疑似乱数とは

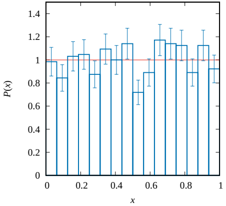
- 計算機でプログラムに従って生成する乱数 (のようなもの)
- 乱数は何に役立つか？
  - 等式のチェック、例外の発見
  - 初期値にランダムネスを入れることで最悪の場合を避ける
  - サンプリングを使ったシミュレーション (→計算機実験 II)
- 乱数を使う場合の注意
  - 計算式に従って生成するため周期は有限であり、必ず何らかの相関がある
  - 初期化 (種の設定) を正しく行う
  - 実際にそれらしい乱数が生成されているか目で見て確認する
- 代表的な乱数発生器のひとつ: メルセンヌ・ツイスター
  - 周期 $2^{19937}-1$、高速、日本製!
  - ヘッダファイル: mersenne_twister.h
  - サンプルプログラム: random.c

様々な分布

- 乱数発生器は通常、一様な整数乱数あるいは実数乱数を生成
- 一様分布以外の分布にしたがう乱数の発生方法の代表例
- 逆関数法
  - 確率分布関数 $F(x)$ の逆関数 $F^{-1}(y)$ と (0,1) の一様乱数 u から $v = F^{-1}(u)$
  - 例: 指数分布 $p(x) = \frac{1}{\mu}e^{-x/\mu}$
    $F(x) = 1 - e^{-x/\mu} \qquad F^{-1}(y) = -\mu \log(1-y)$
  - 一般の確率分布関数について逆関数を求めるのは困難
- 棄却法
  - 確率密度関数を完全に囲むような箱を用意し、その箱の中で一様乱数を生成
  - 確率密度関数の下側の点が生成されたら、その x 座標を乱数として採用。上側の点の場合には再度生成
  - もとの確率密度関数よりも箱が大きくなりすぎると非効率

ヒストグラムの作り方

- 連続変数 (実数) のデータの場合 ([] 内はサンプルプログラムでの変数名)
  - N: サンプル数 [samples]
  - x_{min}: データの最小値 (カットオフ) [xmin]
  - x_{max}: データの最大値 (カットオフ) [xmax]
  - n: ビンの個数 [bins]
  - Δ: ビンの幅 ($\Delta = (x_{\mathrm{max}} - x_{\mathrm{min}})/n$) [dx]



- サイズ n の配列を準備
  - データ毎にどのビンに入るか計算: $j = (x - x_{\mathrm{min}})/\Delta$
  - (必要に応じて) $0 \leq j < n$ であることを確認 (範囲外のデータは無視する)
  - 配列の j 番目の値を 1 増やす
- サンプルプログラム: histogram.c
  (コンパイルには mersenne_twister.h と cmatrix.h が必要)

ビンの個数の設定

- 最適の幅というものはない
- 個数を増やすと表現の自由度は増えるが、各ビンのエラーバーが大きくなる
  - データが統計的に独立である場合、それぞれのビンのカウント数 m はポワソン分布に従う
  - 統計誤差 $\sim \sqrt{m}$
- いくつかの方法・公式が提案されているが、分布の形によっては不適切な場合も
  - スタージェスの公式 $n = \log_2 N + 1$
  - スコットの公式 $\Delta = 3.5\sigma/N^{1/3}$
- 実際には、ビンの個数を何通りか試してみるのが良い
- データを取り直すことが出来ない and/or コストがかかる場合も多いので、生データはいったんファイルに保存しておく