Stan中級編

@simizu706

自己紹介

- 清水裕士
  - 関西学院大学社会学部
- 専門
  - 社会心理学，グループ・ダイナミックス
- 趣味
  - Stan
  - 統計ソフトウェアの開発
- Web
  - Twitter: @simizu706
  - Webサイト: http://norimune.net

今日の主役は

- 今、もっとも開発が熱いMCMC用フリーソフト

中級編

- 中級というのは心理学者のレベルの話
  - 一般的な意味でのStanユーザーのレベルかどうかはわかんない
    - たぶん，これを初級と呼ぶ人もいるとは思う
  - みんながずっと初心者と言い続けるのはその分野が枯れる原因になるので、さっさとみんな中級者を名乗ろう
- 一般に最尤法で解けるレベルが中級と定義
  - ベイズじゃないとできないレベルは上級にしておく
  - 初級は最小二乗法レベル

先に初心者入門を見てください

- http://www.slideshare.net/simizu706/stan-62042940

中級編のメニュー

- 統計モデルを書く
  - まずはモデル式を書けるようになろう
- 一般化線形モデル
  - ベルヌーイ分布，ポアソン分布，二項分布，対数正規分布，ガンマ分布，ベータ分布，負の二項分布，ベータ二項分布
- 階層（分布）モデル
  - 過分散の推定
  - 階層線形モデル
- 潜在変数が含まれるモデル
  - 因子分析
- 尤度関数に構造がある場合のモデル
  - ゼロ過剰分布
  - 混合分布モデル

Stanで統計モデリング

Rでstanを使うための準備

- Rをインストール
  - 必須
- Rstudioをインストール
  - 必須ではないが必須だと思ってもらって構わない
- rstanパッケージをインストール
  - OSによって必要なツールが違うのでリンク先を見て環境に合わせて準備してね
  - https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started-%28Japanese%29

Stan2.10で大幅バージョンアップ

- 一部の文法が非推奨に
  - 代入を意味する<- の代わりに=が推奨される
  - increment_log_prob()の代わりにtaget+=
  - 確率分布_log()の代わりに_lpdf()かlpmf()

- 本合宿では2.11をベースに
  - 2.10にはバグがあるみたいなので
  - 中級編ではこれらの変更が重要

ベイズ統計モデリング

統計モデリングとは

- 確率分布でモデルを作ること
  - 確率分布によって表現された，データ生成についてのモデルを確率モデルと呼ぶ
- 真の分布と予測分布
  - データ生成メカニズムを真の分布と呼ぶ
  - 真の分布を近似する確率分布を予測分布と呼ぶ
  - 統計モデリングは，真の分布を十分近似できる予測分布を作ることが目的

真のモデル＝データ発生メカニズム

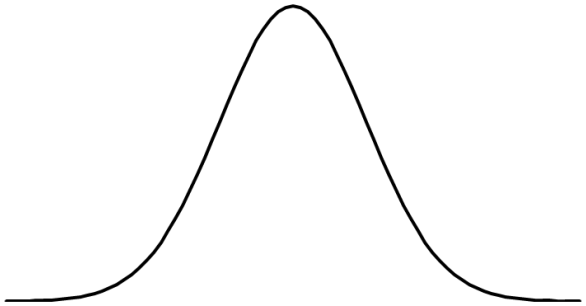
直接は知りえない

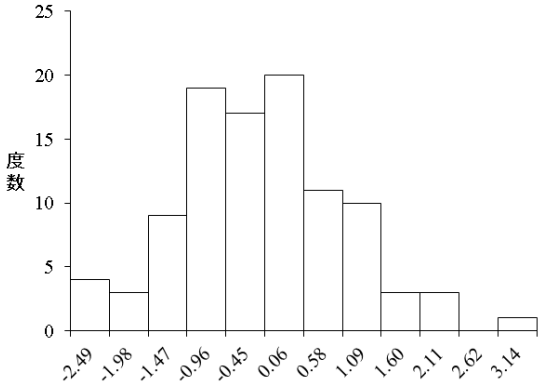
データの発生

直接観測できる

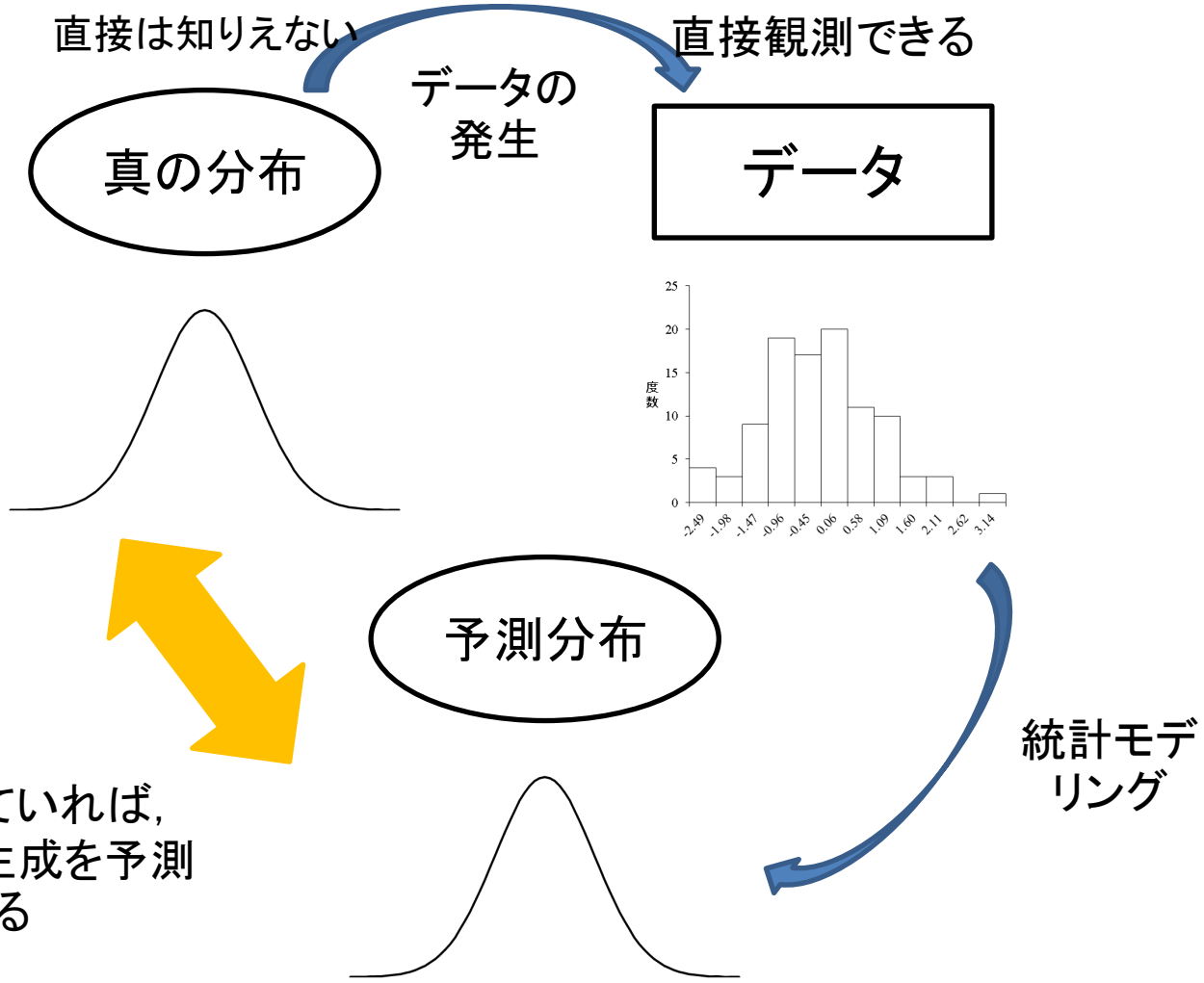
真の分布

データ





予測分布≒データ発生メカニズム



確率モデルと事前分布

- 確率モデル
  - データの生成メカニズムの近似
  - $p(x|\theta)$と表記すると確率分布のこと
    - xはまだ得られていないデータ，θは確率分布のパラメータ
  - $p(X|\theta)$と表記すると尤度関数のこと
    - Xは得られたデータ

- 事前分布
  - パラメータθの確率分布
  - $p(\theta)$と表記する

  - 確率モデル（尤度関数）と事前分布の組が，研究者が想定する「モデル」ということになる

周辺尤度と事後分布

- 周辺尤度
  - $p(X)$のこと
  - $p(X) = \int p(X|\theta)p(\theta)\, d\theta$
  - 「確率モデルと事前分布」が与えられた下での，手元のデータXが得られる確率

- 事後分布
  - データを得た後のパラメータの分布
  - ベイズの定理で求める
  - $p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)} = \frac{\text{尤度関数}*\text{事前分布}}{\text{周辺尤度}}$

予測分布

- 真の分布の近似
  - データを得た後に構成された，データ生成についての確率分布
  - まだ手に入れていないデータをxとすると，
  - $p(x|X)$と表される
    - 事後予測分布ともいう
- 予測分布を作る
  - 確率モデルのパラメータに，推定されたパラメータの事後分布を入れてやればいい
  - $p(x|X) = \int p(x|\theta)p(\theta|X)\, d\theta$
    - 確率モデルを事後分布で重みづけて平均をとったものが，予測分布

真の分布と予測分布

- 真の分布
  - $q(x)$と表す
    - データxの生成メカニズム
- 真の分布に近い予測分布が知りたい
  - $q(x)$を$p(x|X)$で近似する
  - この二つの分布の距離を評価したい
    - カルバックライブラー情報量
    - この期待値をWAICで評価できる

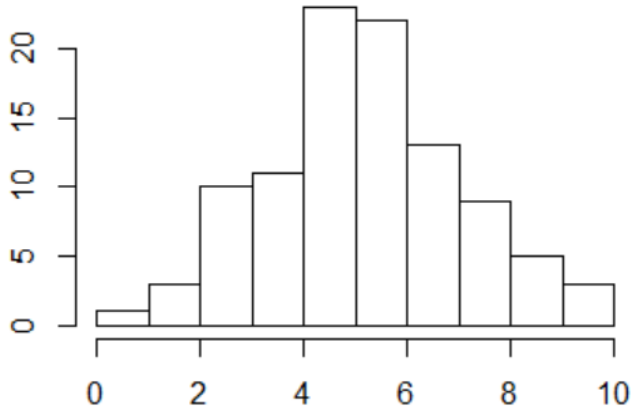
例:正規分布の場合

- 正規分布
  - パラメータは平均μと標準偏差σ
  - $N(\mu, \sigma)$と表記する

- 確率モデルと事前分布
  - $N(x|\mu, \sigma)$と表記する
  - 事前分布は，それぞれのパラメータごとに決める
    - この例では，μは正規分布，σはコーシー分布としておく

データをとってみた

- 真の分布を正規分布とする
  - $q(x) = N(x|\mu = 5, \sigma = 2)$
  - 正規乱数からデータを生成
    - set.seed(123)
    - y <- rnorm(100,5,2)



このデータが生成された
メカニズムは何かな？

確率モデルと事前分布を決める

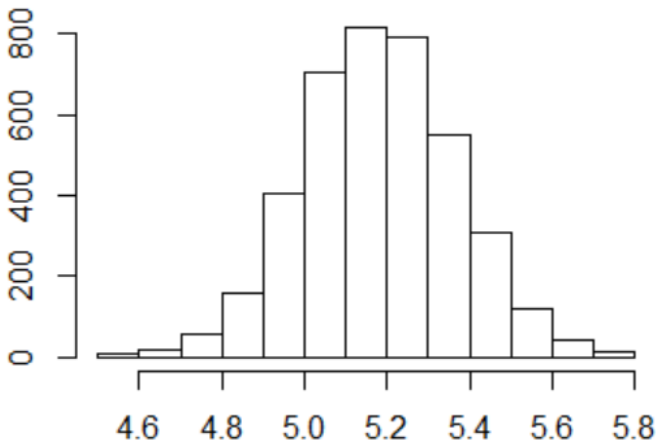
- 確率モデルを決める
  - 理論的，現象的，データの分布などの手がかりから，確率モデルを決める
  - 正規分布以外にもさまざまな分布がありえる
  - 今回は正規分布 $p(X|\mu,\sigma)$

- 事前分布を決める
  - 確率モデルが決まると，推定するべきパラメータが決まる
  - そのパラメータがどういう確率分布に従うかを決める
  - 今回はμは無情報正規分布，σは無情報半コーシー分布
    - $\mu \sim N(0,1000),\ \sigma \sim half\_cauchy(0,5)$

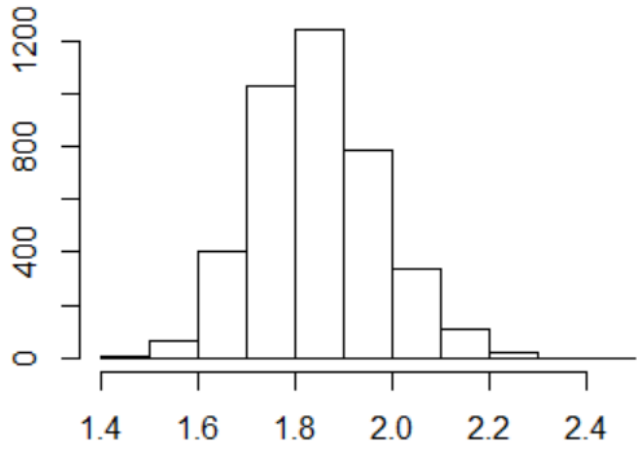
事後分布を推定する

- ベイズ推定を行う
  - 今回はMCMCで求めてみる

- 平均値と*SD*の事後分布



μの事後分布



σの事後分布

分布のパラメータを推定する

- μは要約統計量ではない
  - データを要約するための記述統計量として平均値がよく使われるが・・・
  - 要約統計量と推測統計量は「まったく別物」

- μは真の分布のパラメータ
  - データが生成するメカニズムの要素
    - データを直接要約するものではない
  - 正規分布の場合，要約統計量である算術平均が，μの一致推定量である点がややこしい
    - 分散は要約統計量と推測統計量は導出式が異なる

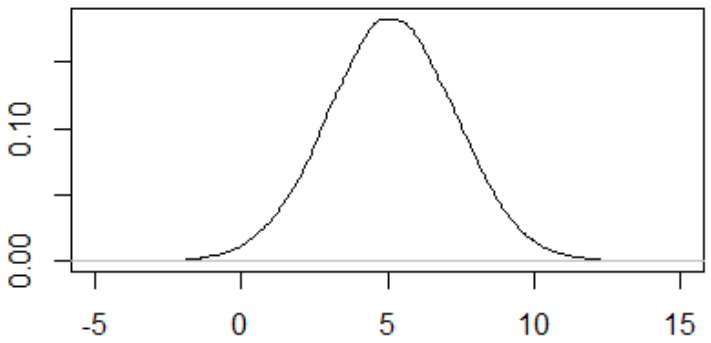
予測分布

- データ生成メカニズムの近似
  - 想定した確率モデル(今回は正規分布)を推定したパラメータの事後分布で重みづけて平均する

- シミュレーションで予測分布を生成

```
mu <- rstan::extract(fit)$mu
sigma <- rstan::extract(fit)$sigma
temp <- rnorm(100000,mu,sigma)
plot(density(temp),xlim=c(-5,15))
```

平均5.18, SD1.85の
正規分布



予測区間

- データがとりうる範囲
  - 95%予測区間：データの値が取りうる95%の範囲
    - 信頼区間とは別物なので注意
    - 信頼区間はパラメータについての範囲

- 予測分布の95パーセンタイル点から計算
  - quantile(temp,probs=c(0.025,0.975))

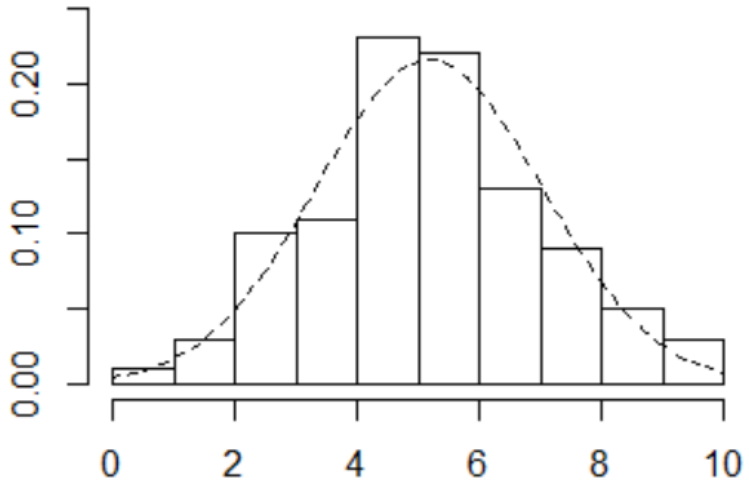
```
> quantile(temp,probs=c(0.025,0.975))
   2.5%   97.5%
1.54233 8.83643
```

  - だいたいデータは1.5～8.8の範囲をとることがわかる

データと合わせてみる

- データと予測分布

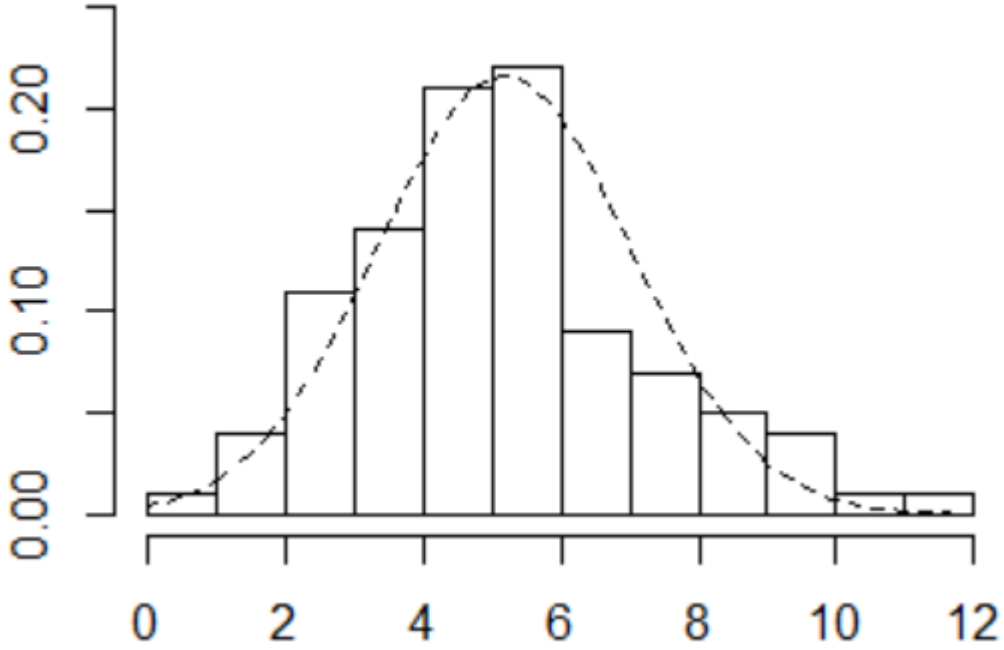
```
tempd <- function(z){
  dnorm(z,5.18,1.85)
}
hist(y,freq=FALSE,ylim=c(0,0.25))
curve(tempd,add=TRUE,lty=2)
```



だいたい合ってる！

同じ真の分布から生成した別データ

- これもまぁだいたい合ってる



統計モデルを自分で書く

統計モデルを書く･･･とは

- 確率モデルと事前分布の両方
  - とりあえず確率モデルを書けるようになろう！
  - 事前分布は階層モデルでなければ簡単

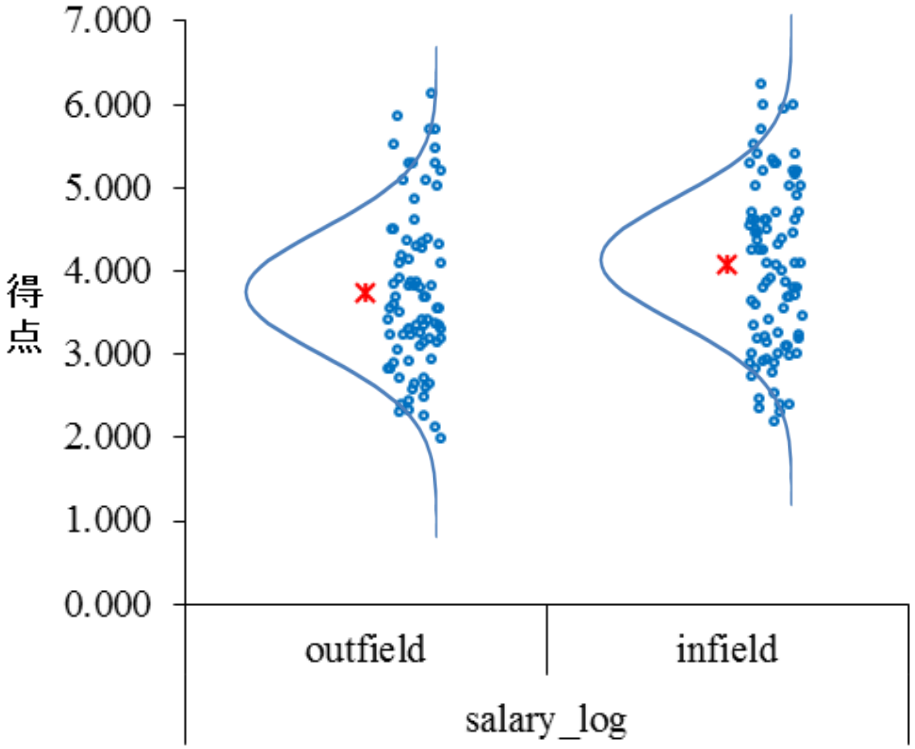
- 重要なのは････
  - あらゆる統計モデリングが，確率分布のパラメータを推定対象としているということ
  - 心理統計学と少し書き方が異なる

平均値の差の推定

- いわゆるt検定
  - 確率モデルとして正規分布を仮定
  - 二つのグループの分散（SD）は等しい

- 確率モデルを書いてみる
  - $y_{1i} \sim N(\mu_1, \sigma)$
  - $y_{2i} \sim N(\mu_1 + \delta, \sigma)$

平均値の差のモデリング



同じSDを持つ二つの正規分布でデータを表現

これを書き直すと・・・

- 回帰モデル的に書いてみる
  - y_{1i}を0，y_{2i}を1とするようなダミー変数$\mathrm{X_i}$を考える
  - そして，y_{1i}の時は平均がμ_1となり，y_{2i}のときは$\mu_1 + \delta$となるようなμを考える
    - $\mu = \mu_1 + \delta\, X_i$

- パラメータに構造を持った正規分布として書ける
  - $y_i \sim N(\mu, \sigma)$
    - $\mu = \mu_1 + \delta\, X_i$

回帰分析

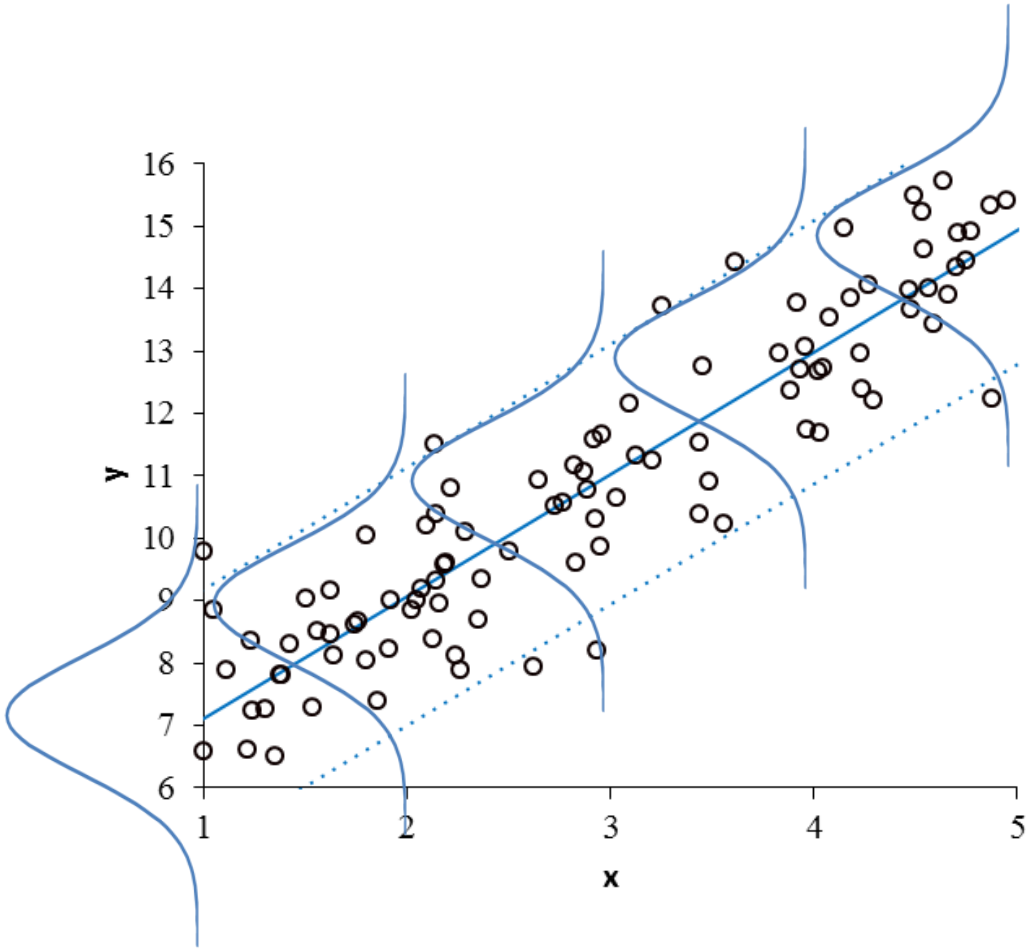
- 回帰分析の仮定
  - 確率モデルは正規分布
  - データはすべて独立に同一の分布から生成

- 回帰分析のモデル
  - $y_i = \alpha + \beta\, X_i + e_i$
  - $e_i \sim N(0, \sigma)$

回帰分析の統計モデル

- データの生成メカニズムを正規分布と仮定
  - $y_i \sim N(\mu_i, \sigma)$
- 平均パラメータμに線形モデルを仮定
  - $\mu_i = \alpha + \beta\ X_i$
- つまり，こういう確率分布となる
  - $y_i \sim N(\alpha + \beta X_i, \sigma)$
    - あとはαとβ，σの事前分布を考えれば良い

平均が$\alpha + \beta X_i$の条件付き正規分布



青い破線は95%予測区間

平均値がXの値によって変わる，条件付き正規分布

$$P(y_i|X_i) = N(y_i|\alpha, \beta, \sigma; X_i)$$

すべてのXの値において，分散が等しい正規分布を仮定
→均一分散の仮定

Stanで回帰分析

- モデルをStanでそのまま書けば良い
  - $y_i \sim N(\alpha + \beta\, X_i, \sigma)$
    - $\alpha \sim N(0,100)$
    - $\beta \sim N(0,100)$
    - $\sigma \sim cauchy(0,5)$

```
model{
  y ~ normal(alpha + beta*x,sigma);
  alpha ~ normal(0,100);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

Stanコード

```
data{
  int N; //サンプルサイズ
  vector[N] y; //目的変数
  vector[N] x; //説明変数
}
parameters{
  real alpha;
  real beta;
  real<lower=0> sigma;
}
model{
  y ~ normal(alpha + beta*x,sigma);
  alpha ~ normal(0,100);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

重回帰分析

- 説明変数が複数になっただけ

$$-\mu = \alpha + \beta_1 X_{1i} + \beta_{2i} X_{2i} + \cdots + \beta_{mi} X_{mi}$$

- 説明変数を行列で定義すると簡単になる

>model.matrix(v1 ~ v2 + v3, dat)

	(Intercept)	v2	v3
1	1	6.4459947	7.9693937
2	1	4.0123892	4.9137136
3	1	4.2399094	3.4056047
4	1	6.1817963	6.2950197
5	1	3.3527517	4.6300356
6	1	6.0085378	5.3836381

モデルを行列で書く

- 行列での表記
  - $\boldsymbol{\mu} = \boldsymbol{X}\,\boldsymbol{\beta}$
  - μは長さNの平均ベクトル，XはN×Mの説明変数行列，βは長さMの回帰係数ベクトル
    - ただし，Nはサンプルサイズ，Mは変数の数
    - またXの1列目はすべての要素が1のベクトル

- 順番に注意！
  - Xについて，行をオブザベーション，列を変数という行列にした場合，Xがさき，βがあとになる
    - 行列演算は，N×Mの行列に長さMのベクトルを書けるためには，左ではなく右から書けないといけない

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  vector[N] y; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
  real<lower=0> sigma;
}
model{
  y ~ normal(x*beta,sigma);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

Stanにおけるサンプリング表記

サンプリングステートメント

- ~を使った表記のこと
  - $y \sim normal(mu, sigma)$;みたいなやつ
  - これは，実は簡略的なコード
    - sampling statement formという
- 内部では，対数確率を足し上げている
  - 対数確率関数をtargetに+=で渡してやることで，modelの中の対数確率がすべて足される
    - 確率だと掛け算，対数確率だと足し算になることに注意
  - target += normal_lpdf(y| mu, sigma);
    - y ~ normal(mu, sigma)と同じ推定結果になる
    - explicit increment formという

lpdfとlpmf

- _lpdf
  - 対数確率密度関数
    - log probability density function
  - 連続分布の場合に，分布名の後につける
    - normal_lpdf()

- _lpmf
  - 対数確率質量関数
    - log probability mass function
  - 離散分布の場合に，分布名の後につける
    - poisson_lpmf()

対数確率分布の書き方

- 確率分布の表記
  - $P(x) = normal(x|\mu, \sigma)$
    - データのあとにバーを書いて，そのあとパラメータ
    - パラメータで条件付けられた確率分布という意味

- Stanでの対数確率関数の表記
  - normal_lpdf(x|mu,sigma);
  - 確率変数の後は，”,”ではなく”|”である点に注意

両者の違い

- sampling statement formの場合
  - ここではサンプリング形式と呼ぶことにする
  - パラメータ推定に関係のない定数は計算しない
    - つまり，確率分布のカーネルのみを計算＝速い
  - 視認性が高い
    - 確率モデルをそのまま表現できる

- explicit increment formの場合
  - ここではターゲット形式と呼ぶことにする
  - 定義された対数確率関数をすべて計算
    - つまり，遅い
  - 推定された_lpが，いわゆるモデルの対数尤度と同じになる
  - これを使わないと計算出来ないモデルもある

回帰分析で比較

- サンプリング形式
  - y ~ normal(x*beta,sigma);
  - beta ~ normal(0,100);
  - sigma ~ cauchy(0,5);

- ターゲット形式
  - target += normal_lpdf(y|x*beta,sigma);
  - target += normal_lpdf(beta|0,100);
  - target += cauchy_lpdf(sigma|0,5);

結果

- サンプリング

	mean	se_mean	sd	2.5%	50%	97.5%	n_eff	Rhat
beta[1]	32.53	0.15	6.47	19.76	32.54	44.91	1884	1
beta[2]	6.74	0.01	0.65	5.47	6.75	7.99	2007	1
sigma	70.92	0.06	3.49	64.46	70.82	78.24	3412	1
lp__	-948.51	0.03	1.24	-951.74	-948.20	-947.09	2249	1

- ターゲット

	mean	se_mean	sd	2.5%	50%	97.5%	n_eff	Rhat
beta[1]	32.47	0.14	6.46	20.03	32.47	45.20	2291	1
beta[2]	6.74	0.01	0.64	5.46	6.74	8.00	2086	1
sigma	70.87	0.06	3.50	64.34	70.77	77.98	3233	1
lp__	-1145.19	0.03	1.28	-1148.54	-1144.85	-1143.76	2243	1

WAICの計算で使う

- WAIC
  - Widely Applicable Information Criterion
  - 広く使える情報量規準
    - 別名Watanabe-Akaike Information Criterion

- 予測分布と真の分布の距離の期待値
  - 真の分布から無限にサイズNのデータを生成させた場合の対数尤度の期待値
  - 真の分布が確率モデルで実現可能でなくても，正則でなくても成立する情報量規準

StanでWAICを計算

- 各データごとの対数尤度を出力する
  - generated quantities{}ブロックで書くことが多い
  - ベクタライズはできない

```
generated quantities{
  vector[N] log_lik;
  for(n in 1:N){
    log_lik[n] = normal_lpdf(y[n]|x[n]*beta,sigma);
  }
}
```

- それをlooパッケージのwaic()に入れる

```
library(loo)
waic(rstan::extract(fit)$log_lik)
```

ロジスティック回帰

2値データを予測

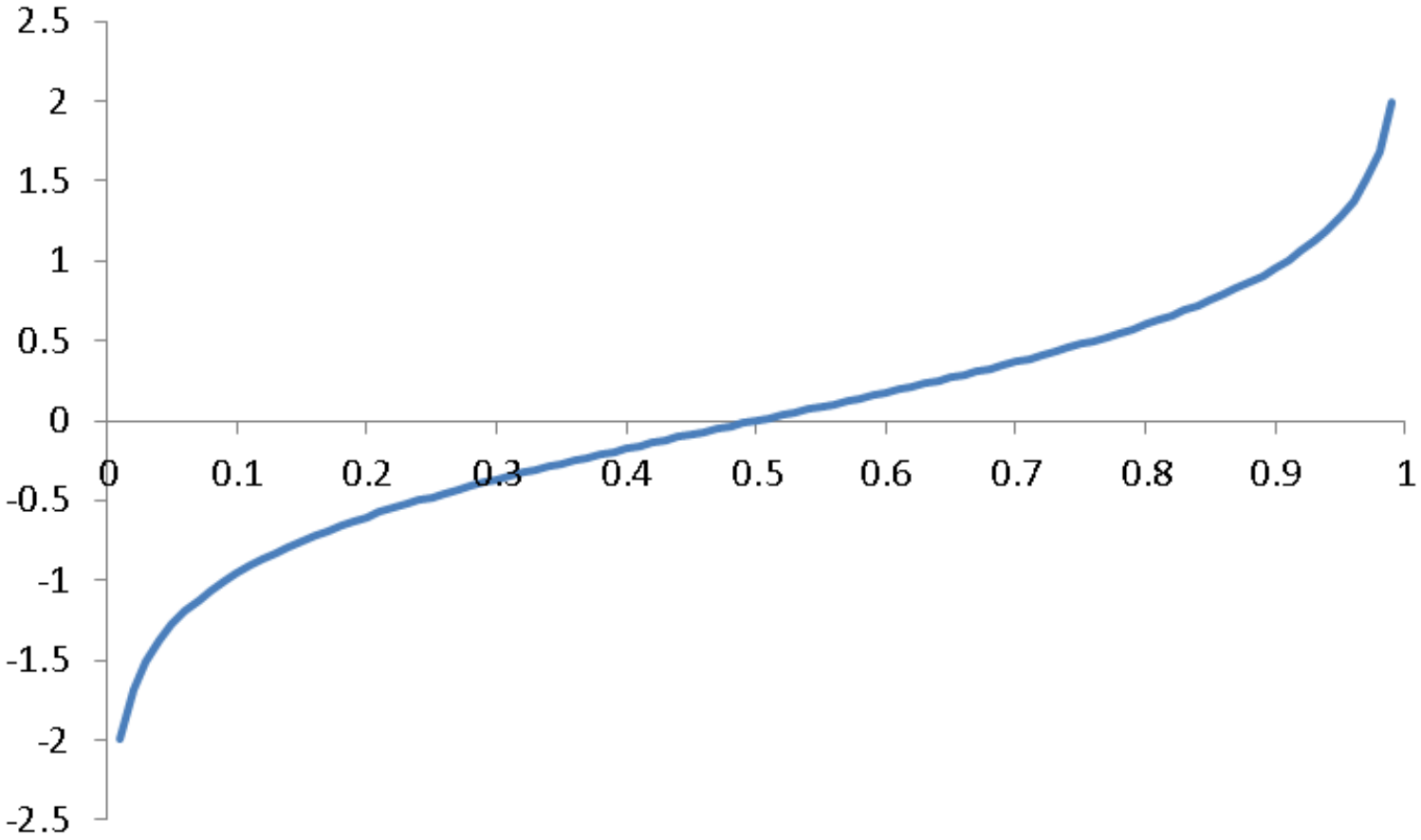
- 0と1のデータの生成メカニズムを考える
  - 予測したいのは1になる確率θ
  - 確率分布はベルヌーイ分布を使う
    - じゃあ「ベルヌーイ回帰」でよくないですか？

- ベルヌーイ分布
  - 試行数が1回だけの二項分布
  - パラメータは成功率θのみ
  - $y_i \sim Bernoulli(\theta_i)$
    - y_iは0と1の2値をとる

θを推定したいが・・・

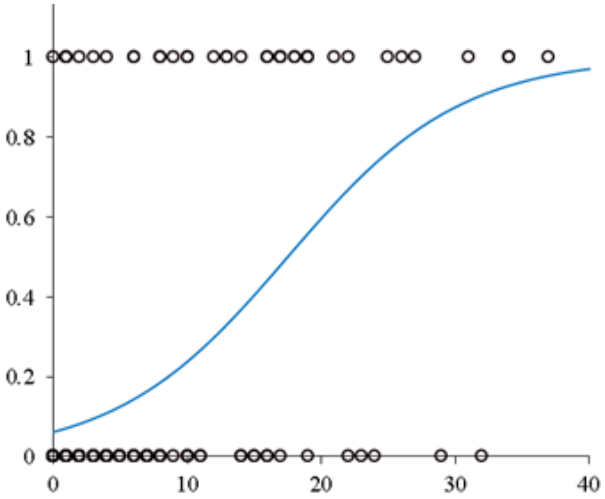
- 成功率θは0~1の範囲しかとらない
  - 線形モデルを仮定した場合，予測値が0~1に収まらないと尤度が計算出来ない
  - 成功率θを直接推定するのではなく，ロジット変換をしたlogit(θ)を推定する
    - logit(θ_i) = log(θ_i /(1-θ_i)) = $\alpha + \beta\ X_i$
- ロジットリンク
  - 0~1の値を，-∞～∞に変換する非線形変換関数

ロジット変換



ロジスティック回帰

- 成功率θをロジット変換する線形モデル
  - 予測が直線ではなく，曲線になる
  - この曲線をロジスティック曲線という
    - だからロジスティック回帰という
  - これをStanで表現する



ロジスティック回帰のモデリング

- 実際にモデリングするときは・・・
  - 成功率θをロジット変換するのではなく
  - 予測値をロジット変換の逆変換を行う
    - つまり，予測値が0～1の範囲になるようにする

- ロジスティック回帰のモデル式は・・・
  - $\theta_i = \mathrm{logit}^{-1}(\alpha + \beta\ X_i)$
    - $\mathrm{logit}^{-1}()$はlogit()の逆関数，つまりロジスティック関数
  - $y_i \sim Bernoulli(\theta_i)$

Stanでの表現

- Stanにはinv_logit()関数がある
  - これをつかって予測値を変換し，ベルヌーイ分布のパラメータとする
  - $\theta_i = inv\_logit(\alpha + \beta\ X_i)$
  - あるいは，$\theta_i = inv\_logit(\boldsymbol{X}_i\boldsymbol{\beta})$
- bernoulli_logit()という確率分布もある
  - bernoulli_logitは，パラメータが$\text{logit}^{-1}(\theta)$であるような確率分布
  - なので，わざわざパラメータを変換しなくて良い
  - しかもこちらのほうが速い

離散分布の注意点

- 目的変数はvectorで宣言できない
  - vectorはreal型を並べたもの
  - 離散分布は整数しかとらないので，int型でしか宣言できない

- int型の配列を使う
  - int y[N];
  - ベルヌーイ以外にも，二項分布，ポアソン分布，負の二項分布などがそれに当てはまる
  - 項目反応理論などでは，int y[N,M]という感じで，2次元配列を使う
    - Mは項目数

Stanコード　θを計算バージョン

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
}
transformed parameters{
  vector[N] theta; //パラメータthetaを宣言
  for(n in 1:N)
    theta[n] = inv_logit(x[n]*beta); //ロジットの逆変換
}
model{
  y ~ bernoulli(theta); //ベルヌーイ分布
  beta ~ normal(0,100);
}
```

目的変数yは離散分布を
仮定するのでint型で宣言

For文は処理が1行だけの場合は
{}を省略することができる

inv_logit()はベクトル化に対応して
いない

Stanコード　θを変換しない

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
}
model{
  y ~ bernoulli_logit(x*beta);
  beta ~ normal(0,100);
}
```

二項分布

試行数が複数の場合

- 例：30回試行して，15回成功した
  - 成功率θを推定したい

- 二項分布
  - $Binomial(y_i) = {}_T C_y \, \theta^{y_i} \, (1-\theta)^{T-y_i}$
  - $y_i \sim Binomial(\mathrm{T}, \theta_i)$
    - ただし，Tは試行数，θは成功率

二項分布のモデリング

- 基本的なモデリングはベルヌーイと同じ
  - $y_i \sim Binomial(T, \theta_i)$
    - ただし，試行数は推定対象ではなく，データから与える
  - $\theta_i = logit^{-1}(X_i\,\beta)$

- Stanでのモデリング
  - ベルヌーイと同じ
    - inv_logit()を使う
  - binomial_logit()という確率分布もある

Stanコード1

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int t[N]; //試行数
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
}
transformed parameters{
  vector[N] theta;
  for(n in 1:N)
    theta[n] = inv_logit(x[n]*beta); //ロジットの逆変換
}
model{
  y ~ binomial(t,theta); //二項分布
  beta ~ normal(0,100);
}
```

Stanコード2

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int t[N];//試行数
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
}
model{
  y ~ binomial_logit(t,x*beta);
  beta ~ normal(0,100);
}
```

ポアソン回帰分析

ポアソン分布

- レアな事象が生起した回数をモデリング
  - カウントデータについての確率分布
    - 非負の整数についての離散分布

- ポアソン分布
  - $Poisson(y_i) = \frac{\lambda^{y_i} e^{-\lambda}}{y_i!}$
  - $y_i \sim Poisson(\lambda)$
    - ただし，λは平均パラメータ
    - 単位時間に平均何回生起するか

ポアソン回帰のモデリング

- 平均パラメータλの推定
  - ただし，λは正の値をとる
  - そこで，対数リンクを用いる
    - log(λ)を線形モデルで推定する

- 確率モデル
  - $y_i \sim Poisson(\lambda_i)$
  - $\log(\lambda_i) = X_i\beta$

Stanでのモデリング

- リンク関数ではなく，線形モデルを変換
  - ロジスティック回帰と同じ
  - log(λ)を予測するのではなく，λをexp()関数に入れた線形モデルで予測する

- poisson_log()という確率分布もある
  - Stanでは，リンク関数を使うよりも，最初からリンク関数が込みになっている分布を使うほうが速い

Stanコード1

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
}
transformed parameters{
  vector[N] lambda; //平均パラメータλを宣言
  for(n in 1:N)
    lambda[n] = exp(x[n]*beta); //対数リンクの逆変換
}
model{
  y ~ poisson(lambda); //ポアソン分布
  beta ~ normal(0,100);
}
```

Stanコード2

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
}
model{
  y ~ poisson_log(x*beta); //ポアソン分布
  beta ~ normal(0,100);
}
```

ガンマ分布

ガンマ分布

- あるレアな出来事がα回生起するまでの時間
  - $y \sim Gamma(\alpha, \lambda)$
    - 平均λのポアソン分布に従う事象が，α回生じるまでにかかる時間についての分布
    - 正の値をとる連続値の分布

$$Gamma(y|\alpha, \lambda) = \frac{\lambda^{\alpha}}{\Gamma(\alpha)} y^{\alpha-1} \exp(-\lambda y)$$

- ガンマ分布の平均と分散
  - 平均＝α/λ
  - 分散=α/λ^2

ガンマ回帰のモデリング

- 平均からλを計算
  - λ=α/平均
  - λは正の値を取るので対数リンクを使う
    - 逆数リンクを使うこともある

- 確率モデル（対数リンクの場合）
  - $y_i \sim gamma(\alpha, \lambda_i)$
  - $\lambda_i = \frac{\alpha}{\exp(X_i\beta)}$

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  vector<lower=0>[N] y; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //回帰係数
  real<lower=0> alpha;//形状パラメータ
}
model{
  vector[N] lambda;//尺度パラメータ
  for(i in 1:N){
    lambda[i] = alpha / exp(x[i]*beta);
  }
  y ~ gamma(alpha, lambda);
  beta ~ normal(0,100);
  alpha ~ cauchy(0,5);
}
```

対数正規分布による回帰

対数正規分布

- 対数をとると正規分布になる分布
  - $\log(y) \sim Normal(\mu, \sigma)$
    - となるような，yについての分布
    - 正の値をとる連続値の分布
  - $Lognormal(y|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma x} \exp(-\frac{(\log(x)-\mu)^2}{2\sigma^2})$
- ガンマ分布に形は似ているが・・
  - ガンマ分布は，待ち時間についての分布
  - 対数正規分布は，ベキの関係を表す分布
    - 年収とか，ネットワークサイズとか

モデリング

- 正規分布と同じモデリング
  - $y_i \sim Lognormal(\mu_i, \sigma)$
  - $\mu_i = X_i\beta$

- StanにはlognormalというStan関数がある
  - 特に何も考えずにそのままできる

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  vector<lower=0>[N] y; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //回帰係数
  real<lower=0> sigma;//sd
}
model{
  y ~ lognormal(x*beta,sigma);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

ベータ分布

ベータ分布

- 確率が従う分布
  - 0~1の範囲を取る連続分布

$$Beta(y|\alpha,\beta) = \frac{1}{\mathrm{B}(\alpha,\beta)}\, y^{\alpha-1}(1-y)^{\beta-1}$$

    - 平均 = $\frac{\alpha}{\alpha+\beta}$，分散 = $\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$

- 使いどころ
  - 二項分布の成功率θの事前分布など
  - データが直接確率で手に入った場合などの回帰分析などで使える
    - もし分母と分子が両方データであるなら，二項分布の方が良い

モデリング

- ベータ分布のパラメータの構造
  - 回帰係数のβと紛らわしいので，ここではaとbとする
  - $y_i \sim Beta(a_i, b_i)$
    - ベータ分布の平均をμ，尺度パラメータをσとすると，
    - $a_i = \mu_i \, \sigma$
    - $b_i = (1 - \mu_i) \, \sigma$

- ベータ回帰分析のモデリング
  - $y_i \sim Beta(a_i, b_i)$
  - $a_i = inv\_logit(X_i \beta) \, \sigma$
  - $b_i = \left(1 - inv\_logit(X_i \beta)\right) \sigma$

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  vector<lower=0,upper=1>[N] y; //0~1の制約をつける
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //回帰係数
  real<lower=0> sigma;//尺度パラメータ
}
model{
  vector[N] a;//ベータ分布のパラメータ
  vector[N] b;//ベータ分布のパラメータ
  for(i in 1:N){
    a[i] = inv_logit(x[i]*beta)*sigma;
    b[i] = (1-inv_logit(x[i]*beta))*sigma;
  }
  y ~ beta(a,b);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

負の二項分布

ポアソン分布の過分散問題

- 離散分布はパラメータが1つのものが多い
  - 二項分布・・・成功率θのみ
  - ポアソン分布・・・平均λのみ

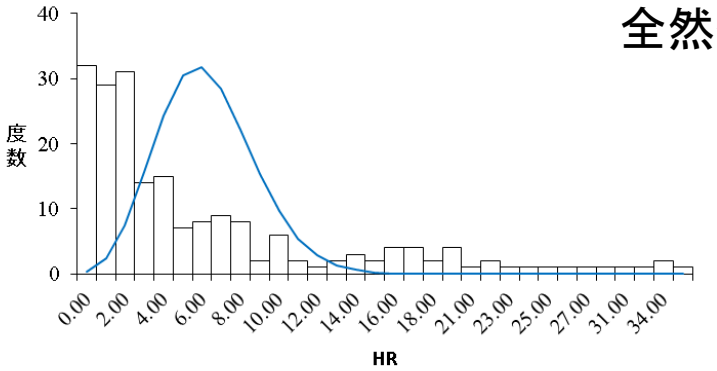
- 分散は自動的に決まってしまう
  - 二項分布・・・分散=$T\theta(1-\theta)$
  - ポアソン分布・・・分散=λ
  - データの分布が分散がより大きい場合，予測が大きく外れてしまう・・・過分散問題

負の二項分布

- ポアソン分布の過分散も推定
  - HRをポアソンと負の二項分布で推定

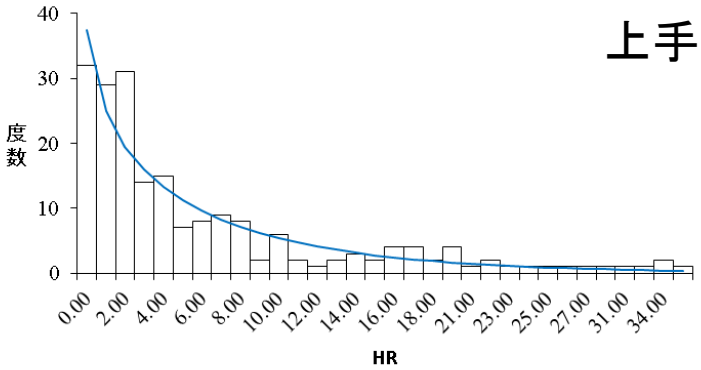
ポアソン分布

全然ダメ！



負の二項分布

上手くいきそう！



負の二項分布

- 二項分布の逆バージョン
  - ベルヌーイ試行において，成功率θでα回成功するまでに必要な失敗回数yについての確率

$$NegBinom(y|\theta,\alpha) = \binom{y+\alpha-1}{y}\theta^{\alpha}(1-\theta)^{y}$$

連続変量に拡張

- ガンマ関数を用いて，成功数αが整数でなくてもいけるように拡張

$$NegBinom(y|\theta,\alpha) = \frac{\Gamma(y+\alpha)}{y!\Gamma(\alpha)}\,\theta^{\alpha}(1-\theta)^{y}$$

  - ガンマ関数
    - $\Gamma(u+1) = u!$

ポアソン分布との関係

- ポアソン分布

  $P(y) = \frac{\lambda^y e^{-\lambda}}{y!}$

  – このとき，λが平均μのガンマ分布に従うと仮定すると・・・

    - $\lambda \sim Gamma\left(\alpha, \frac{\alpha}{\mu}\right)$

- 混合した確率分布は次の負の二項分布になる

$$P(y|\mu,\alpha) = \int_0^\infty Poisson(y|\lambda)Gamma(\lambda|\mu,\alpha)\, d\lambda$$
$$= \frac{\Gamma(y+\alpha)}{y!\Gamma(\alpha)} \left(\frac{\alpha}{\mu+\alpha}\right)^\alpha \left(\frac{\mu}{\mu+\alpha}\right)^y$$

ただし，$\theta = \frac{\alpha}{\mu+\alpha}$

負の二項分布で回帰

- モデリング
  - ポアソンと同様，対数リンクを用いる
  - $y_i \sim negative\_binomial(\mu, \alpha)$
  - $\mu = \exp(X_i \beta)$

- Stanでのモデリング
  - 上記の意味での負の二項分布は，Stanでは neg_binomial_2()という関数名となっている
  - 対数リンクを込みにしたneg_binomial_2_log()もある

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
  real<lower=0> alpha;//尺度パラメータ
}
model{
  y ~ neg_binomial_2_log(x*beta,alpha); //負の二項分布（対数リンク）
  beta ~ normal(0,100);
  alpha ~ cauchy(0,5);
}
```

ベータ二項分布

ベータ二項分布

- 二項分布の過分散を解決する
  - 二項分布のパラメータθに個人差を考慮した分布
    - θがベータ分布にしたがって変動すると仮定
    - 二項分布とベータ分布に混合分布
  - 離散分布なので，yは非負の整数
  - $BetaBinomial(y|T,\alpha,\beta) = \binom{T}{y}\frac{\mathrm{B}(y+\alpha,T-y+\beta)}{\mathrm{B}(\alpha,\beta)}$
  - Tは試行数，αとβはベータ分布のパラメータ
  - B()はベータ関数
    - $\mathrm{B}(u,v) = \frac{\Gamma(u)\Gamma(v)}{\Gamma(u+v)}$

モデリング

- ベータ分布と同様
  - 回帰係数βと紛らわしいので，パラメータをaとbとする
  - $y_i \sim BetaBinomial(T, a_i, b_i)$
    - ベータ分布の平均をμ，尺度パラメータをσとすると，
    - $a_i = \mu_i \, \sigma$
    - $b_i = (1 - \mu_i) \, \sigma$

- Stanでのベータ回帰分析のモデリング
  - $y_i \sim Beta\_Binomial(T, a, b)$
    - $a_i = inv\_logit(X_i\beta) \, \sigma$
    - $b_i = \left(1 - inv\_logit(X_i\beta)\right) \sigma$

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int y[N]; //目的変数
  int t[N];//試行数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //回帰係数
  real<lower=0> sigma;//尺度パラメータ
}
model{
  vector[N] a;//ベータ二項分布のパラメータ
  vector[N] b;//ベータ二項分布のパラメータ
  for(i in 1:N){
    a[i] = inv_logit(x[i]*beta)*sigma;
    b[i] = (1-inv_logit(x[i]*beta))*sigma;
  }
  y ~ beta_binomial(t,a,b);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

欠損値があるデータ

Stanでの欠損値推定

- 欠損データをパラメータとして扱う
  - parameres{}ブロックで指定する
  - データと同じベクトルやマトリックスに欠損値を含めて推定することはできない
    - このあたりが若干面倒な所
- 欠損値を推定する＝すべての情報を活用
  - 完全情報最尤法と同等の推定精度が期待できる
  - ただし，従属変数が1つの線形モデルでは，ほとんど意味がない
    - 欠損データの推定ができるという意味はある

回帰分析で欠損データ推測

- 2変数，N=200のデータを生成
  - v2の順位が低い100人のv1を欠損させる

B	C	D	E
ID	v1	v2	rank
1	6.012185	5.822171	44
2	.	4.281513	152
3	4.91206	5.211129	82
4	.	4.129565	163
5	.	3.674865	180
6	.	4.17858	158
7	4.818966	5.23195	80
8	3.683755	5.587327	56
9	.	4.329512	148
10	6.017103	6.634813	14
11	6.534832	6.022749	32
12	.	4.610202	128
13	.	3.554074	187
14	.	4.730005	114
15	5.603038	5.895204	38
16	6.253573	5.585684	57
17	.	4.208933	156
18	.	3.243447	195
19	.	3.721799	178
20	5.70241	5.975867	36
21	5.436709	5.04447	98

回帰分析をしてみる

- v1をv2で回帰
  - v1の観測部分と欠損部分でデータを分ける
  - v2についても同様
- v1が観測される場合
  - 普通にv2で回帰
- v1が欠損の場合
  - v1の欠損部分をパラメータとして扱い，上の回帰で推定される回帰係数と切片を使って欠損値を推定

Stanコード

```
data{
  int N_obs; //観測されたデータサイズ
  int N_miss;//欠損データサイズ
  int M; //変数の数（切片含む）
  vector[N_obs] y_obs; //目的変数
  matrix[N_obs,M] x_obs; //観測された部分の説明変数
  matrix[N_miss,M] x_miss; //欠損部分の説明変数
}
parameters{
  vector[M] beta; //vectorで宣言
  real<lower=0> sigma;
  vector[N_miss] y_miss;//欠損データ
}
model{
  y_obs ~ normal(x_obs*beta,sigma);
  y_miss ~ normal(x_miss*beta,sigma);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

Rコード

```
dat2 <- read.clipboard(na.strings=".")
y_obs = dat2$v1[!is.na(dat2$v1)]
x_obs = model.matrix(v1~v2,dat2)
x_miss = model.matrix(ID~v2,subset(dat2,is.na(v1)))
N_obs = length(dat2$v1[!is.na(dat2$v1)])
N_miss = length(dat2$v1[is.na(dat2$v1)])
datastan = list(y_obs=y_obs,x_obs=x_obs,x_miss=x_miss,
                N_obs=N_obs,N_miss=N_miss,M=2)
fit.miss <- stan("missing.stan",data=datastan)
fit.miss
```

結果

欠損値を推定した回帰分析

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	0.72	0.03	0.69	-0.60	0.25	0.72	1.17	2.12	401	1.01
beta[2]	0.84	0.01	0.12	0.59	0.76	0.84	0.91	1.07	416	1.01
sigma	0.72	0.00	0.05	0.63	0.69	0.72	0.76	0.84	1499	1.00
y_miss[1]	4.30	0.01	0.75	2.84	3.80	4.29	4.82	5.74	4000	1.00
y_miss[2]	4.17	0.01	0.74	2.71	3.67	4.17	4.66	5.65	4000	1.00
y_miss[3]	3.79	0.01	0.77	2.31	3.27	3.81	4.31	5.32	4000	1.00
y_miss[4]	4.21	0.01	0.76	2.70	3.72	4.23	4.72	5.73	4000	1.00
y_miss[5]	4.35	0.01	0.75	2.90	3.85	4.34	4.84	5.84	4000	1.00
y_miss[6]	4.58	0.01	0.74	3.15	4.09	4.58	5.07	6.03	4000	1.00

普通に回帰分析

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	0.71	0.02	0.67	-0.66	0.27	0.72	1.15	2.00	952	1
beta[2]	0.84	0.00	0.11	0.61	0.76	0.84	0.91	1.07	953	1
sigma	0.72	0.00	0.05	0.63	0.69	0.72	0.76	0.84	1276	1
lp__	-17.40	0.04	1.28	-20.70	-17.99	-17.04	-16.48	-15.97	1011	1

確率変数が1つの場合は，欠損値を推定してもしなくても，パラメータの推定は同じ

相関係数の場合

- v1とv2を両方確率変数として相関を推定
  - 相関係数の真値は0.7
    - 完全データの相関が0.7であるようにした
    - 平均とSDはともにμ=5，σ=1
- 普通に相関を計算すると・・・
  - v1が欠損したv2も捨てることになる
    - v1はv2が低い場合に欠損しているので，データ全体はv2に依存して（低い場合）に特に欠測することになる
  - 相関係数はバイアスが生じる

普通に相関係数を推定

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
mu[1]	5.58	0	0.09	5.40	5.52	5.57	5.64	5.75	2233	1
mu[2]	5.81	0	0.06	5.69	5.77	5.81	5.85	5.94	2214	1
sigma[1]	0.90	0	0.07	0.78	0.85	0.89	0.94	1.04	2702	1
sigma[2]	0.63	0	0.04	0.55	0.60	0.63	0.66	0.73	2353	1
rho[1,1]	1.00	0	0.00	1.00	1.00	1.00	1.00	1.00	4000	NaN
rho[1,2]	0.58	0	0.07	0.43	0.54	0.58	0.63	0.70	2402	1
rho[2,1]	0.58	0	0.07	0.43	0.54	0.58	0.63	0.70	2402	1
rho[2,2]	1.00	0	0.00	1.00	1.00	1.00	1.00	1.00	2843	1

- 相関係数の推定にバイアス
  - ρ=0.58
    - なお，真値は0.70
  - 平均やSDにもバイアスが生じている

欠損を組み込んだStanコード

```
data{
  int N_obs; //観測されたデータサイズ
  int N_miss;//欠損データサイズ
  int M;//変数の数
  vector[M] x_obs[N_obs];//観測された部分
  vector[N_miss] x_miss; //v1が欠損部分のv2
}
parameters{
  vector[M] mu;
  corr_matrix[M] rho; //vectorで宣言
  vector<lower=0>[2] sigma;
}
transformed parameters{
  cov_matrix[M] Sig_mat;
  Sig_mat = quad_form_diag(rho,sigma);
}
model{
  x_obs ~ multi_normal(mu,Sig_mat);
  x_miss ~ normal(mu[2],sigma[2]);
  mu ~ normal(0,100);
  sigma ~ cauchy(0,5);
  rho ~ lkj_corr(1);
}
```

v2については，全データで平均と標準偏差が計算されることがポイント

Rコード

```
x_obs = subset(dat2,!is.na(v1),select=c(v1,v2))
x_miss = dat2$v2[is.na(dat2$v1)]
N_obs = length(dat2$v1[!is.na(dat2$v1)])
N_miss = length(dat2$v1[is.na(dat2$v1)])
datastan = list(x_obs=x_obs,x_miss=x_miss,
                N_obs=N_obs,N_miss=N_miss,M=2)
fit.miss2 <- stan("missing2.stan",data=datastan)
print(fit.miss2,pars=c("mu","sigma","rho"))
```

結果

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
mu[1]	4.91	0	0.13	4.66	4.83	4.91	4.99	5.15	1846	1
mu[2]	5.00	0	0.07	4.86	4.95	5.00	5.05	5.14	2865	1
sigma[1]	1.10	0	0.10	0.93	1.03	1.09	1.16	1.31	1839	1
sigma[2]	1.01	0	0.05	0.92	0.97	1.00	1.04	1.11	2302	1
rho[1,1]	1.00	0	0.00	1.00	1.00	1.00	1.00	1.00	4000	NaN
rho[1,2]	0.74	0	0.06	0.62	0.71	0.75	0.78	0.83	1515	1
rho[2,1]	0.74	0	0.06	0.62	0.71	0.75	0.78	0.83	1515	1
rho[2,2]	1.00	0	0.00	1.00	1.00	1.00	1.00	1.00	4000	1

- 真値=0.7に近い結果
  - ρ=0.74
  - 平均とSDも真値である5と1に近い

打ち切りデータの回帰分析

打ち切りデータ

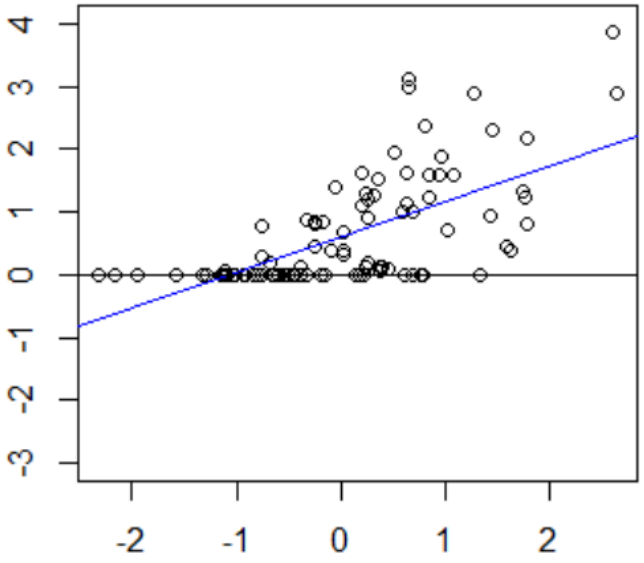
- 測定上，打ち切られてしまったデータ
  - 試験が簡単すぎて，100点が続出した場合
  - ある傾向より小さい場合に得点が0になってしまうようなデータ

- 乱数を生成
  - 片方の変数の0未満の値を0に変換
  - 人工的に打ち切りデータを作ってみる

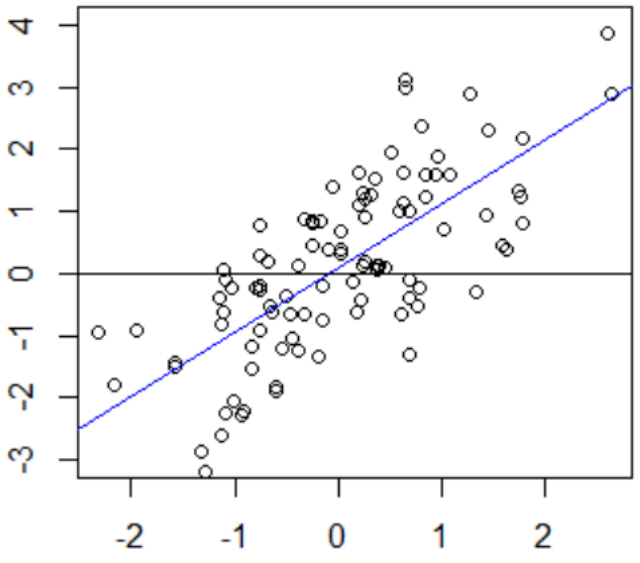
散布図

```
x <- rnorm(100,0,1)
y <- 0 + 1*x + rnorm(100,0,1)
z <- ifelse(y<0,0,y)
plot(x,z)
abline(lm(z~x))
```

真値：α=0，β=1，σ=1



打ち切りデータの回帰直線



真の回帰直線

打ち切り部分を欠損値とみなす

- 欠損値をパラメータとして推定
  - 今回の場合，値が0のデータは，欠損していると考える
  - そして，欠損しているデータは0以下の実数パラメータであると仮定して推定する
- あとは欠損がある場合の回帰と同じ
  - βとσは共通で，観測されたデータと打ち切りデータをそれぞれ正規分布でモデリング

Stanコード1

```
data{
  int N_obs; //観測されたデータサイズ
  int N_cens;//打ち切りデータサイズ
  int M; //変数の数（切片含む）
  vector[N_obs] y_obs; //目的変数
  matrix[N_obs,M] x_obs; //観測された部分の説明変数
  matrix[N_cens,M] x_cens; //打ち切り部分
  real L;//打ち切りポイント
}
parameters{
  vector[M] beta; //vectorで宣言
  real<lower=0> sigma;
  vector<upper=L>[N_cens] y_cens;
}
model{
  y_obs ~ normal(x_obs*beta,sigma);
  y_cens ~ normal(x_cens*beta,sigma);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

Rコード

```
dat2 <- data.frame(z,x)
y_obs = dat2$z[dat2$z!=0]
x_obs = model.matrix(z~x,subset(dat2,z!=0))
x_cens = model.matrix(z~x,subset(dat2,z==0))
N_obs = length(dat2$z[dat2$z!=0])
N_cens = length(dat2$z[dat2$z==0])
datastan = list(y_obs=y_obs,x_obs=x_obs,x_cens=x_cens,
                N_obs=N_obs,N_cens=N_cens,M=2,L=0)
fit.cens <- stan("censored.stan",data=datastan)
fit.cens
```

結果

真値：α=0，β=1，σ=1

普通の回帰分析

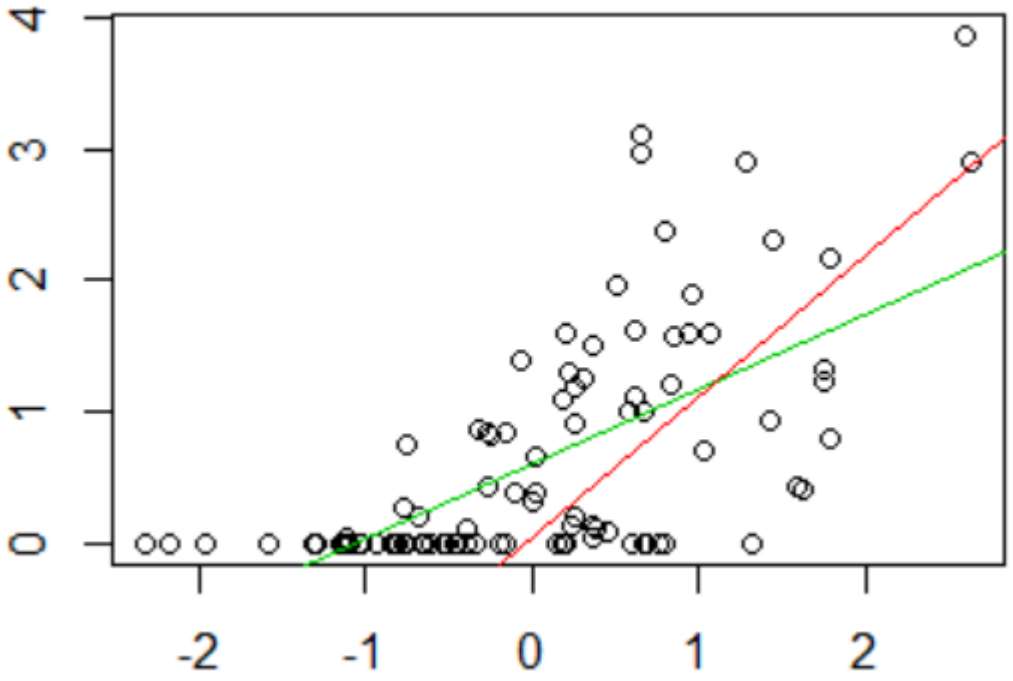
	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	0.60	0.00	0.07	0.47	0.56	0.60	0.65	0.73	3042	1
beta[2]	0.57	0.00	0.07	0.44	0.52	0.57	0.62	0.70	2598	1
sigma	0.68	0.00	0.05	0.59	0.65	0.68	0.71	0.78	2474	1
lp__	-10.44	0.03	1.20	-13.53	-10.97	-10.12	-9.58	-9.11	1922	1

打ち切りデータの回帰分析

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	0.05	0.00	0.14	-0.24	-0.04	0.06	0.15	0.30	1378	1
beta[2]	1.07	0.00	0.14	0.82	0.98	1.07	1.16	1.37	1259	1
sigma	1.01	0.00	0.11	0.82	0.93	1.00	1.07	1.24	1621	1
y_cens[1]	-1.08	0.01	0.74	-2.78	-1.54	-0.97	-0.49	-0.07	4000	1
y_cens[2]	-1.55	0.01	0.89	-3.50	-2.12	-1.47	-0.87	-0.13	4000	1
y_cens[3]	-0.56	0.01	0.49	-1.83	-0.81	-0.43	-0.19	-0.02	4000	1
y_cens[4]	-0.97	0.01	0.68	-2.56	-1.38	-0.87	-0.44	-0.05	4000	1
y_cens[5]	-0.73	0.01	0.57	-2.13	-1.04	-0.59	-0.28	-0.03	4000	1

回帰係数が急になり，残差のSDが大きく推定されている
真値に近い結果となっている

散布図



緑：普通に推定
赤：打ち切り回帰で推定

別の書き方

- 欠損を推定せずに推定
  - 欠損パラメータを積分消去する
  - すると，累積分布関数で表現できる

- 下限打ち切りの場合
  - 対数累積分布関数・・・normal_lcdf()を使う
    - target += normal_lcdf(L | x*beta,sigma);

- 上限打ち切りの場合
  - 対数相補累積分布関数・・・normal_lccdf()を使う
    - target += normal_lccdf(U | x*beta,sigma);

Stanコード2

```
data{
  int N_obs; //観測されたデータサイズ
  int N_cens;//打ち切りデータサイズ
  int M; //変数の数（切片含む）
  vector[N_obs] y_obs; //目的変数
  matrix[N_obs,M] x_obs; //観測された部分の説明変数
  matrix[N_cens,M] x_cens; //打ち切り部分
  real L;//打ち切りポイント
}
parameters{
  vector[M] beta; //vectorで宣言
  real<lower=0> sigma;
}
model{
  y_obs ~ normal(x_obs*beta,sigma);
  target += normal_lcdf(L|x_cens*beta,sigma);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

こっちのほうが計算速度は速い

結果2

- 打ち切り部分を積分消去したモデル

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	0.05	0.00	0.14	-0.24	-0.04	0.06	0.14	0.30	2213	1
beta[2]	1.07	0.00	0.13	0.83	0.98	1.07	1.16	1.36	2169	1
sigma	1.00	0.00	0.11	0.82	0.93	0.99	1.07	1.23	2083	1
lp__	-49.02	0.03	1.19	-52.06	-49.59	-48.70	-48.15	-47.67	1828	1

- 打ち切り部分を欠損推定したモデル

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	0.05	0.00	0.14	-0.24	-0.04	0.06	0.15	0.30	1378	1
beta[2]	1.07	0.00	0.14	0.82	0.98	1.07	1.16	1.37	1259	1
sigma	1.01	0.00	0.11	0.82	0.93	1.00	1.07	1.24	1621	1
y_cens[1]	-1.08	0.01	0.74	-2.78	-1.54	-0.97	-0.49	-0.07	4000	1
y_cens[2]	-1.55	0.01	0.89	-3.50	-2.12	-1.47	-0.87	-0.13	4000	1
y_cens[3]	-0.56	0.01	0.49	-1.83	-0.81	-0.43	-0.19	-0.02	4000	1
y_cens[4]	-0.97	0.01	0.68	-2.56	-1.38	-0.87	-0.44	-0.05	4000	1
y_cens[5]	-0.73	0.01	0.57	-2.13	-1.04	-0.59	-0.28	-0.03	4000	1

離散分布の過分散問題

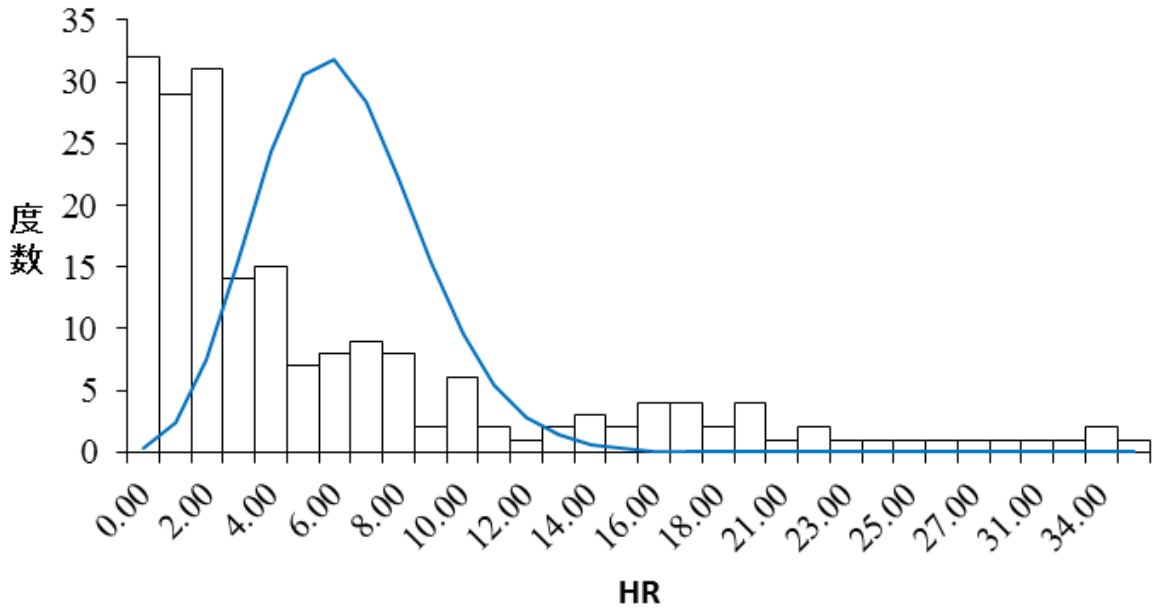
二項分布とポアソン分布

- どちらも生起数についての離散分布
  - パラメータが一つしかない
  - 平均パラメータから分散が一意に決まる
    - 二項分布： 平均=θ, 分散=T$\pi(1-\theta)$
    - ポアソン分布： 平均=λ, 分散= λ

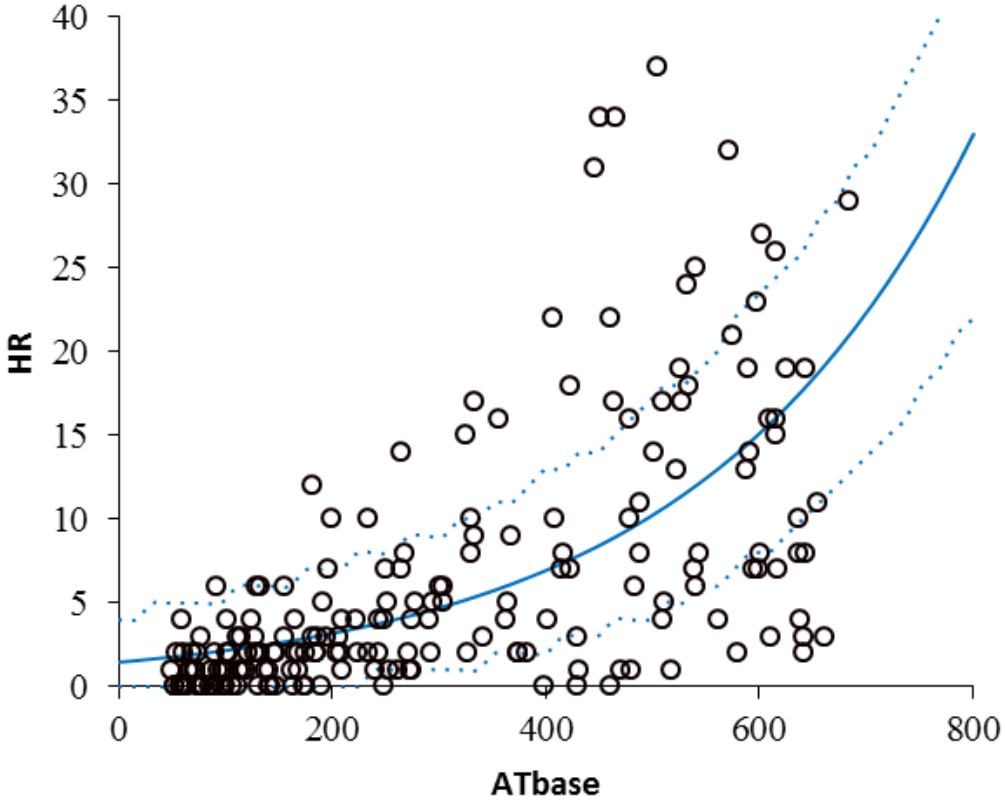
- データの散らばりが分布に合わない場合
  - 過分散が生じているという

ホームラン数のデータ

- ホームラン
  - レアな事象が生じる回数
  - しかし，ポアソン分布には全然合わない



ホームランを打席数で予測



予測区間にデータが
全然収まってない！

過分散を変量効果で解決

- 固定効果
  - 全体的な効果を表すパラメータ
    - グローバルパラメータ
  - 平均やSDなど

- 変量効果
  - ここの対象ごとに異なる効果を表すパラメータ
    - ローカルパラメータ
  - 対象による細やかな違いを表現できる

モデルで書いてみる

- 普通のポアソン分布のモデル
  - $y_i \sim Poisson(\lambda_i)$
  - $\log(\lambda_i) = X_i\beta$

- λ_iに変量効果r_iを加える
  - $y_i \sim Poisson(\lambda_i)$
  - $\log(\lambda_i) = X_i\beta + r_i$
  - $r_i \sim N(0, \sigma)$

書き方を変えると・・・

- 変量効果を含むモデル
  - $y_i \sim Poisson(\lambda_i)$
  - $\log(\lambda_i) \sim N(\mu_i, \sigma)$
  - $\mu_i = X_i\beta$
    - これは結局λ_iが平均=μ，SD=σの対数正規分布に従っていることと同じ　つまり，$\lambda_i \sim Lognormla(\mu_i, \sigma)$

- 階層モデル
  - 確率分布のパラメータが，さらに別の確率分布に従い，高次のパラメータが存在する
  - 確率分布が階層的になっている

Stanでの書き方

- Stanはリンク関数とセットの確率分布がある
  - poisson_log()
  - これを使うととても書きやすいし，コードがベクトル化できてスピードも速くなる

- ポアソン分布＋変量効果のモデル
  - $y_i \sim Poisson\_\log(\lambda'_i)$
  - $\lambda'_i \sim normal(\mu_i, \sigma)$
  - $\mu_i = X_i\beta$

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //回帰係数
  real<lower=0> sigma;//変量効果のSD
  vector[N] lambda;//変量効果パラメータ
}
model{
  y ~ poisson_log(lambda); //ポアソン分布
  lambda ~ normal(x*beta,sigma);//λがさらに正規分布に従う
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

二項分布＋変量効果も同じ

- 二項分布に変量効果
  - $y_i \sim Binomial(\theta_i)$
  - $logit(\theta_i) = X_i\beta + r_i$
    - $r_i \sim N(0, \sigma)$

- Stanのbinomial_logit()を使って書いてみる
  - $y_i \sim binomial\_logit(\theta_i')$
  - $\theta_i' \sim normal(X_i\beta, \sigma)$
    - $\beta \sim normal(0{,}100)$
    - $\sigma \sim cauchy(0{,}5)$

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int t[N];//試行数
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] beta; //vectorで宣言
  real<lower=0> sigma;//変量効果のSD
  vector[N] theta;//変量効果パラメータ
}
model{
  y ~ binomial_logit(t,theta);
  theta ~ normal(x*beta,sigma);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);
}
```

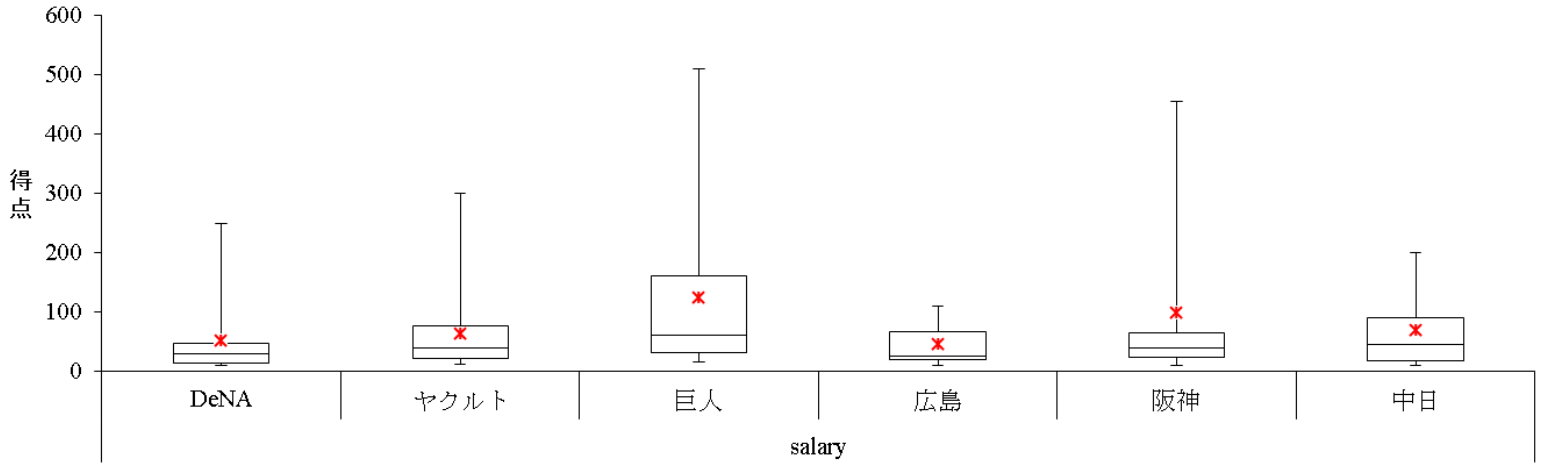
階層線形モデル

複数のグループからデータを取る

- 回帰分析の仮定
  - 同一の分布から、独立に生成
  - グループが複数あると、独立性が満たされない
- グループ間変動を変量効果で表現
  - 切片や回帰係数が集団で異なる
  - 変量効果で集団による違いを表現
  - 固定効果で全体の特徴を表現

またもやプロ野球データ

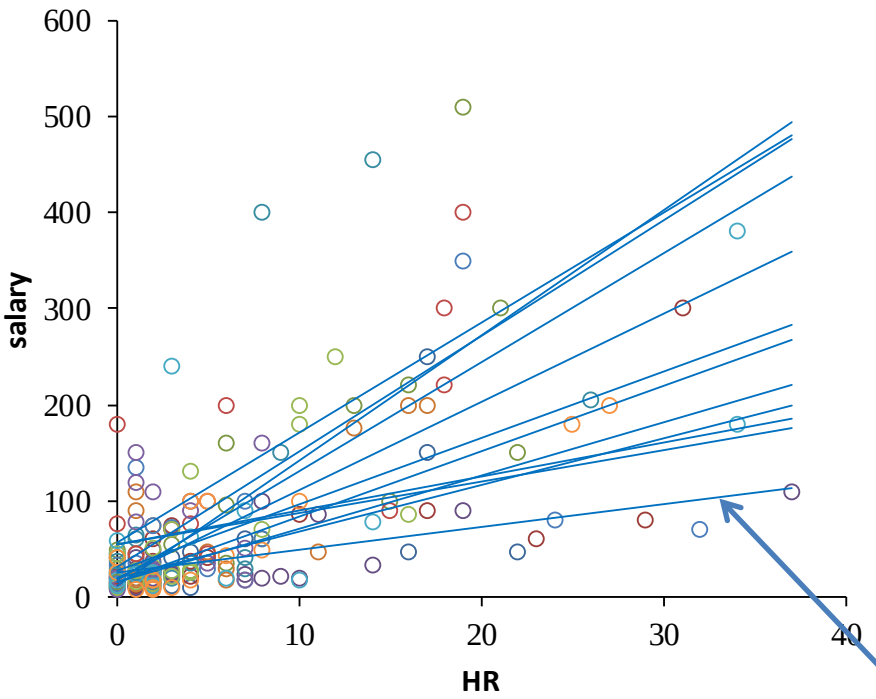
- 日本プロ野球は12球団に分かれる
  - 球団によって，モデルが変わる可能性



広島・・・！？

年俸をHRで予測

- チーム別の回帰直線



teamID	切片	傾き
DeNA	16.415	5.509
ヤクルト	19.767	4.872
巨人	10.757	13.057
広島	25.402	2.376
阪神	31.144	12.030
中日	20.227	9.165
オリックス	54.865	3.291
ソフトバンク	57.501	11.436
ロッテ	16.689	11.390
楽天	55.265	3.497
西武	27.366	6.923
日本ハム	14.588	6.832

広島・・・！？

チームによってモデルは違うっぽい

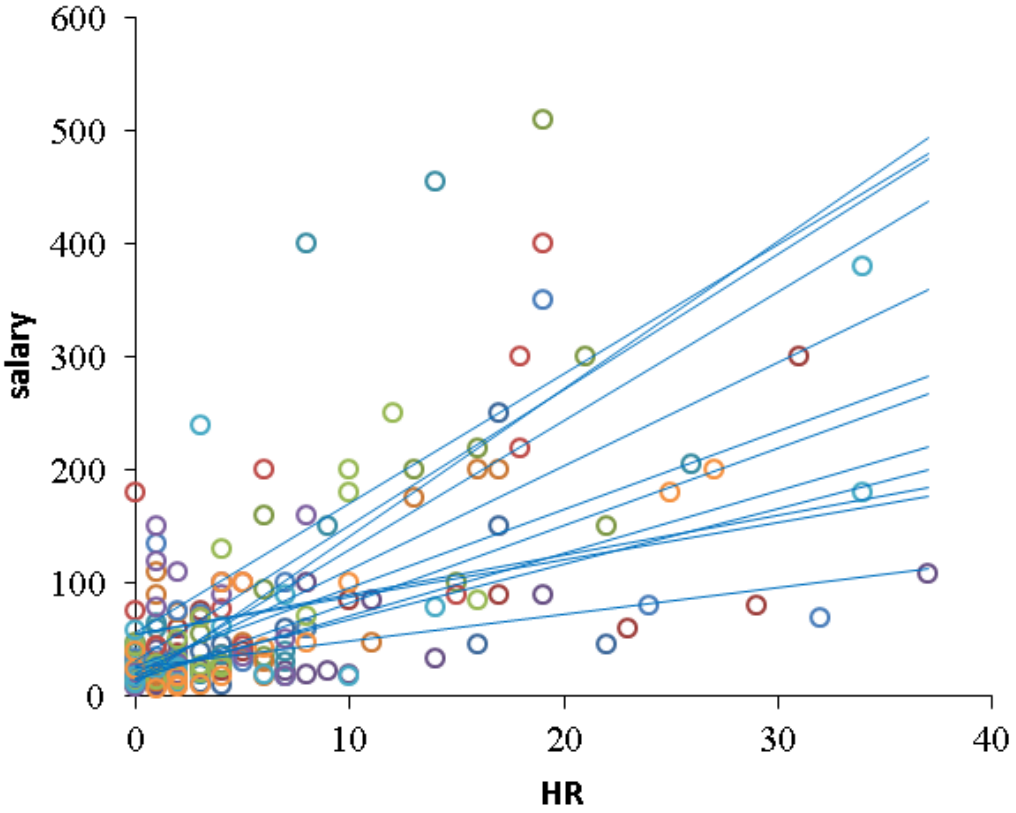
- お金がある球団ほど，HRの効果が大きそう
  - でもそれをどうやってモデリングしたらいいだろうか？

- 愚直な考え：12個の回帰モデルを解釈する
  - 巨人のHRの効果はいくら，広島の効果はいくら・・・
  - 12ならまだいいが，集団が100個の場合は？
    - パラメータ数が増える→オーバーフィッティング
  - そもそも個々の集団に興味がない場合は？
    - 実験的に作られた集団それぞれの効果が出ても・・・

変量効果で集団間変動を表現

- グループごとの違いを同時にモデリング
  - 回帰直線をグループごとに仮定することなく，一度に推定する
  - しかし，グループごとの違いをうまく表現する
- そこで使えるのが変量効果
  - 個々の推定値には興味がないが，全体的な散らばりはモデルに含めておきたい場合

回帰係数が集団によって違う



この例では、切片は集団間で同じとする

回帰係数だけが違うモデリング

- まずは単回帰分析の復習
  - $y_i \sim N(\mu_i, \sigma)$
  - $\mu_i = \alpha + X_i\beta$
- 集団による違いをjで表現
  - $y_{ij} \sim N(\mu_{ij}, \sigma)$
  - $\mu_{ij} = \alpha + X_{ij}\beta_j$
    - jの添え字が増えただけ
    - 切片は同じと仮定する

変量効果の導入

- 切片と回帰係数が正規分布に従う
  - $y_{ij} \sim N(\mu_{ij}, \sigma)$
  - $\mu_{ij} = \alpha + X_{ij}\beta_j$
  - $\beta_j \sim N(\gamma, \tau)$
    - パラメータが、別の確率モデルを持つ＝階層モデル
    - γは回帰係数の固定効果
    - τは回帰係数の集団間変動（変量効果のSD）
- 次のように書いても良い
  - $y_{ij} \sim N(\mu_{ij}, \sigma)$
  - $\mu_{ij} = \alpha + X_{ij}\beta_j$
  - $\beta_j = \gamma + r_j$
  - $r_j \sim N(0, \tau)$

Stanでのモデリング

- 集団を意味するjをコードで表現
  - サンプルは199人
  - 球団は12個
  - →こういうデータを使う

playerID	teamID	salary	HR
1	1	46.0	16
2	1	60.0	7
3	1	150.0	17
4	1	46.0	22
5	1	28.0	2
6	1	26.0	2
7	1	250.0	17
8	1	18.5	1
9	1	40.0	3
10	1	40.0	0
11	1	11.0	1
12	1	46.0	4
13	1	23.0	1
14	1	10.5	4
15	1	11.5	3
16	1	10.0	0
17	1	13.5	0
18	2	80.0	29
19	2	85.0	10
20	2	60.0	23
21	2	90.0	17
22	2	300.0	31
23	2	40.0	5

Stanでのモデリング

- 集団を識別するID変数をint型で宣言
  - 個人の数をN、集団の数をGとする
  - data{}ブロックで、teamIDを宣言
  - int teamID[N]

- 回帰係数の変量効果を集団数Gで宣言
  - real beta[G]と宣言して、model{}ブロックでは、
  - for(i in 1:N) beta[teamID[i]]・・・　と書く
  - これで、個人iが所属しているチームの回帰係数を指定していることになる

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int G; //集団の数
  vector[N] y; //目的変数
  vector[N] x; //説明変数
  int teamID[N]; //集団のID　集団効果のラベルに使う
}
parameters{
  real alpha;//切片
  real beta[G];//回帰係数(変量効果)
  real gamma; //回帰係数(固定効果)
  real<lower=0> sigma; //残差のSD
  real<lower=0> tau; //集団間変動(SD)
}
model{
  vector[N] mu;
  for(i in 1:N) mu[i] = alpha+x[i]*beta[teamID[i]];
  y ~ normal(mu,sigma);
  beta ~ normal(gamma,tau); //集団効果が平均=γ,SD=tauの正規分布
  alpha ~ normal(0,100);
  gamma ~ normal(0,100);
  sigma ~ cauchy(0,5);
  tau ~ cauchy(0,5); //事前分布のパラメータの事前分布
}
```

説明変数が1つだけで、かつ、回帰係数のみに変量効果を仮定したモデル

切片と回帰係数の両方が変量効果

- モデリング
  - $y_{ij} \sim N(\mu_{ij}, \sigma)$
  - $\mu_{ij} = X_{ij}\boldsymbol{\beta}_j$　ただし、Xは行列、βは要素2ベクトル
  - $\boldsymbol{\beta}_j \sim MultiNormal(\boldsymbol{\gamma}, \mathbf{T})$
    - ベクトルβが、平均ベクトルγ、共分散行列Tの多変量正規分布に従う

- 多変量正規分布
  - 平均がベクトル、SDが共分散行列に代わる
  - モデリングに工夫が必要なので注意

Stanで多変量正規分布を使う

- パラメータはベクトル＋配列で宣言
  - 説明変数の数+1をM、集団の数をGとする
  - vector[M] beta[G];
    - 多変量正規分布はベクトルをサンプリングするから
  - と宣言する

- サンプリングステートメントの書き方
  - 多変量正規分布の書き方
    - vector ~ multi_normal(vector, cov_matrix)
  - つまり、beta ~ multi_normal(gamma, Tau)

共分散行列のモデリング

- 共分散行列はパラメータとして宣言しない
  - 共分散行列の事前分布の指定が難しい
  - そこで、SDと相関行列に分ける

- SDと相関行列をパラメータとして宣言
  - parameters{}ブロック
  - corr_matrix[M] omega; M×Mの相関行列
  - vector[M] tau; 説明変数の数だけベクトルで宣言

共分散行列のモデリング

- 共分散行列を宣言
  - cov_matrix型で宣言
  - cov_matrix[M] Tau;　M×Mの共分散行列
- SDと相関行列から共分散行列を作る
  - Tau = quad_form_diag(omega, tau);
    - これは、
    - Tau = diag_matrix(tau) * omega * diag_matrix(tau)
    - と同じ意味

相関行列の事前分布

- lkj_corr()を使う
  - lkj_corr(2)がStanではよく使われる
  - 弱情報事前分布

  - 今回は無情報事前分布にしたいので,
  - omega ~ lkj_corr(1);
  - としておく

Stanコード　前半

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int G; //集団の数
  vector[N] y; //目的変数
  matrix[N,M] x; //説明変数
  int teamID[N]; //集団のID　集団効果のラベルに使う
}
parameters{
  vector[M] beta[G];//変量効果：ベクトルが集団の数ある
  vector[M] gamma; //回帰係数(固定効果)
  real<lower=0> sigma; //残差のSD
  vector<lower=0>[M] tau; //変量効果のSDベクトル
  corr_matrix[M] omega;//相関行列
}
```

betaとgamma，そしてtauがrealからvector[M]に変わっている点に注意

Stanコード　後半

```
transformed parameters{
  cov_matrix[M] Tau;
  Tau = quad_form_diag(omega,tau);
}
model{
  vector[N] mu;
  for(i in 1:N) mu[i] = x[i]*beta[teamID[i]];

  y ~ normal(mu,sigma);
  beta ~ multi_normal(gamma,Tau);

  gamma ~ normal(0,100);
  sigma ~ cauchy(0,5);
  tau ~ cauchy(0,5);
  omega ~ lkj_corr(1);
}
```

一部の説明変数を変量効果に指定

- 固定要因と変量要因を分ける
  - 固定要因をX、変量要因をZとする
    - XとZは一部混ざっていても問題はない
    - すべての変数について変量効果を仮定する場合、XとZは同じ行列となる
- もっとも一般的な書き方
  - 固定効果だけの変数、両方を推定したい変数、そして変量効果だけを推定したい変数など、いろんなモデルを表現できる

モデリング

- 固定効果と変量効果を分ける
  - $y_{ij} \sim N(\mu_{ij}, \sigma)$
  - $\mu_{ij} = X_{ij}\gamma + Z_{ij}r_j$
    - γは固定効果、rが変量効果
  - $r_j \sim MultiNormal(\mathbf{0}, \mathbf{T})$
    - 変量効果rは平均0ベクトル、共分散行列Tの多変量正規分布に従う

Stanでのモデリング

- rep_vector()を使うと便利
  - rep_vector(0,Q)と書くと，0がQ個並んだベクトルを作ることができる
  - 多変量正規分布の平均に0ベクトルを入れたい場合に重宝する

Stanコード　前半

```
data{
  int N; //サンプルサイズ
  int G; //集団の数
  int P;//固定効果の数
  int Q;//変量効果の数
  vector[N] y; //目的変数
  matrix[N,P] x;//固定効果の説明変数
  matrix[N,Q] z;//変量効果の説明変数
  int teamID[N]; //集団のID　集団効果のラベルに使う
}
parameters{
  vector[P] gamma; //回帰係数(固定効果)
  vector[Q] r[G];//変量効果：ベクトルが集団の数ある
  real<lower=0> sigma; //残差のSD
  vector<lower=0>[Q] tau; //変量効果のSDベクトル
  corr_matrix[Q] omega;//相関行列
}
```

Stanコード　後半

```
transformed parameters{
  cov_matrix[Q] Tau;
  Tau = quad_form_diag(omega,tau);
}
model{
  vector[N] mu;
  for(i in 1:N) mu[i] = x[i]*gamma+z[i]*r[teamID[i]];

  y ~ normal(mu,sigma);
  r ~ multi_normal(rep_vector(0,Q),Tau);

  gamma ~ normal(0,100);
  sigma ~ cauchy(0,5);
  tau ~ cauchy(0,5);
  omega ~ lkj_corr(1);
}
```

変量効果のSDの分布

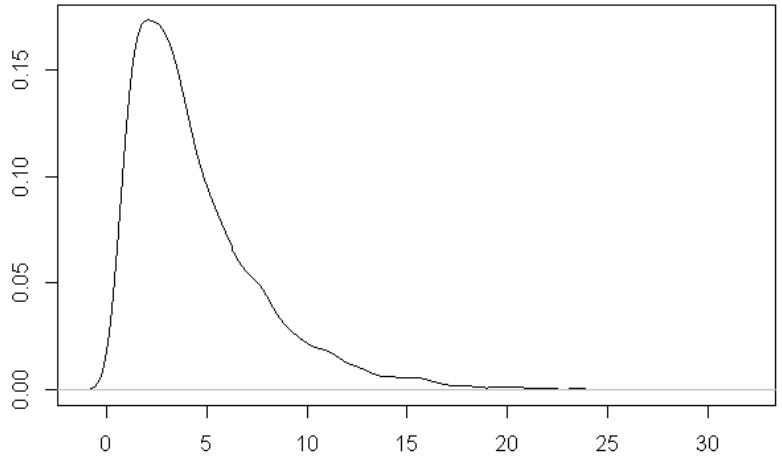
- 平均値が大きそうに見えても注意が必要

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	28.63	0.11	6.03	16.63	24.61	28.57	32.85	39.96	2841	1.00
beta[2]	7.51	0.05	1.25	5.06	6.71	7.49	8.28	10.00	767	1.00
tau[1]	4.57	0.22	3.50	0.75	2.09	3.59	5.97	13.95	264	1.01
tau[2]	3.57	0.02	0.97	1.96	2.89	3.47	4.15	5.66	1822	1.00

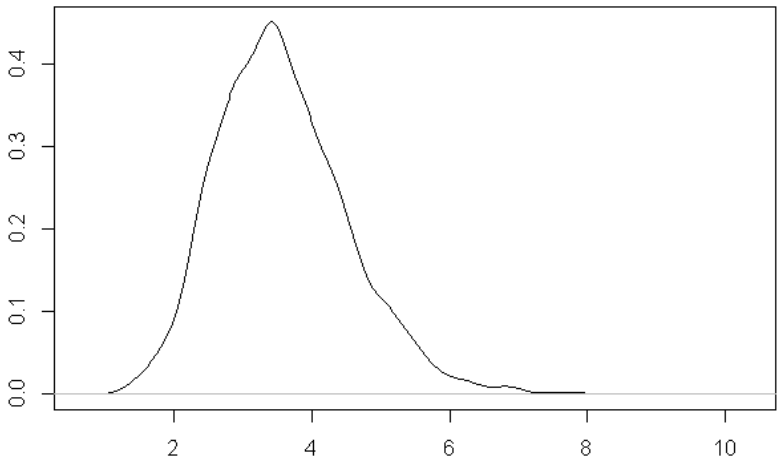
  - tau1は、tau2よりも大きそうに見えるが・・・

分布をみてみる

- tau1の分布



- tau2の分布



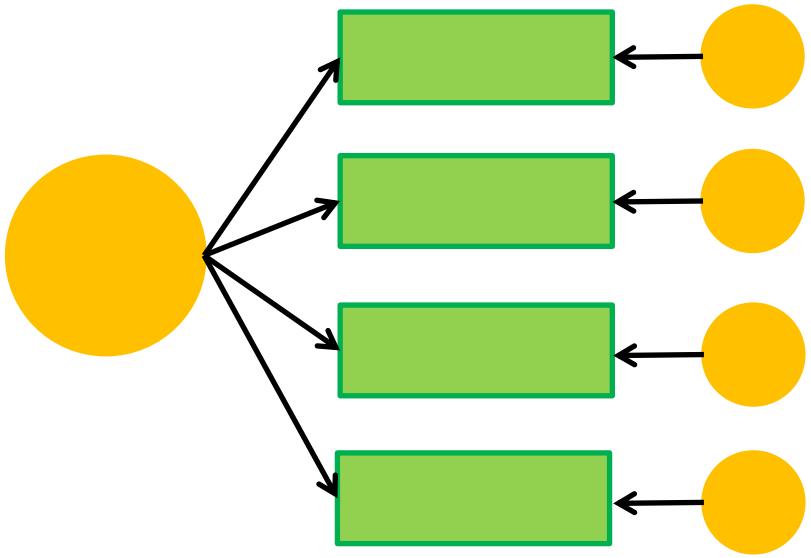
事後確率最大値を確認

- MAPを計算
  - カーネル密度推定を利用する方法
  - map <- function(z){
    density(z)$x[which.max(density(z)$y)]
    }
  - tau1=2.07
  - tau2=3.42

潜在変数があるモデル

因子分析法

- 知能の次元を推定する統計手法
  - スピアマンが一般知能と特殊知能に分離するために開発した



因子分析のモデル

- 各項目が，因子得点と因子負荷量で表現
  - $y_{ij} = \alpha_j + \theta_{ik}\lambda_{jk}' + e_{ij}$
  - $e_{ij} \sim N(0, \psi)$
  - $\theta_i \sim N(0,1)$
    - iは個人，jは項目，kは因子を識別する添字
    - αは項目の平均，λは因子負荷量，ψは誤差SD
- 確率モデル
  - $y_{ij} \sim N(\mu_{ij}, \psi_j)$
  - $\mu_{ij} = \alpha_j + \theta_{ik}\lambda_{jk}'$
  - $\theta_i \sim N(0,1)$

因子分析の確率モデル

- 各観測変数と潜在変数は正規分布に従う
  - $y_{ij} \sim N(\alpha_j + \theta_{ik}\lambda'_{jk}, \psi_j)$
    - iは個人，jは項目を識別する添字
    - αは項目の平均，λ_jは因子負荷量，ψ_jは誤差SD
  - $\theta_i \sim N(0, 1)$
    - 因子得点は標準正規分布に従う

- 項目でfor文を回してコードする
  - 個人はベクタライズする

因子負荷量の不定性

- 因子の回転の不定性
  - 誤差SDが推定できても、因子負荷量は回転について自由度を持つ
- 因子負荷量の符号の不定性
  - 因子ごとに、負荷量と因子得点の符号がお互いに逆転しても、解は同じになる
- そこで使えるのがcholesky_factor_cov()
  - 回転の自由度の分、0の固定母数を持つ
  - 一番上の負荷量の符号を正に制約

cholesky_factor_cov型

	F1	F2	F3	F4
m1	positive	0	0	0
m2	free	positive	0	0
m3	free	free	positive	0
m4	free	free	free	positive
m5	free	free	free	free
m6	free	free	free	free
m7	free	free	free	free
m8	free	free	free	free
m9	free	free	free	free
m10	free	free	free	free

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int F; //因子数
  vector[M] y[N]; //変数
}
parameters{
  vector[M] alpha;
  cholesky_factor_cov[M,F] lambda;
  vector<lower=0>[M] psi;
  matrix[N,F] theta;
}
model{
  matrix[N,M] mu;
  mu = theta*lambda';
  for(m in 1:M)
    y[m] ~ normal(alpha[m]+mu[m],psi[m]);
  for(f in 1:F)
    theta[f] ~ normal(0,1);
  to_vector(lambda) ~ normal(0,100);
  alpha ~ normal(0,100);
  psi ~ cauchy(0,5);
}
```

因子負荷量行列を、cholesky_factor_covで宣言しているところがポイント

因子得点の周辺化

- 因子得点を積分で消去する
  - そうすると，個々の因子得点を推定せずに，項目についてのパラメータのみを推定することになる
  - 推定が速くなる
- 尤度関数は，
  - $L(y_{ij}|\alpha,\lambda,\psi) = \int_{-\infty}^{\infty} N\left(y_{ij}\,\middle|\alpha_j + \theta_{ik}\lambda_{jk}{}',\psi_j\right) N\left(\theta_{ij}\,|0,1\right)\, d\,\theta$
- 周辺化すると尤度関数は多変量正規分布になる
  - $L(\boldsymbol{y}_i|\mathbf{A},\boldsymbol{\Lambda},\boldsymbol{\Psi}) = Multi\_Normal(\boldsymbol{y}_i\,|\mathbf{A},\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\boldsymbol{t}} + \boldsymbol{\Psi})$
    - Aは平均ベクトル、Λは因子負荷量行列、Ψは誤差SDの対角行列

Stanコード

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int F; //因子数
  vector[M] y[N]; //変数
}
parameters{
  vector[M] alpha;
  cholesky_factor_cov[M,F] lambda;
  vector<lower=0>[M] psi;
}
transformed parameters{
  cov_matrix[M] Psi;
  Psi = tcrossprod(lambda)+diag_matrix(psi);
}
model{
  y ~ multi_normal(alpha,Psi);
  to_vector(lambda) ~ normal(0,100);
  alpha ~ normal(0,100);
  psi ~ cauchy(0,5);
}
```

ただし、これでもうまくいかないことも

- 連鎖ごとに因子負荷量の符号が入れ替わる
  - ラベルスウィッチング問題
- 解決法1
  - あとで自力でラベルを合わせる
  - 面倒だけど、まぁできないことはない
- 解決法2
  - 変分ベイズを使う
    - 詳細は後述
  - 因子分析ならいい方法かもしれない

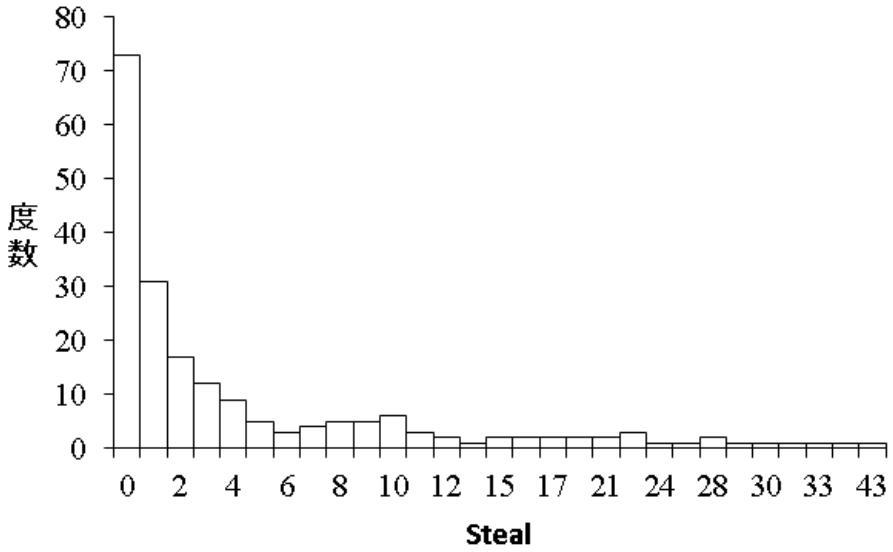
それでもダメなら

- fa2stanを使う
  - これらの問題をたぶんだいたい解決してる
  - 回転もできる
  - 推定が速くなるような工夫もしてる
    - ウィシャート分布でさらに速く

ゼロ過剰分布

ゼロ過剰分布

- 想定している分布よりゼロデータが多い
  - 盗塁のデータとか
  - そもそも盗塁をしようとしていない人と、盗塁を試みる人を分けてモデリングする必要がある

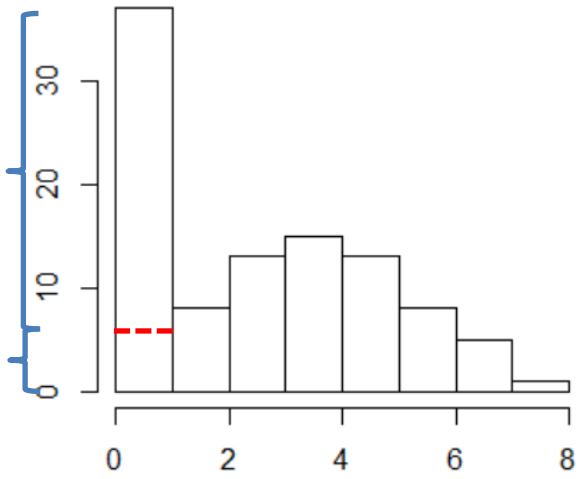


ゼロ過剰ポアソン

- ポアソン分布よりもゼロが多いデータ
  - $y \sim Bernoulli(\theta)$でゼロデータか、ポアソン分布に従うかが決まる
  - ゼロデータならすべて0、そうでないなら、平均がλのポアソン分布に従うとする

ベルヌーイ分布による0
盗塁しないやつ

ポアソン分布による0
盗塁を試みて0のやつ



モデリング

- ベルヌーイ分布とポアソン分布を混ぜる
  - ベルヌーイ分布で0か0以外の値かをモデリング
  - ポアソン分布で，0以外のときの値をモデリング

- データが0か0以外かでモデルが変わる
  - y=0のとき
    - ベルヌーイ分布で0になる確率と，ベルヌーイ分布が1になる確率×ポアソン分布で0になる確率の和
  - y＞0のとき
    - ベルヌーイ分布が1になる確率×ポアソン分布の確率
  - $p(y_i|\theta,\lambda) = \begin{cases} (1-\theta) + \theta\ Poisson(0|\lambda) & if\ y_i = 0 \\ \theta\ Poisson(y_i|\lambda) & if\ y_i > 0 \end{cases}$

Stanでモデリング

- yが0と0以外で条件わけ
  - y=0のとき
    - $P(y_i) = \theta * Poisson(0|\lambda) + (1 - \theta)$
  - y>0のとき
    - $P(y_i) = \theta * Poisson(y_i|\lambda)$
  - サンプリング形式ではなく，ターゲット形式で表現する

Stanでモデリング

- Stanでは対数確率を足し上げる
  - target += を使う
  - 確率の掛け算は，対数確率では足し算になる

- 確率の足し算の表現
  - Stanでは，対数確率を一度確率に戻して，それを足してまた対数にするための関数がある
    - log_sum_exp()
  - log_sum_exp(log(1-θ), log(θ)+poisson_lpmf(0|λ))

Stanコード1

```
data{
  int N; //サンプルサイズ
  int y[N]; //目的変数
}
parameters{
  real<lower=0,upper=1> theta;
  real<lower=0> lambda;
}
model{
  for(n in 1:N){
    if(y[n]==0)
      target += log_sum_exp(log(1-theta),log(theta)+poisson_lpmf(0|lambda));
    else
      target += log(theta)+poisson_lpmf(y[n]|lambda);
  }
}
```

ゼロ過剰ポアソン回帰

- 説明変数を加えただけ
  - ロジスティック回帰とポアソン回帰を使う
- ロジスティック回帰
  - 盗塁をするのかしないのか，その確率を予測
- ポアソン回帰
  - 盗塁をした場合，その盗塁数を予測

Stanでモデリング

- yが0と0以外で条件わけ
  - y=0のとき
    - $P(y_i) = \theta * Poisson(0|\lambda_i) + (1 - \theta)$
  - y>0のとき
    - $P(y_i) = \theta * Poisson(y_i|\lambda_i)$

  - $\theta_i = inv\_logit(X_i\alpha)$
  - $\lambda_i = \exp(X_i\beta)$

  - αはロジスティック回帰の係数，βはポアソン回帰の係数

Stanコード2

```
data{
  int N; //サンプルサイズ
  int M; //変数の数（切片含む）
  int y[N]; //目的変数
  matrix[N,M] x; //説明変数 matrixで宣言
}
parameters{
  vector[M] alpha;
  vector[M] beta;
}
model{
  vector[N] theta;
  vector[N] lambda;
  for(n in 1:N){
    theta[n] = inv_logit(x[n]*alpha);
    lambda[n] = exp(x[n]*beta);
    if(y[n]==0)
      target += log_sum_exp(log(1-theta[n]),log(theta[n])+poisson_lpmf(0|lambda[n]));
    else
      target += log(theta[n])+poisson_lpmf(y[n]|lambda[n]);
  }
}
```

盗塁を体重で予測

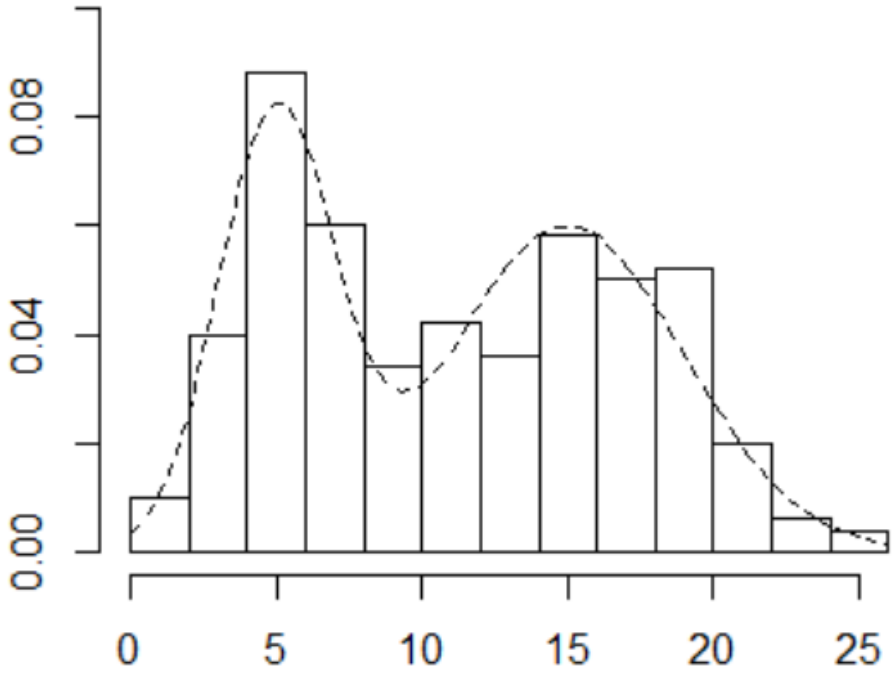
	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
alpha[1]	3.92	0.14	1.39	1.14	2.95	3.93	4.88	6.58	93	1.00
alpha[2]	-0.04	0.00	0.02	-0.07	-0.05	-0.04	-0.03	-0.01	94	1.00
beta[1]	4.87	0.01	0.29	4.31	4.67	4.86	5.06	5.44	1342	1.00
beta[2]	-0.03	0.00	0.00	-0.04	-0.04	-0.03	-0.03	-0.03	1337	1.00
lp__	-814.73	0.04	1.32	-818.00	-815.41	-814.47	-813.75	-813.02	873	1.01

- ロジスティック回帰の結果
  - alphaの結果
  - 体重が重いほど，そもそも盗塁を試みない傾向
- ポアソン回帰の結果
  - betaの結果
  - 体重が重いほど，盗塁数が少ない傾向

混合分布モデル

混合正規分布

- 複数の正規分布の組み合わせ
  - 2つの正規分布の場合



混合分布の確率モデル

- K個の正規分布から生成された場合
  - $z_i \sim Categorical(\pi)$
    - Categorical分布は，多項のベルヌーイ分布
    - zは確率ベクトルπに従って，1~Kのどの正規分布から生成されたかを意味する離散パラメータ
- データは個体iが所属する正規分布から生成
  - $y_i \sim N(\mu_{z_i}, \sigma_{zi})$
    - μ_{zi}は，zが1～Kに対応する平均値のパラメータ
    - σ_{zi}も同様
    - ある種の階層モデル

離散パラメータの消去

- zは離散パラメータなので累積和で消去する
  - すると，zは消えて，π，μ，σがパラメータになる

$$P(y_i|\pi,\mu,\sigma) = \sum_{k-1}^{K} \pi_k \, N(y_i|\,\mu_k,\sigma_k)$$

  - これをStanでモデリングするためには，ゼロ過剰分布と同様，ターゲット形式で書く必要がある

Stanでのモデリング

- それぞれの正規分布の対数確率
  - まず個人ごとにそれぞれの正規分布から生成される対数尤度を計算する
  - その際，log(π)も足す
- target+=で全データの尤度を足し上げる
  - 全データの対数尤度をtarget+=に足し上げる
  - 各個人の尤度はlog_sum_exp()で合成

Stanコード

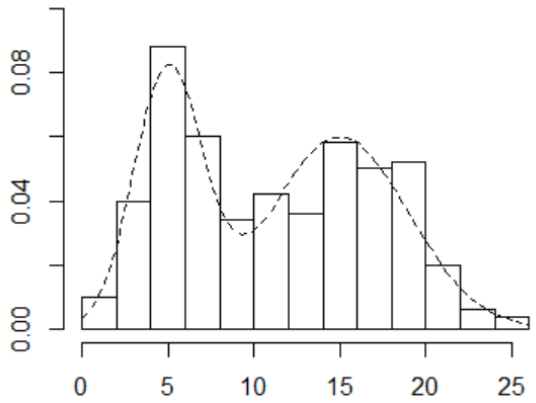
```
data{
  int N; //サンプルサイズ
  int K; //正規分布の数
  vector[N] y; //データ
}
parameters{
  simplex[K] pi;
  real mu[K];
  real<lower=0> sigma[K];
}
model{
  real logprob[K];
  for(n in 1:N){
    for(k in 1:K){
      logprob[k] = log(pi[k])+normal_lpdf(y[n]|mu[k],sigma[k]);
    }
    target += log_sum_exp(logprob);
  }
}
```

データ生成

- 混合率0.4と0.6の2つの正規分布

```
y <- rnorm(100,5,2)
x <- rnorm(150,15,4)
z <- c(y,x)

tempd <- function(z){
  0.4*dnorm(z,5,2)+0.6*dnorm(z,15,4)
}
hist(z,freq=FALSE,ylim=c(0,0.10))
curve(tempd,add=TRUE,lty=2)
```



  - 真値は
    - 左側はπ=0.4，平均=5，SD=2
    - 右側はπ=0.6，平均=15，SD=4

混合分布モデルとMCMCの問題

- ラベルスウィッチング問題
  - 1~K個の正規分布を推定するが，どのラベルがどの正規分布対応するかは，コントロールが難しい
  - マルコフ連鎖を複数走らせた場合，連鎖ごとにラベルの対応が異なってしまい，事後分布が収束していないと判断されてしまう（Rhatが悪くなる）

- 多峰性問題
  - 近くの分布の平均値同士が合わさってしまい，独立な分布として事後分布が推定されない

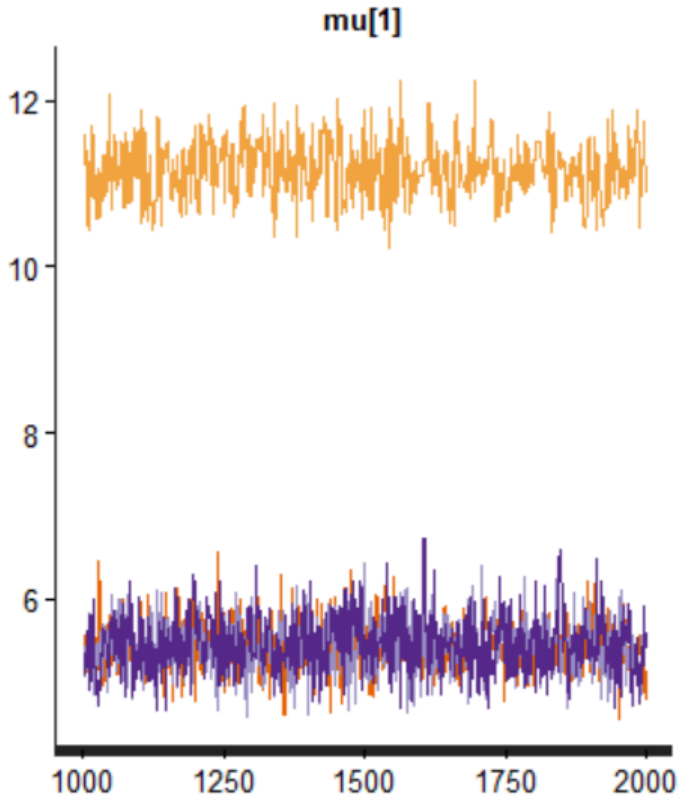
MCMCで走らせた結果

	mean	se_mean	sd	2.5%	50%	97.5%	n_eff	Rhat
pi[1]	5.600000e-01	0.18	0.25	0.33	0.44	1.000000e+00	2	6.74
pi[2]	4.400000e-01	0.18	0.25	0.00	0.56	6.700000e-01	2	6.74
mu[1]	6.880000e+00	1.75	2.49	4.91	5.56	1.159000e+01	2	8.57
mu[2]	-1.288360e+03	1605.64	2529.45	-8460.57	15.11	1.625000e+01	2	2.34
sigma[1]	3.010000e+00	1.18	1.68	1.66	2.14	6.230000e+00	2	7.30
sigma[2]	2.317324e+307	Inf	Inf	3.28	4.07	1.619936e+308	4000	NaN
lp__	-5.964600e+02	205.08	290.10	-767.39	-763.04	-9.249000e+01	2	201.03

Rhatがかなり悪い

トレースプロット

- 連鎖ごとにラベルが異なっている



連鎖を1つにすると・・・

- うまくいくこともある

	mean	se_mean	sd	2.5%	50%	97.5%	n_eff	Rhat
pi[1]	0.41	0.00	0.05	0.32	0.41	0.50	295	1
pi[2]	0.59	0.00	0.05	0.50	0.59	0.68	295	1
mu[1]	5.40	0.02	0.30	4.89	5.37	6.02	304	1
mu[2]	15.24	0.03	0.55	14.02	15.27	16.25	323	1
sigma[1]	2.01	0.01	0.24	1.61	2.00	2.51	335	1
sigma[2]	3.98	0.02	0.41	3.22	3.96	4.83	294	1
lp__	-763.88	0.07	1.52	-767.79	-763.56	-761.96	486	1

変分ベイズによる推定

- 変分近似を用いてベイズ推定を行う
  - 同時事後分布が，すべての周辺事後分布の積で近似できると仮定
  - パラメータがすべて独立だという仮定

- vb()を使って推定する
  - Stanモデルをvb()にいれる
  - fit.mix <- vb(model.mix, data=datastan)

結果

	mean	sd	2.5%	25%	50%	75%	97.5%
pi[1]	0.58	0.04	0.50	0.55	0.58	0.60	0.65
pi[2]	0.42	0.04	0.35	0.40	0.42	0.45	0.50
mu[1]	15.21	0.36	14.48	14.97	15.21	15.44	15.95
mu[2]	5.45	0.24	4.98	5.29	5.44	5.60	5.97
sigma[1]	3.84	0.31	3.28	3.63	3.83	4.04	4.46
sigma[2]	2.05	0.18	1.73	1.92	2.04	2.16	2.40
lp__	0.00	0.00	0.00	0.00	0.00	0.00	0.00

平均値は上手く推定できている
しかし，事後分布の*SD*はかなり小さくなっている

混合分布モデルまとめ

- MCMCだとうまくいかないことが多い
  - しかも，データが大きいか変数が多いと，ものすごい時間がかかって，あまり現実的ではない
- 変分ベイズも視野に入れる
  - ただ，SDが若干小さくなることがある
  - パラメータが独立であると仮定しやすいモデルだと使えるかもしれない
    - 回帰分析などは致命的にSDが小さくなることがある