



Tricks about drawing random numbers, I

A way (one of many) to get random normal deviates:

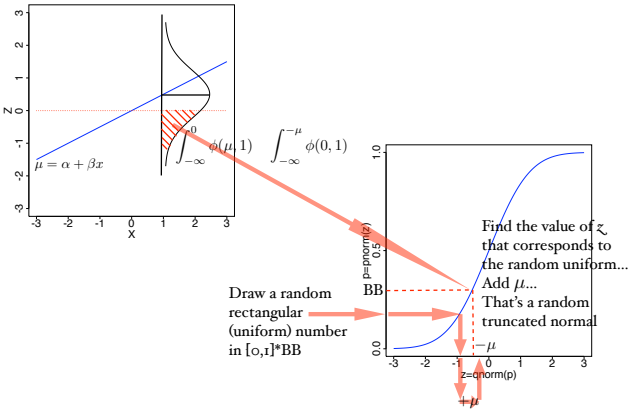
a) Draw a random rectangular (uniform) number in [0,1]



b) Find the value of *z* [that's qnorm(*p*) in R] that corresponds to the random uniform... That's a random normal

Tricks about drawing random numbers, II

A way to get random truncated-normal deviates:



Tricks about drawing random numbers, III

Multivariate normal deviates:

$$\mathbf{y} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Compute

$$\mathbf{y} = \mathbf{S}\mathbf{z}$$

where $\mathbf{z}$ is a vector of independent random normal draws and $\mathbf{S}$ is the Cholesky factor of Σ

Notes:
$$\mathbf{y}'\boldsymbol{\Sigma}^{-1}\mathbf{y}$$
$$\mathbf{y}'\mathbf{S}^{-1}(\mathbf{S}^{-1})'\mathbf{y}$$
$$\mathbf{z}'\mathbf{z}$$
$$\mathbf{z} = (\mathbf{S}^{-1})'\mathbf{y}$$
$$\mathbf{S}'\mathbf{z} = \mathbf{S}'(\mathbf{S}^{-1})'\mathbf{y}$$
$$\mathbf{S}'\mathbf{z} = \mathbf{y}$$

The above is in R's layout. See p. 88 in Bock's (original) chapter 2, but note that R's $\mathbf{S}$ is Bock's $\mathbf{S}'$.