回帰 (regression)

- 黒木玄

- 2022-07-13〜2022-07-18, 2023-03-23, 2024-01-06, 2024-07-10, 2025-05-19

このノートではJulia言語 (https://julialang.org/)を使用している:

- Julia言語のインストールの仕方の一例 (https://nbviewer.org/github/genkuroki/msfd28/blob/master/install.ipynb)

自明な誤りを見つけたら, 自分で訂正して読んで欲しい. 大文字と小文字の混同や書き直しが不完全な場合や符号のミスは非常によくある.

このノートに書いてある式を文字通りにそのまま読んで正しいと思ってしまうとひどい目に会う可能性が高い. しかし, 数学が使われている文献には大抵の場合に文字通りに読むと間違っている式や主張が書いてあるので, 内容を理解した上で訂正しながら読んで利用しなければいけない. 実践的に数学を使う状況では他人が書いた式をそのまま信じていけない.

このノートの内容よりもさらに詳しいノートを自分で作ると勉強になるだろう. 膨大な時間を取られることになるが, このノートの内容に関係することで飯を食っていく可能性がある人にはそのためにかけた時間は無駄にならないと思われる.

このノートブックはGoogle Colabで実行できる (https://colab.research.google.com/github/genkuroki/Statistics/blob/master/2022/14%20Regression.ipynb).

目次

- 1 回帰 (regression)
  - 1.1 回帰の超一般論
  - 1.2 データの数値から値が決まるパラメータ x がない場合
  - 1.3 回帰の例
- 2 線形回帰
  - 2.1 線形回帰のデータの形
  - 2.2 線形回帰モデルの構成要素
  - 2.3 デザイン行列 (計画行列, design matrix)
  - 2.4 線形回帰の統計モデルの正規分布による記述
  - 2.5 正規分布で書かれた統計モデルの最尤法から最小二乗法による線形回帰が得られること
  - 2.6 直交射影の公式
  - 2.7 βとσ²の不偏推定量
  - 2.8 例: 母平均の推定の場合
  - 2.9 例: 単回帰の場合
  - 2.10 Julia言語による回帰直線の計算の最も簡単な例
  - 2.11 多変量正規分布の定義
  - 2.12 問題: 多変量正規分布とχ²分布の関係
  - 2.13 真の回帰函数と推定された回帰函数
  - 2.14 (真の)回帰直線の値の信頼区間 (標準正規分布版)
  - 2.15 (真の)回帰直線の値の信頼区間 (t分布版)
  - 2.16 予測区間
  - 2.17 「回帰函数の値の信頼区間」と「予測区間」の違い
- 3 線形回帰の計算例
  - 3.1 信頼区間と予測区間のプロット
    - 3.1.1 信頼区間と予測区間のテストプロット
    - 3.1.2 回帰直線の信頼区間と予測区間
    - 3.1.3 多項式回帰の信頼区間と予測区間 (オーバーフィッティングの例)
  - 3.2 信頼区間と予測区間に対応するP値函数のプロット
    - 3.2.1 信頼区間と予測区間に対応するP値函数のテストプロット
    - 3.2.2 回帰直線の信頼区間と予測区間に対応するP値函数
    - 3.2.3 回帰直線の信頼区間に対応するP値函数の動画
    - 3.2.4 多項式回帰の信頼区間と予測区間に対応するP値函数
- 4 ロジスティック回帰
  - 4.1 ロジスティック函数とロジット函数
  - 4.2 ロジスティック回帰のデータの形
  - 4.3 ロジスティック回帰でのリンク函数
  - 4.4 ロジスティック回帰の統計モデル
  - 4.5 最尤法
  - 4.6 スコア統計量とFisher情報量行列
  - 4.7 問題: 一般の場合のスコア統計量とFisher情報量行列
  - 4.8 βの最尤推定量の分布の正規分布近似
  - 4.9 ロジスティック回帰における $\beta_0+\beta_1 x$ に関するWald型のP値函数と信頼区間
  - 4.10 ロジスティック回帰における β_1 に関するWald型のP値函数と信頼区間

4.11 ロジスティック回帰における β_1 に関するWald型のP値函数と信頼区間の動画
5 x_i 達の値も1または0の場合のロジスティック回帰
5.1 x_i 達の値も1または0の場合にロジスティック回帰モデルは2つの二項分布モデルに等しい
5.2 x_i 達の値も1または0の場合のスコア統計量とFisher情報量行列
5.3 x_i 達の値も1または0の場合のWald型のP値函数と信頼区間
5.4 x_i 達の値も1または0の場合のWilson型のP値函数と信頼区間
5.4.1 A=0で定まる条件付き確率分布の正規分布近似
5.4.2 与えられた対数オッズ比パラメータの値 β_1 に対する β_0 の推定量に関する公式
5.4.3 対数オッズ比パラメータ β_1 に関するWilson型のP値函数と信頼区間の構成
5.5 x_i 達の値も1または0の場合にロジスティック回帰の一般化の役に立ち方

In [1]:

```julia
# Google Colabと自分のパソコンの両方で使えるようにするための工夫

import Pkg

"""すでにPkg.add済みのパッケージのリスト (高速化のために用意)"""
_packages_added = [info.name for (uuid, info) in Pkg.dependencies() if info.is_direct_dep]

"""_packages_added内にないパッケージをPkg.addする"""
add_pkg_if_not_added_yet(pkg) = if !(pkg in _packages_added)
    println(stderr, "# $(pkg).jl is not added yet, so let's add it.")
    Pkg.add(pkg)
end

"""expr::Exprからusing内の`.`を含まないモジュール名を抽出"""
function find_using_pkgs(expr::Expr)
    pkgs = String[]
    function traverse(expr::Expr)
        if expr.head == :using
            for arg in expr.args
                if arg.head == :. && length(arg.args) == 1
                    push!(pkgs, string(arg.args[1]))
                elseif arg.head == :(:) && length(arg.args[1].args) == 1
                    push!(pkgs, string(arg.args[1].args[1]))
                end
            end
        else
            for arg in expr.args arg isa Expr && traverse(arg) end
        end
    end
    traverse(expr)
    pkgs
end

"""必要そうなPkg.addを追加するマクロ"""
macro autoadd(expr)
    pkgs = find_using_pkgs(expr)
    :(add_pkg_if_not_added_yet.($(pkgs)); $expr)
end

isdir("images") || mkdir("images")
ENV["LINES"], ENV["COLUMNS"] = 100, 100
using Base.Threads
using LinearAlgebra
using Printf
using Random
Random.seed!(4649373)

@autoadd begin
#using BenchmarkTools
#using DataFrames
using Distributions
#using Memoization
using Optim
using QuadGK
using RCall
#using Roots
#using SpecialFunctions
using StaticArrays
#using StatsBase
using StatsFuns
using StatsPlots
default(fmt = :png, size = (400, 250),
    titlefontsize = 10, guidefontsize=9, plot_titlefontsize = 12)
#using SymPy
end

@rimport stats as R
```

In [2]:

```
#=
# Override https://github.com/jverzani/SymPyCore.jl/blob/main/src/SymPy/show_sympy.jl#L31-
@eval SymPy begin
function Base.show(io::IO,  ::MIME"text/latex", x::SymbolicObject)
    out = _sympy_.latex(↓(x), mode="inline",fold_short_frac=false)
    out = replace(out, r"\\frac{"⇒"\\dfrac{")
    print(io, string(out))
end
end
=#
```

In [3]:

```
safemul(x, y) = x == 0 ? x : isinf(x) ? typeof(x)(Inf) : x*y
safediv(x, y) = x == 0 ? x : isinf(y) ? zero(y) : x/y

x ≲ y = x < y || x ≈ y

mypdf(dist, x) = pdf(dist, x)
mypdf(dist::DiscreteUnivariateDistribution, x) = pdf(dist, round(Int, x))

distname(dist::Distribution) = replace(string(dist), r"{.*}" ⇒ "")
myskewness(dist) = skewness(dist)
mykurtosis(dist) = kurtosis(dist)
function standardized_moment(dist::ContinuousUnivariateDistribution, m)
    μ, σ = mean(dist), std(dist)
    quadgk(x → (x - μ)^m * pdf(dist, x), extrema(dist)...)[1] / σ^m
end
myskewness(dist::MixtureModel{Univariate, Continuous}) =
    standardized_moment(dist, 3)
mykurtosis(dist::MixtureModel{Univariate, Continuous}) =
    standardized_moment(dist, 4) - 3
```

Out[3]: `mykurtosis (generic function with 2 methods)`

1 回帰 (regression)

このノートでは **回帰** (regression)について簡単に説明する.

統計学入門の教科書でこのノートにあるような解説が十分に書いてあるものを見つけることができなかったので, このノートの解説は貴重なものになる可能性がある.

1.1 回帰の超一般論

回帰 (regression)は, データの数値が

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

の形式で,

各々の y_i が数値 x_i に依存して確率的に決まっている

と考えられる状況を, パラメータ $x = (x_1, \dots, x_n)$, $\theta = (\theta_1, \dots, \theta_d)$ を持つ $y = (y_1, \dots, y_n)$ に関する確率密度函数(または確率質量函数)

$$p(y|x, \theta) \quad (\text{または } P(y|x, \theta))$$

でモデル化したときのパラメータ θ の推定の問題として, 一般的に定式化可能である.

この形の統計モデルを(最も一般化された) **回帰モデル** と呼ぶことにする.

1.2 データの数値から値が決まるパラメータ x がない場合

$x = (x_1, \dots, x_n)$ がない場合として回帰ではない統計モデルが得られる.

例: 比率の信頼区間の計算で使われる統計モデルは, 二項分布モデル

$$P(k|n, p) = \binom{n}{k} p^k (1-p)^{n-k} \quad (k = 0, 1, \dots, n)$$

であった. この場合には y にあたる変数が k で, x にあたる変数はない.

例: 平均の信頼区間の t 分布を使った計算法の基礎になる統計モデルは, 正規分布の標本分布

$$p(y|\mu,\sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2\right)$$
$$(\mu,\sigma\in\mathbb{R},\ \sigma>0,\ y=(y_1,\dots,y_n)\in\mathbb{R}^n)$$

であるとみなせる(実際には中心極限定理による近似がうまく行っているという弱い仮定のもとで, その平均の信頼区間は実用的に使用可能). この場合には上の v にあたる変数がこの場合の v で, x にあたる変数はない.

1.3 回帰の例

例: 線形回帰 (linear regression)の最も簡単な単回帰の場合の統計モデルは次になる:

$$p(y|x,\beta_0,\beta_1,\sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-(\beta_0+\beta_1x_i))^2\right).$$

パラメータ $\beta_0,\beta_1,\sigma\in\mathbb{R}$, $\sigma>0$ の推定は線形回帰と呼ばれる. この場合には, 最尤法によるパラメータの推定は最小二乗法そのものになる. **例終**

例: ロジスティック回帰 (logistic regression)の最も簡単な場合の統計モデルは次になる:

$$\begin{aligned}P(y|x,\beta) &= \prod_{i=1}^{n}\left(p_i^{y_i}(1-p_i)^{1-y_i}\right)\\ &= \prod_{i=1}^{n}\left(f(\beta_0+\beta_1x_i)^{y_i}(1-f(\beta_0+\beta_1x_i))^{1-y_i}\right) \qquad (y_i\in\{1,0\}).\end{aligned}$$

ここで

$$f(t) = \mathrm{logistic}(t) = \frac{1}{1+e^{-t}}, \quad p_i = f(\beta_0+\beta_1x_i)$$

である. この統計モデルは各 x_i ごとに, y_i が 1 になる確率 p_i が x_i から

$$p_i = \mathrm{logistic}(\beta_0+\beta_1x_i) = \frac{1}{1+\exp(-(\beta_0+\beta_1x_i))}$$

で決まることを意味している. これは

$$\mathrm{logit}(p_i) = \log\frac{p_i}{1-p_i} = \beta_0+\beta_1x_i.$$

と同値である. **例終**

以下の節ではこれらの回帰について詳しく説明する.

2 線形回帰

以下の説明よりももっと複雑な場合も線形回帰として扱えるが, 基本になる簡単な場合のみを扱う.

2.1 線形回帰のデータの形

データは以下の形で得られると仮定する:

$$(x_1,y_1),\dots,(x_n,y_n)\in\mathbb{R}^2.$$

これを2つのベクトルで表す:

$$x = \begin{bmatrix}x_1\\ \vdots\\ x_n\end{bmatrix}, \quad y = \begin{bmatrix}y_1\\ \vdots\\ y_n\end{bmatrix}.$$

2.2 線形回帰モデルの構成要素

$r<n$ であると仮定する.

r 個の函数 $f_1,\dots,f_r:\mathbb{R}\to\mathbb{R}$ が与えられているとし, その一次結合を決める **回帰係数** パラメータ $\beta_1,\dots,\beta_r\in\mathbb{R}$ が与えられているとし,

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_r \end{bmatrix}, \quad f(x_i) = \begin{bmatrix} f_1(x_i) \\ \vdots \\ f_r(x_i) \end{bmatrix}$$

とおく. このとき,

$$f(x_i)^T \beta = \begin{bmatrix} f_1(x_i) & \cdots & f_r(x_i) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_r \end{bmatrix} = \sum_{j=1}^{r} \beta_j f_j(x_i)$$

ここで, $f(x_i)^T$ は縦ベクトル $f(x_i)$ を転置して得られる横ベクトルを意味する.

さらに, 分散パラメータ $\sigma^2 > 0$ が与えられているとする.

例: $r = 2$ で $f_j(x_i)$, β_j のインデックス j を $1, 2$ ではなく, $0, 1$ を動かすことにする. このとき,

$$f_0(x_i) = 1, \quad f_1(x_i) = x_i$$

の場合は次のようになる.

2.3 デザイン行列 (計画行列, design matrix)

デザイン行列 X を次のように定める:

$$\begin{aligned} X &= \begin{bmatrix} f(x_1) & f(x_2) & \cdots & f(x_n) \end{bmatrix}^T \\ &= \begin{bmatrix} f(x_1)^T \\ f(x_2)^T \\ \vdots \\ f(x_n)^T \end{bmatrix} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_r(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_r(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_r(x_n) \end{bmatrix}. \end{aligned}$$

すなわち, 以上の文脈において, デザイン行列 X は r 次元の縦ベクトル $f(x_i)$ 達を横に n 個並べてできる行列の転置として定義され, $n \times r$ 行列になる. $r < n$ と仮定していたので, X は縦方向に長い行列になる.

このとき,

$$X\beta = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_r(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_r(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_r(x_n) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_r \end{bmatrix} = \left[\sum_{j=1}^{r} \beta_j f_j(x_i) \right]_{i=1}^{n}.$$

以下では簡単のため, デザイン行列 X のランクは可能な最大値である r であると仮定する. (ほとんどの場合にそうなる.)

このとき $\mathbb{R}^n$ の部分空間 $X\mathbb{R}^r = \{ X\beta \mid \beta \in \mathbb{R}^r \}$ の次元は r になり, 行列 X の定める線形写像 $X : \mathbb{R}^r \to \mathbb{R}^n$, $\beta \mapsto X\beta$ は単射になる.

さらに, そのとき $r \times r$ 行列 $X^T X$ が可逆になることも示せる. ここで X^T は X の転置を表す.

例: $r = 2$ で $f_j(x_i)$, β_j のインデックス j は $1, 2$ ではなく, $0, 1$ を動くことにし,

$$f_0(x_i) = 1, \quad f_1(x_i) = x_i$$

と仮定する. このとき, デザイン行列 X は次のようになる:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}.$$

このとき, $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$ ならば

$$X\beta = \begin{bmatrix} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ \cdots \\ \beta_0 + \beta_1 x_n \end{bmatrix}.$$

以下の節を最初に読む場合にはこの例の場合に限定して考えてもよい. より一般的な場合についてはこの特別な場合について理解した後に考えてもよい.

2.4 線形回帰の統計モデルの正規分布による記述

統計モデルとして, $y \in \mathbb{R}^n$ に関する次の確率密度函数を採用する:

$$p(y|X, \beta, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\|y - X\beta\|^2\right).$$

ここで, $\| \ \|^2$ は通常のEuclidノルムの2乗(成分の2乗の和)を意味する:

$$\|y - X\beta\|^2 = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{r} \beta_j f_j(x_i)\right)^2.$$

$Y = [Y_i]_{i=1}^n$ をこの統計モデルに従うベクトル値確率変数であると仮定する. Y は次のように表される:

$$Y = X\beta + \varepsilon$$

ここで, ベクトル値確率変数 $\varepsilon = [\varepsilon_i]_{i=1}^n$ の成分達 $\varepsilon_1, \dots, \varepsilon_n$ はそれぞれが平均 0, 分散 σ^2 の正規分布に従う独立な確率変数達になる. ゆえに,

$$E[Y] = X\beta, \quad E[\varepsilon\varepsilon^T] = \sigma^2 I.$$

ここで I は n 次の単位行列を表す.

このことは一般に次が成立していることから確かめられる:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \cdots & a_1 b_n \\ a_2 b_1 & a_2 b_2 & \cdots & a_2 b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_n b_1 & a_n b_2 & \cdots & a_n b_n \end{bmatrix}.$$

(沢山の数をまとめて1つの記号で表すベクトルや行列の記号法は複雑な式を簡潔に表現するために非常に強力であるが, 簡潔になり過ぎてしまい, 初学者にとって何をやっているのか, 分かり難くなってしまうことも多い. そういう状況に自分が陥った場合には, すべての成分を書き下して, 具体的な式を書いてみると納得できることが多い. 簡潔で抽象的な式を見たら, 面倒臭く具体的な式を書くと理解し易くなることがあり, 逆に面倒臭く具体的に書かれた式を, ベクトルや行列をうまく使って簡潔に書き直すと理解し易くなることもある. 解説文の通りの式の表記にこだわる必然性はまったくない.)

このモデル内では, 仮想的なデータの数値 Y_i が $f_j(x_i)$ 達の一次結合 $\sum_{j=1}^r \beta_j f_j(x_i)$ で近似され, それらの差(残差と呼ばれる)

$$Y_i - \sum_{j=1}^{r} \beta_j f_j(x_i)$$

が平均 0, 分散 σ^2 の正規分布に従って独立にランダムに決まっている:

$$Y_i - \sum_{j=1}^{r} \beta_j f_j(x_i) \sim_{\text{i.i.d.}} \text{Normal}(0, \sigma) \quad (i = 1, 2, \dots, n).$$

注意: 以上の設定において Y はベクトル値確率変数だが, デザイン行列 X は確率変数では **ない**.

2.5 正規分布で書かれた統計モデルの最尤法から最小二乗法による線形回帰が得られること

上の統計モデルのデータの数値 x, y に関する尤度函数(ゆうどかんすう, 尤度はデータの数値と統計モデルの適合度(フィットの良さ)の指標の1つ)

$$(\beta, \sigma^2) \mapsto p(y|X, \beta, \sigma^2)$$

を最小化するパラメータ β, σ^2 の値から, 最小二乗法が得られることを説明しよう. (デザイン行列 X はベクトル x から決まる.)

尤度函数(ゆうどかんすう, モデルのパラメータとデータの数値の適合度の指標の1つ)

$$(\beta, \sigma^2) \mapsto p(y|X, \beta, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\|y - X\beta\|^2\right)$$

の対数の -2 倍は次の形になる:

$$L(\beta, \sigma^2) = -2 \log p(y|X, \beta, \sigma^2) = \frac{1}{\sigma^2}\|y - X\beta\|^2 + n \log \sigma^2 + n \log(2\pi).$$

これを最小化するパラメータ β, σ^2 の値を求めよう.

データの数値として与えられた y とパラメータ β に依存する $X\beta$ の距離の2乗 $\|y - X\beta\|^2$ を最小にするベクトル $X\beta$ はベクトル $y \in \mathbb{R}^n$ の部分空間 $X\mathbb{R}^r$ への直交射影 $\hat{y}$ になる.

そのとき, $X : \mathbb{R}^r \to \mathbb{R}^n$ は単射なので $\hat{y} = X\hat{\beta}$, $\hat{\beta} \in \mathbb{R}^r$ と一意に表される.

以下の図を見よ.

回帰係数 β をそのような $\hat{\beta}$ として求める方法は, 距離の2乗 $\|y - X\beta\|^2$ を最小化するので, **最小二乗法** と呼ばれる.

以上の議論から, **最小二乗法は直交射影そのものに過ぎない** ことがわかる.

$y - \hat{y} = y - X\hat{\beta}$ を **残差** (residual error)と呼ぶ.

$\hat{\sigma}^2$ を y とその直交射影 $\hat{y}$ の距離の2乗の n 分の 1 と定める:

$$\hat{\sigma}^2 = \frac{1}{n}\|y - \hat{y}\|^2 = \frac{1}{n}\|y - X\hat{\beta}\|^2.$$

このとき,

$$L(\hat{\beta}, \sigma^2) = \frac{n\hat{\sigma}^2}{\sigma^2} + n\log\sigma^2 + n\log(2\pi).$$

を最小化する σ^2 は $\sigma^2 = \hat{\sigma}^2$ になることを確認できる.

(一般に $\sigma^2 > 0$ の函数 $\sigma^2 \mapsto a/\sigma^2 + \log\sigma^2$ $(a > 0)$ は $\sigma^2 = a$ で最小になる. 実際, $f(s) = a/s + \log s$ について $f'(s) = (s - a)/s^2$ なので, $s < a$ のとき $f'(s) < 0$ で $f(s)$ は単調減少し, $s > a$ のとき $f'(s) > 0$ で $f(s)$ は単調増加する. ゆえに $f(s)$ は $s = a$ で最小になる.)

以上のようにして, 直交射影で求めた $\hat{\beta}$ と $\hat{\sigma}^2$ は最尤法の解になっている.

In [4]:

```
plot(s -> 3/s + log(s), 2, 4; label="",
    title="f(s) = 3/s + log(s)", xguide="s")
```

Out[4]:

2.6 直交射影の公式

ベクトル y の部分空間 $X\mathbb{R}^r$ への直交射影 $\hat{y} = X\hat{\beta}$ の具体的な形を求めよう.

ベクトル $y - \hat{y} = y - X\hat{\beta}$ は部分空間 $X\mathbb{R}^r$ と直交するので, 任意の $\gamma \in \mathbb{R}^r$ について

$$0 = (X\gamma, y - X\hat{\beta}) = (X\gamma)^T(y - X\hat{\beta}) = \gamma^T X^T(y - X\hat{\beta}).$$

ここで, $(\ ,\ )$ は通常の内積(成分の積の和)を表す. ベクトル $\gamma \in \mathbb{R}^r$ は任意なので,

$$X^T(y - X\hat{\beta}) = 0, \quad \text{すなわち} \quad X^T X\hat{\beta} = X^T y.$$

X のランクが可能な最大値 r になると仮定していたので, X^TX は可逆になるのであった. ゆえに

$$\hat{\beta} = (X^TX)^{-1}X^Ty, \quad \hat{y} = X(X^TX)^{-1}X^Ty.$$

これと,

$$\hat{\sigma}^2 = \frac{1}{n}\|y - X\hat{\beta}\|^2$$

を合わせると, 尤度函数 $(\beta, \sigma^2) \mapsto p(y|X, \beta, \sigma^2)$ を最大化するパラメータ値を計算できる(コンピュータ上に実装できる).

これが正規分布で記述された統計モデルの最尤法の解である.

注意: 以上の議論によって, 最小二乗法は本質的に直交射影を作る操作であることも分かる. このように内積に関する線形代数を理解していれば, その中に最小二乗法の理論も含まれていると考えることができる. 線形代数は普遍的に役に立つ道具である.

2.7 βとσ²の不偏推定量

以下では, $\hat{\beta}$ と $\hat{s}^2 = (n/(n-r))\hat{\sigma}^2 = (1/(n-r))\|y - X\hat{\beta}\|^2$ がそれぞれ β と σ^2 の不偏推定量になっていることを示す.

この節では, 前々節で記述した統計モデルに従う確率変数 Y とは別に, ベクトル値確率変数 $y = [y_i]_{i=1}^n$ で次の条件を満たすものを使用する:

$$y = X\beta + e, \quad E[e] = 0, \quad E[ee^T] = \sigma^2 I.$$

このようなベクトル値確率変数 y を使うことは, 正規分布で記述された統計モデルの設定を大幅に弱めることを意味している. この節ではそのような状況を扱う.

上の条件中の $E[ee^T] = \sigma^2 I$ は e の分散共分散行列であることに注意せよ.

(一般に n 次元ベクトル値確率変数 $V = [V_i]_{i=1}^n$ について, $\mu = E[V]$ のとき, $n \times n$ の対称行列 $E[(V-\mu)(V-\mu)^T]$ を V の分散共分散行列 (variance-covariance matrix) と呼ぶ. その (i,i) 成分は V_i の分散になっており, $i \neq j$ に関する (i,j) 成分は V_i と V_j の共分散になっている.)

$e = [e_i]_{i=1}^n$ と書くとき, $E[ee^T] = \sigma^2 I$ は

$$E[e_ie_j] = \sigma^2\delta_{ij}$$

が成立することを意味している. ここで δ_{ij} はKroneckerのデルタである. ($i = j$ のとき $\delta_{ij} = 1$ で $i \neq j$ のとき $\delta_{ij} = 0$.)

上の仮定の下で, y の平均と分散共分散行列はそれぞれ次のようになる:

$$E[y] = X\beta, \quad E[(y - X\beta)(y - X\beta)^T] = E[ee^T] = \sigma^2 I.$$

この設定における最小二乗法は次のように書ける:

$$\hat{\beta} = (X^TX)^{-1}X^Ty, \quad \hat{\sigma}^2 = \frac{1}{n}\|y - X\hat{\beta}\|^2.$$

このとき,

$$E[\hat{\beta}] = (X^TX)^{-1}X^TX\beta = \beta$$

なので, $\hat{\beta}$ は β の不偏推定量になっている.

以下では σ^2 の不偏推定量を構成しよう.

$$\hat{\beta} - \beta = (X^TX)^{-1}X^T(X\beta + e) - \beta = (X^TX)^{-1}X^Te$$

より,

$$(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T = (X^TX)^{-1}X^Tee^TX(X^TX)^{-1}$$

なので, ベクトル値確率変数 $\hat{\beta} = (X^TX)^{-1}X^Ty$ の分散共分散行列は次のようになる:

$$E[(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T] = (X^TX)^{-1}X^TE[ee^T]X(X^TX)^{-1} = \sigma^2(X^TX)^{-1}.$$

2つ目の等号で $E[ee^T] = \sigma^2 I$ を使った.

$$X\hat{\beta} - X\beta = X(\hat{\beta} - \beta), \quad (X\hat{\beta} - X\beta)(X\hat{\beta} - X\beta)^T = X(\hat{\beta} - \beta)(\hat{\beta} - \beta)^TX^T$$

より, $X\hat{\beta}$ の分散共分散行列は次のように計算される:

$$E[(X\hat{\beta} - X\beta)(X\hat{\beta} - X\beta)^T] = XE[(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T]X^T = \sigma^2X(X^TX)^{-1}X^T.$$

2つ目の等号で, 上で示した $E[(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T] = \sigma^2(X^TX)^{-1}$ を使った.

$X(X^TX)^{-1}X^T$ は $\mathbb{R}^n$ からその部分空間 $X\mathbb{R}^r$ への直交射影を与える行列であった. (特に対称行列でかつ二乗しても不変であること $\left(X(X^TX)^{-1}X^T\right)^2 = X(X^TX)^{-1}X^T$ にも注意せよ.)

$$
\begin{aligned}
y - X\hat{\beta} &= y - X(X^TX)^{-1}X^Ty = (I - X(X^TX)^{-1}X^T)y \\
E[y - X\hat{\beta}] &= E[y] - X(X^TX)^{-1}X^TE[y] = X\beta - X(X^TX)^{-1}X^TX\beta = 0
\end{aligned}
$$

より, 残差 $y - X\hat{\beta}$ の分散共分散行列は次のように計算される:

$$
\begin{aligned}
E[(y - X\hat{\beta})(y - X\hat{\beta})^T] &= (I - X(X^TX)^{-1}X^T)E[yy^T](I - X(X^TX)^{-1}X^T)^T \\
&= \sigma^2(I - X(X^TX)^{-1}X^T)(I - X(X^TX)^{-1}X^T)^T \\
&= \sigma^2(I - X(X^TX)^{-1}X^T).
\end{aligned}
$$

ここで, $E[yy^T] = \sigma^2 I$ および, $X(X^TX)^{-1}X^T$ が $\mathbb{R}^n$ から $X\mathbb{R}^r$ への直交射影を与える行列であったことから, $I - X(X^TX)^{-1}X^T$ が $\mathbb{R}^n$ から $X\mathbb{R}^r$ の直交補空間への射影を与える行列になり, 特に対称行列でかつ二乗しても不変になることを使った.

(一般に $P^T = P$, $P^2 = P$ のとき, $(I - P)^T = I - P$ かつ $(I - P)^2 = I - P$ となる.)

トレースの中で行列の順序を巡回的に回してもトレースの値は不変なので,

$$
\mathrm{tr}(X(X^TX)^{-1}X^T) = \mathrm{tr}((X^TX)^{-1}X^TX) = \mathrm{tr}(I_r) = r.
$$

ここで I_r は r 次の単位行列を表す. この結果と

$$
\|y - X\hat{\beta}\|^2 = (y - X\hat{\beta})^T(y - X\hat{\beta}) = \mathrm{tr}\left((y - X\hat{\beta})(y - X\hat{\beta})^T\right)
$$

を使うと,

$$
\begin{aligned}
E[\|y - X\hat{\beta}\|^2] &= \mathrm{tr}\left(E[(y - X\hat{\beta})(y - X\hat{\beta})^T]\right) \\
&= \sigma^2\,\mathrm{tr}(I - X(X^TX)^{-1}X^T) = (n - r)\sigma^2.
\end{aligned}
$$

2つめの等号で上で示した $E[(y - X\hat{\beta})(y - X\hat{\beta})^T] = \sigma^2(I - X(X^TX)^{-1}X^T)$ を用い, 3つ目の等号で上で示した $\mathrm{tr}(X(X^TX)^{-1}X^T) = r$ を使った.

ゆえに,

$$
\hat{s}^2 = \frac{1}{n - r}\|y - X\hat{\beta}\|^2 = \frac{n}{n - r}\hat{\sigma}^2
$$

とおくと,

$$
E[\hat{s}^2] = \sigma^2.
$$

すなわち, $\hat{s}^2 = \|y - X\hat{\beta}\|^2/(n - r)$ は σ^2 の不偏推定量である.

σ^2 の不偏推定量を作るためには残差の二乗和 $\|y - X\hat{\beta}\|^2$ を n ではなく, $n - r$ で割らなければいけない.

注意: 特にこの節を見れば分かるように, 線形回帰の理解のために最も重要な基礎知識は線形代数である. 特に直交射影に関する線形代数が重要になる.

2.8 例: 母平均の推定の場合

線形回帰は母平均の推定を含む.

$r = 1$ とし, $f_j(x_i)$ と β_j のインデックス j として 1 ではなく 0 を使うことにし,

$$
f_0(x_i) = 1, \quad \beta_0 = \mu, \quad \hat{\beta}_0 = \hat{\mu}
$$

の場合について考える. このとき, デザイン行列 X は n 次元縦ベクトルになり,

$$
X = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad X\beta = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix}.
$$

ゆえに, $X^TX = n$, $X^Ty = \sum_{i=1}^n y_i$ となるので,

$$
\hat{\mu} = (X^TX)^{-1}X^Ty = \frac{1}{n}\sum_{i=1}^n y_i =: \bar{y}.
$$

さらに, $y - X\hat{\beta} = [y_i - \hat{\mu}]_{i=1}^n = [y_i - \bar{y}]_{i=1}^n$ より,

$$\hat{\sigma}^2 = \frac{1}{n}\|y - X\hat{\beta}\| = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2.$$

これは, σ^2 を, 不偏推定量になるように $n-1$ で割らずに, n で割って作った場合の標本分散になっている.

前節で示したように σ^2 の不偏推定量 $\hat{s}^2$ は $n - r = n - 1$ で割ることによって得られる:

$$\hat{s}^2 = \frac{1}{n-1}\|y - X\hat{\beta}\| = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2.$$

この結果は標本の不偏分散の話の線形回帰に関する結果を使った再現になっている.

2.9 例: 単回帰の場合

$r = 2$ とし, $f_j(x_i)$ と β_j のインデックス j として $1, 2$ ではなく $0, 1$ を使うことにし,

$$f_0(x_i) = 1, \quad f_1(x_i) = x_i$$

の場合を考える. さらに, 以下のようにおく:

$$\begin{aligned}
\bar{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \\
\overline{x^2} &= \frac{1}{n}\sum_{i=1}^{n} x_i^2, \quad \overline{y^2} = \frac{1}{n}\sum_{i=1}^{n} y_i^2, \quad \overline{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i.
\end{aligned}$$

このとき, x_i 達と y_i 達の不偏補正をしていない標本分散と標本共分散はそれぞれ次のように書ける:

$$\begin{aligned}
\hat{\sigma}_x^2 &:= \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \overline{x^2} - \bar{x}^2, \\
\hat{\sigma}_y^2 &:= \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 = \overline{y^2} - \bar{y}^2, \\
\hat{\sigma}_{xy} &:= \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \overline{xy} - \bar{x}\bar{y}.
\end{aligned} \tag{$*$}$$

このとき, デザイン行列 X については, 以下が成立していることをちょっとした計算で確認できる(自分で確認してみよ):

$$\begin{aligned}
X &= \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad X^T X = n \begin{bmatrix} 1 & \bar{x} \\ \bar{x} & \overline{x^2} \end{bmatrix}, \\
X^T y &= n \begin{bmatrix} \bar{y} \\ \overline{xy} \end{bmatrix}, \quad (X^T X)^{-1} = \frac{1}{n(\overline{x^2} - \bar{x}^2)} \begin{bmatrix} \overline{x^2} & -\bar{x} \\ -\bar{x} & 1 \end{bmatrix}.
\end{aligned}$$

ゆえに,

$$\overline{x^2}\bar{y} - \bar{x}\overline{xy} = \overline{x^2}\bar{y} - \bar{x}^2\bar{y} + \bar{x}^2\bar{y} - \bar{x}\overline{xy} = \hat{\sigma}_x^2\bar{y} - \hat{\sigma}_{xy}\bar{x}$$

が成立していることに注意すれば,

$$\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix} = \hat{\beta} = (X^T X)^{-1} X^T y = \begin{bmatrix} \bar{y} - (\hat{\sigma}_{xy}/\hat{\sigma}_x^2)\bar{x} \\ \hat{\sigma}_{xy}/\hat{\sigma}_x^2 \end{bmatrix}.$$

すなわち, 次の回帰直線の公式が得られた:

$$\hat{\beta}_0 + \hat{\beta}_1 x_* = \bar{y} + \frac{\hat{\sigma}_{xy}}{\hat{\sigma}_x^2}(x_* - \bar{x}).$$

$\hat{\sigma}^2$ の公式も求めよう.

$$y - X\hat{\beta} = (I - X(X^T X)^{-1} X^T)y$$

より,

$$
\begin{aligned}
\|y - X\hat{\beta}\|^2 &= y^T(I - X(X^TX)^{-1}X^T)^T(I - X(X^TX)^{-1}X^T)y \\
&= y^T(I - X(X^TX)^{-1}X^T)y \\
&= y^Ty - (X^Ty)^T(X^TX)^{-1}X^Ty.
\end{aligned}
$$

そして,

$$
\begin{aligned}
&y^Ty = n\overline{y^2}, \\
&(X^Ty)^T(X^TX)^{-1}X^Ty = \frac{n}{\hat{\sigma}_x^2}\left(\overline{x^2} - 2\bar{x}\bar{y}\overline{xy} + \overline{xy}^2\right)
\end{aligned}
$$

および上の(∗)を使って整理すると,

$$
\hat{\sigma}^2 = \frac{1}{n}\|y - X\hat{\beta}\|^2 = \frac{\hat{\sigma}_x^2\hat{\sigma}_y^2 - \hat{\sigma}_{xy}^2}{\hat{\sigma}_x^2}.
$$

前々節の結果より, σ^2 の不偏推定量 $\hat{s}^2$ は n ではなく, $n - r = n - 2$ で割ることによって次のようにして得られる:

$$
\hat{s}^2 = \frac{1}{n-2}\|y - X\hat{\beta}\|^2 = \frac{n}{n-2}\frac{\hat{\sigma}_x^2\hat{\sigma}_y^2 - \hat{\sigma}_{xy}^2}{\hat{\sigma}_x^2}.
$$

x_i 達と y_i 達の不偏分散と不偏共分散を

$$
s_x^2 := \frac{n}{n-1}\hat{\sigma}_x^2, \quad s_y^2 := \frac{n}{n-1}\hat{\sigma}_y^2, \quad s_{xy} := \frac{n}{n-1}\hat{\sigma}_{xy}
$$

と書くと,

$$
\begin{aligned}
&\hat{\beta}_0 = \bar{y} - \frac{\hat{\sigma}_{xy}}{\hat{\sigma}_x^2}\bar{x} = \bar{y} - \frac{s_{xy}}{s_x^2}\bar{x}, \quad \hat{\beta}_1 = \frac{\hat{\sigma}_{xy}}{\hat{\sigma}_x^2} = \frac{s_{xy}}{s_x^2}, \\
&\hat{\sigma}^2 = \frac{n-1}{n}\frac{s_x^2s_y^2 - s_{xy}^2}{s_x^2}, \quad \hat{s}^2 = \frac{n-1}{n-2}\frac{s_x^2s_y^2 - s_{xy}^2}{s_x^2}.
\end{aligned}
$$

以上の公式は

- 「標本分布について」のノート

2.10 Julia言語による回帰直線の計算の最も簡単な例

In [5]:

```julia
# パッケージの読み込みなど
using Distributions
using StatsPlots
using Random
Random.seed!(4649373)

# テストデータのランダム生成
n = 100
x = rand(Normal(0, 1), n);
e = rand(Normal(0, 1), n);
β = [-1, 2]
σ = 2
y = @. β[1] + β[2]*x + σ*e;
scatter(x, y; label="data", legend=:topleft, msc=:auto)
plot!(xstar -> β[1] + β[2]*xstar; label="true model", c=:black, ls=:dash)
```

Out[5]:

In [6]:
```
# デザイン行列
X = x .^ (0:1)';

# デザイン行列の上から5行分を表示
X[1:5, :]
```

Out[6]:
```
5×2 Matrix{Float64}:
 1.0   1.52932
 1.0   0.631079
 1.0  -1.29149
 1.0   0.259151
 1.0  -0.266555
```

In [7]:
```
# 線形回帰
# 以下は β̂ = X'X\X'y もしくは β̂ = inv(X'*X)*X'*y と同値
β̂ = X \ y
```

Out[7]:
```
2-element Vector{Float64}:
 -0.8598014214421746
  2.174733393051476
```

In [8]:
```
inv(X'*X)*X'*y
```

Out[8]:
```
2-element Vector{Float64}:
 -0.8598014214421751
  2.174733393051477
```

In [9]:
```
# 結果をプロット
scatter(x, y; label="data", legend=:topleft, msc=:auto)
plot!(xstar -> β[1] + β[2]*xstar; label="true model", c=:black, ls=:dash)
plot!(xstar -> β̂[1] + β̂[2]*xstar; label="MLE", c=2, lw=1.5)
```

Out[9]:

MLEは最尤推定(さいゆうほう, maximum likelihood estimate)の略である.

Julia言語 (https://julialang.org/)(download (https://julialang.org/downloads/)) の current stable release をダウンロードして, インストールして, 実行して,

```
julia>
```

で] を押した後に,

```
pkg> add Distributions, StatsPlots
```

を実行し,

```
pkg>
```

でバックスペースキーを押して,

```
julia>
```

の状態に上のコードを貼り付ければ上と同じことをできるはずである.

Julia言語では最小二乗法による回帰は `β̂ = X \ y` だけで可能である.

`X'` は `X` の転置(の複素共役)を意味する.

上のコードをコピー＆ペーストすることによって以下のスクリーンショットのように計算して, グラフを作ることができる.

MSYS2 MINGW64 Shell

```
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.7.3 (2022-05-06)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> using Distributions

julia> using StatsPlots

julia> using Random

julia> Random.seed!(4649373)
TaskLocalRNG()

julia> n = 100
σ = 2
y = @. β[1] + β[2]*x + σ*e;
scatter(x, y; label="data", legend=:topleft, msc=:auto)
plot!(xstar -> β[1] + β[2]*xstar; label="true model", c=:black, ls=:dash)100

julia> x = rand(Normal(0, 1), n);

julia> e = rand(Normal(0, 1), n);

julia> β = [-1, 2]
2-element Vector{Int64}:
 -1
  2

julia> σ = 2
2
```

```
julia> σ = 2
2

julia> y = @. β[1] + β[2]*x + σ*e;

julia> scatter(x, y; label="data", legend=:topleft, msc=:auto)


julia> plot!(xstar -> β[1] + β[2]*xstar; label="true model", c=:black, ls=:dash)


julia> X = x .^ (0:1)';

julia> X[1:5, :]
5×2 Matrix{Float64}:
 1.0   0.843695
 1.0   0.286651
 1.0  -0.117361
 1.0   1.43215
 1.0  -1.39703

julia> β = X \ y
2-element Vector{Float64}:
 -1.0404163851842536
  2.235682111246987

julia> scatter(x, y; label="data", legend=:topleft, msc=:auto)

julia> plot!(xstar -> β[1] + β[2]*xstar; label="true model", c=:black, ls=:dash)


julia> plot!(xstar -> β[1] + β[2]*xstar; label="estimate", c=2, lw=1.5)

julia>
```

$\hat{\beta}$ = X \ y の部分の $\hat{\beta}$ が潰れて β と表示されている点は気にしないことにした.

できあがったグラフは以下のように表示されている.

ここまで無事にたどり付けなかった場合には, エラーメッセージをインターネットで検索すれば解決法が見つかることが多い.

2.11 多変量正規分布の定義

$\mu \in \mathbb{R}^n$ と固有値がすべて正の n 次の実対称行列 Σ に対して, 多変量正規分布

$$\mathrm{MvNormal}(\mu, \Sigma)$$

の確率密度函数を次のように定める:

$$p(y|\mu, \Sigma) = \frac{1}{\det(2\pi\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(y-\mu)^T \Sigma^{-1}(y-\mu)\right) \quad (y \in \mathbb{R}^n).$$

例: $\mu = (m, m, \dots, m)$, $\Sigma = \sigma^2 I$ (I は n 次の単位行列で $\sigma^2 > 0$)のとき,

$$p(y|\mu, \Sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - m)^2\right).$$

これは正規分布の標本分布 $\mathrm{Normal}(m, \sigma^2)^n$ の密度函数に等しい.

2.12 問題: 多変量正規分布とχ²分布の関係

前節の n 変量正規分布 $\mathrm{MvNormal}(\mu, \Sigma)$ を考える:

$$p(y|\mu, \Sigma) = \frac{1}{\det(2\pi\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(y-\mu)^T \Sigma^{-1}(y-\mu)\right) \quad (y \in \mathbb{R}^n).$$

この分布に従う n 次元ベクトル値確率変数を Y と書き, 確率変数 χ^2 を

$$\chi^2 = (Y-\mu)^T \Sigma^{-1}(Y-\mu)$$

と定めると, χ^2 は自由度 n のχ²分布に従うことを示せ.

解答例: 固有値達 σ_i^2 がすべて正の実対称行列 Σ はある直交行列 U によって次のように対角化できる:

$$\Sigma = UD^2U^{-1} = UD^2U^T, \quad D = \mathrm{diag}(\sigma_1, \dots, \sigma_n), \quad \sigma_i > 0.$$

このとき,

$$\det(2\pi\Sigma)^{1/2} = \prod_{i=1}^{n} (2\pi\sigma_i^2)^{1/2} = (2\pi)^{n/2}\sigma_1 \cdots \sigma_n.$$

さらに, $x = [x_i]_{i=1}^n = D^{-1}U^T(y-\mu)$ とおくと, $\Sigma^{-1} = UD^{-2}U^T$ より

$$(y-\mu)^T \Sigma^{-1}(y-\mu) = x^T x$$

となり, 直交行列による変換が体積を保つことより,

$$|dy_1 \cdots dy_n| = \sigma_1 \cdots \sigma_n \, |dx_1 \cdots dx_n|$$

となるので,

$$\begin{aligned} p(y|\mu, \Sigma)\,|dy_1 \cdots dy_n| &= \frac{1}{(2\pi)^{n/2}\sigma_1 \cdots \sigma_n} \exp\left(-\frac{1}{2}(y-\mu)^T \Sigma^{-1}(y-\mu)\right) |dy_1 \cdots dy_n| \\ &= \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}x^T x\right) |dx_1 \cdots dx_n| \\ &= \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^2\right) |dx_1 \cdots dx_n|. \end{aligned}$$

ゆえに, 変数 x に対応するベクトル値確率変数を $X = [X_i]_{i=1}^n$ と書くと, $X_1, \dots, X_n$ はそれぞれが標準正規分布に従う独立な確率変数達になり,

$$\chi^2 = (Y-\mu)^T \Sigma^{-1}(Y-\mu) = X^T X = \sum_{i=1}^{n} X_i^2$$

となるので, χ^2 は自由度 n のχ²分布に従う.

解答終

2.13 真の回帰函数と推定された回帰函数

「βとσ²の不偏推定量」の節の仮定の下で, $x_* \in \mathbb{R}$ の函数

$$f(x_*)^T\beta = \sum_{j=1}^{r} \beta_j f_j(x_*)$$

を **真の回帰函数** と呼び,

$$f(x_*)^T\hat{\beta} = \sum_{j=1}^{r} \hat{\beta}_j f_j(x_*)$$

を **推定された回帰函数** と呼ぶことにする.

注意: 真の回帰函数の「真の」の意味は「現実における真の」という意味ではない. 仮想的なデータを生成するために用いたモデルのパラメータの設定を意味するに過ぎない.

2.14 (真の)回帰直線の値の信頼区間 (標準正規分布版)

この節では, 「βとσ²の不偏推定量」の節で仮定した条件に加えて, ベクトル値 $\hat{\beta} = (X^TX)^{-1}X^Ty$ が近似的に多変量正規分布に従っており(この仮定は拡張された中心極限定理によって多くの場合に成立している), $\hat{s}^2 \approx \sigma^2$ という近似が成立していると仮定する.

このとき, 「βとσ²の不偏推定量」の節で示した結果

$$E[\hat{\beta}] = \beta, \quad E[(\hat{\beta}-\beta)(\hat{\beta}-\beta)^T] = \sigma^2(X^TX)^{-1}$$

より, 次の近似が成立している:

$$\hat{\beta} \sim \mathrm{MvNormal}(\beta,\ \hat{s}^2(X^TX)^{-1}),\ \text{approximately}.$$

ゆえに, $x_* \in \mathbb{R}$ について,

$$f(x_*)^T\hat{\beta} \sim \mathrm{Normal}\left(f(x_*)^T\beta,\ \widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}}\right),\ \text{approximately}.$$

ここで,

$$\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}} = \hat{s}\sqrt{f(x_*)^T(X^TX)^{-1}f(x_*)}$$

これより, 真の回帰函数の $x_* \in \mathbb{R}$ の値 $f(x_*)^T\beta$ に関する仮説 $f(x_*)^T\beta = f(x_*)^T\beta_0$ のP値と $f(x_*)^T\beta$ の値の信頼区間を定義できる.

(真の)回帰函数の x_* における値に関する仮説 $f(x_*)^T\beta = f(x_*)^T\beta_0$ のP値の定義: $t(x_*, \beta_0)$ を

$$t(x_*, \beta_0) = \frac{f(x_*)^T\hat{\beta} - f(x_*)^T\beta_0}{\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}}} = \frac{f(x_*)^T\hat{\beta} - f(x_*)^T\beta_0}{\hat{s}\sqrt{f(x_*)^T(X^TX)^{-1}f(x_*)}}$$

と定め, P値を次のように定める:

$$\mathrm{pvalue}_{\mathrm{Normal}}(y|X, f(x_*)^T\beta = f(x_*)\beta_0) = 2(1 - \mathrm{cdf}(\mathrm{Normal}(0,1), |t(x_*, \beta_0)|)).$$

(真の)回帰函数の値 $f(x_*)^T\beta$ に関する信頼度 $1-\alpha$ の信頼区間の定義: $z_{\alpha/2}$ を

$$z_{\alpha/2} = \mathrm{quantile}(\mathrm{Normal}(0,1), 1-\alpha/2).$$

と定め, 信頼区間を次のように定める:

$$\mathrm{confint}_{\mathrm{Normal}}^{f(x_*)^T\beta}(y|X) = \left[f(x_*)^T\hat{\beta} - z_{\alpha/2}\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}},\ f(x_*)^T\hat{\beta} + z_{\alpha/2}\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}}\right].$$

注意: 「真の」は「現実における真の」を意味しない. 単に仮想的なデータを生成するモデルのパラメータの設定を意味するに過ぎない.

2.15 (真の)回帰直線の値の信頼区間 (t分布版)

前節の標準正規分布を使って定義されたP値と信頼区間の t 分布を使った補正を構成しよう.

そのために, 前節までに仮定していた条件よりもさらに強い次の条件を仮定する:

$$e = [e_i]_{i=1}^n \sim \mathrm{Normal}(0,\sigma)^n = \mathrm{MvNormal}(0, \sigma^2 I).$$

これは正規分布で記述されていた統計モデル内の設定に戻ったことを意味している.

このとき,

$$\hat{\beta} \sim \mathrm{MvNormal}(\beta,\ \sigma^2(X^TX)^{-1}).$$

ゆえに, $x_* \in \mathbb{R}$ について,

$$f(x_*)^T\hat{\beta} \sim \mathrm{Normal}\Big(f(x_*)^T\beta,\ \mathrm{SE}_{f(x_*)^T\hat{\beta}}\Big).$$

ここで,

$$\mathrm{SE}_{f(x_*)^T\hat{\beta}} = \sigma\sqrt{f(x_*)^T(X^TX)^{-1}f(x_*)}.$$

さらに,

$$y - X\hat{\beta} = (X\beta + e) - X(X^TX)^{-1}X^T(X\beta + e) = (I - X(X^TX)^{-1}X^T)e$$

が部分空間 $X\mathbb{R}^r$ の直交補空間($n-r$ 次元になる)へのベクトル値確率変数 $e \sim \mathrm{MvNormal}(0, \sigma^2 I)$ の直交射影であることから,

$$\frac{(n-r)\hat{s}^2}{\sigma^2} = \|y - X\hat{\beta}\|^2 \sim \mathrm{Chisq}(n-r)$$

となることを示せる. (「多変量正規分布とχ²分布の関係」を使えば容易に示される.)

$\hat{y} = X\hat{\beta}$ と $y - \hat{y} = y - X\hat{\beta}$ が独立であることより(その独立性はそれらが直交することから導かれる), $f(x_*)^T\hat{\beta}$ と $(n-r)\hat{s}^2/\sigma^2 = \|y - X\hat{\beta}\|^2$ が独立になることも導かれるので, 一般に独立な確率変数達 $Z \sim \mathrm{Normal}(0,1)$ と $Y \sim \mathrm{Chisq}(\nu)$ について,

$$\frac{Z}{\sqrt{Y/\nu}} \sim \mathrm{TDist}(\nu)$$

が成立することより,

$$t(x_*, \beta) = \frac{f(x_*)^T\hat{\beta} - f(x_*)^T\beta}{\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}}} = \frac{f(x_*)^T\hat{\beta} - f(x_*)^T\beta}{\hat{s}\sqrt{f(x_*)^T(X^TX)^{-1}f(x_*)}} \sim \mathrm{TDist}(n-r).$$

これを使うと t 分布を使って補正したP値と信頼区間を以下のように定義できる:

(真の)回帰函数の値に関する仮説 $f(x_*)^T\beta = f(x_*)^T\beta_0$ のP値の定義: $t(x_*, \beta_0)$ を

$$t(x_*, \beta_0) = \frac{f(x_*)^T\hat{\beta} - f(x_*)^T\beta_0}{\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}}} = \frac{f(x_*)^T\hat{\beta} - f(x_*)^T\beta_0}{\hat{s}\sqrt{f(x_*)^T(X^TX)^{-1}f(x_*)}}$$

と定め, P値を次のように定める:

$$\mathrm{pvalue}_{\mathrm{TDist}}(y|X, f(x_*)^T\beta = f(x_*)\beta_0) = 2(1 - \mathrm{cdf}(\mathrm{TDist}(n-r), |t(x_*, \beta_0)|)).$$

(真の)回帰函数の値 $f(x_*)^T\beta$ の信頼度 $1-\alpha$ の信頼区間の定義: $t_{\nu,\alpha/2}$ を

$$t_{\nu,\alpha/2} = \mathrm{quantile}(\mathrm{TDist}(\nu), 1 - \alpha/2).$$

と定め, 信頼区間を次のように定める:

$$\mathrm{confint}_{\mathrm{TDist}}^{f(x_*)^T\beta}(y|X) = \left[f(x_*)^T\hat{\beta} - t_{n-r,\alpha/2}\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}},\ f(x_*)^T\hat{\beta} + t_{n-r,\alpha/2}\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}}\right].$$

注意: 「真の」は「現実における真の」を意味しない. 単に仮想的なデータを生成するモデルのパラメータの設定を意味するに過ぎない.

2.16 予測区間

前節の仮定 $e \sim \mathrm{Normal}(0, \sigma)^n$ に加えて、さらに

$$\begin{bmatrix} e \\ e_* \end{bmatrix} \sim \mathrm{Normal}(0, \sigma)^{n+1}$$

と仮定し, 任意に $x_* \in \mathbb{R}$ を取り,

$$y_* = f(x_*)^T\beta + e_*$$

とおく. このとき,

$$y_* - f(x_*)^T\hat{\beta} = e_* + f(x_*)^T\beta - f(x_*)^T\hat{\beta} \sim \mathrm{Normal}\left(0, \mathrm{SE}_{y_* - f(x_*)^T\hat{\beta}}\right).$$

ここで,

$$\begin{aligned}\left(\mathrm{SE}_{y_* - f(x_*)^T\hat{\beta}}\right)^2 &= (e_* \text{ の分散}) + (f(x_*)^T\beta - f(x_*)^T\hat{\beta} \text{ の分散}) \\ &= \sigma^2 + \sigma^2 f(x_*)^T(X^TX)^{-1}f(x_*).\end{aligned}$$

すなわち,

$$\mathrm{SE}_{y_* - f(x_*)^T\hat{\beta}} = \sigma\sqrt{1 + f(x_*)^T(X^TX)^{-1}f(x_*)}.$$

前節の $\mathrm{SE}_{f(x_*)^T\hat{\beta}}$ との違いは平方根の中に e_* の分散から出て来た項の 1 が含まれていることである.

これより,

$$\widehat{\mathrm{SE}}_{y_* - f(x_*)^T\hat{\beta}} = \hat{s}\sqrt{1 + f(x_*)^T(X^TX)^{-1}f(x_*)}.$$

とおくと, 前節と同様にして,

$$\frac{y_* - f(x_*)^T\hat{\beta}}{\widehat{\mathrm{SE}}_{y_* - f(x_*)^T\hat{\beta}}} \sim \mathrm{TDist}(n-r).$$

これを使って, y_* の **予測区間** (prediction interval)を次のように定義できる:

$$\begin{aligned}&\mathrm{predint}^{y_*}_{\mathrm{TDist}}(y|X, x_*) \\ &= \left[f(x_*)^T\hat{\beta} - t_{n-r,\alpha/2}\,\widehat{\mathrm{SE}}_{y_* - f(x_*)^T\hat{\beta}},\ f(x_*)^T\hat{\beta} + t_{n-r,\alpha/2}\,\widehat{\mathrm{SE}}_{y_* - f(x_*)^T\hat{\beta}}\right].\end{aligned}$$

前節の $\widehat{\mathrm{SE}}_{f(x_*)^T\hat{\beta}}$ と $\widehat{\mathrm{SE}}_{y_* - f(x_*)^T\hat{\beta}}$ の違いは, 後者の定義式の平方根中に 1 が含まれていることである. だから, 予測区間は信頼区間よりも必ず広くなる. そうなる理由は y_* の定義を見れば明らかで, y_* の定義にはノイズの項 e_* が含まれている. その分だけ区間の幅が広くなる.

2.17 「回帰函数の値の信頼区間」と「予測区間」の違い

回帰函数の値の信頼区間は, 残差を含まない回帰函数の値(最も簡単な場合には回帰直線上の値)の信頼区間であり, 残差の分の確率的揺らぎを持つデータの数値の予測には使えない.

予測区間は, 残差による確率的揺らぎの大きさを含んでおり, データの数値の予測に使える.

実用的にはこの節より前の部分にある面倒な計算を理解していなくても, この違いだけを理解していれば足りることが多いように思われる.

しかし, 言葉での説明では分かり難いの次の節の計算例の視覚化を参照せよ.

3 線形回帰の計算例

以下において CI, PI はそれぞれ

- 真の回帰函数の値の信頼区間 (confidence interval)
- データの値(＝真の回帰函数＋ノイズ)の予測区間 (prediction interval)

の略である.

3.1 信頼区間と予測区間のプロット

In [10]:

```
function plot_linreg_confint_predint(x, y, F...; α = 0.05, kwargs...)
    n = length(x)
    r = length(F)
    X = [fⱼ(xᵢ) for xᵢ in x, fⱼ in F] # design matrix
    β̂ = X \ y # equivalent to β̂ = X*(X'X)\X'y
    ŷ = X * β̂ # orthogonal projection of y onto XR^r
    ŝ = norm(y - ŷ)/√(n - r)  # ŝ² is the unbiased estimator of σ²
    @show n r α
    m = @show quantile(TDist(n-r), 1-α/2)
    @show β̂ ŝ

    f̂(xstar) = sum(β̂ⱼ * fⱼ(xstar) for (β̂ⱼ, fⱼ) in zip(β̂, F))
    f(xstar) = [fⱼ(xstar) for fⱼ in F]
    invXX = inv(X'X)
    g(xstar) = ŝ * √(    f(xstar)' * invXX * f(xstar)) # for CI
    h(xstar) = ŝ * √(1 + f(xstar)' * invXX * f(xstar)) # for PI

    a, b = extrema(x)
    a, b = a - 0.05(b-a), b + 0.05(b-a)
    scatter(x, y; label="data", msc=:auto, c=1)
    plot!(xstar -> f̂(xstar), a, b; label="", c=:red)
    plot!(xstar -> f̂(xstar) - m*g(xstar), a, b;
        label="$(100(1-α))% CI", c=2, ls=:dash)
    plot!(xstar -> f̂(xstar) + m*g(xstar), a, b;
        label="", c=2, ls=:dash)
    plot!(xstar -> f̂(xstar) - m*h(xstar), a, b;
        label="$(100(1-α))% PI", c=3, ls=:dashdot)
    plot!(xstar -> f̂(xstar) + m*h(xstar), a, b;
        label="", c=3, ls=:dashdot)
    plot!(size=(520, 250), legend=:outertopright)
    plot!(; kwargs...)
end
```

Out[10]: plot_linreg_confint_predint (generic function with 1 method)

3.1.1 信頼区間と予測区間のテストプロット

In [11]:

```
Random.seed!(464937345105963)
x = 0:20
e = rand(Normal(0, 5), length(x))
y = @. 1 + x + e
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:1)...)
```

```
n = 21
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0930240544083096
β̂ = [2.5195937065195744, 0.8781138823583465]
ŝ = 3.9801432134453814
```

Out[11]:

橙色の破線で囲まれた信頼区間はデータと確率的に相性が良い回帰直線が含まれる範囲だと考えて良い.

緑色のdashdot線で囲まれた予測区間は既知のデータと相性が良い新たなもう１つのデータ点 $(x_{n+1} y_{n+1})$ が含まれる範囲だと考えてよい.

In [12]:

```
@rput x y
reg = R"""reg = lm(y ~ x)"""
R"""
xstars = data.frame(x = seq(-1, 21, 0.02))
conf.interval = predict(reg, newdata = xstars, interval = 'confidence', level = 0.95)
pred.interval = predict(reg, newdata = xstars, interval = 'prediction', level = 0.95)

plot(x, y, ylim = c(-17, 37))
lines(xstars$x, conf.interval[, 1], col = 'red')
lines(xstars$x, conf.interval[, 2], col = 'orange', lty=2)
lines(xstars$x, conf.interval[, 3], col = 'orange', lty=2)
lines(xstars$x, pred.interval[, 2], col = 'darkgreen', lty=4)
lines(xstars$x, pred.interval[, 3], col = 'darkgreen', lty=4)
"""
reg
```

Out[12]:

```
RObject{VecSxp}

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     2.5196       0.8781
```

以上によって, このノートのコードの結果とRによる計算結果のプロットは一致していることがわかった.

3.1.2 回帰直線の信頼区間と予測区間

In [13]:

```
Random.seed!(4649373)
n = 40
β₀, β₁ = β = [-1, 2]
@show β
f_true(x) = β₀ + β₁*x
x = rand(Normal(1, 1), n)
y = f_true.(x) + rand(Normal(0, 2), n)

scatter(x, y; label="data", legend=:topleft, msc=:auto)
plot!(f_true, extrema(x)...; label="y = f_true(x)", c=:blue)
plot!(size=(520, 250), legend=:outertopright)
plot!(ylim=(-8, 9))
```

```
β = [-1, 2]
```

Out[13]:

In [14]:

```
# 0次函数による回帰函数の推定
# 実質的に xᵢ 達を無視して, yᵢ 達だけを使った平均の推定になる.

@show β
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:0)...)
plot!(ylim=(-8, 9))
```

```
β = [-1, 2]
n = 40
r = 1
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0226909200367604
β̂ = [1.1019327498267735]
ŝ = 3.418537006157765
```

Out[14]:

In [15]:

```
# 回帰直線の推定

@show β
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:1)...)
plot!(ylim=(-8, 9))
```

```
β = [-1, 2]
n = 40
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0243941639119694
β̂ = [-1.0999950892202406, 2.4189615549368337]
ŝ = 2.202805671106067
```

Out[15]:

3.1.3 多項式回帰の信頼区間と予測区間 (オーバーフィッティングの例)

In [16]:

```
Random.seed!(4649373)

n = 30
f_true(x) = sin(x)
x = range(-π, π, n)
y = f_true.(x) + rand(Normal(0, 0.3), n)
scatter(x, y; label="data", legend=:topleft, msc=:auto)
plot!(f_true, extrema(x)...; label="y = f_true(x)", c=:blue)
plot!(size=(520, 250), legend=:outertopright)
plot!(ylim=(-2.5, 2.5))
```

Out[16]:

In [17]:

```
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:0)...)
plot!(ylim=(-2.5, 2.5))
```

```
n = 30
r = 1
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.045229642132704
β̂ = [-0.04660146725463895]
ŝ = 0.8308323950167983
```

Out[17]:

In [18]:

```
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:1)...)
plot!(ylim=(-2.5, 2.5))
```

```
n = 30
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0484071417952445
β̂ = [-0.04660146725463893, 0.2736800464953368]
ŝ = 0.6578091550765395
```

Out[18]:

In [19]:

```
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:3)...)
plot!(ylim=(-2.5, 2.5))
```

```
n = 30
r = 4
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0555294386428726
β̂ = [-0.07647096737236962, 0.9635052707437338, 0.008493481944703464, -0.1091362362032411]
ŝ = 0.317473403472657
```

Out[19]:

In [20]:

```
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:5)...)
plot!(ylim=(-2.5, 2.5))
```

```
n = 30
r = 6
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.063898561628025
β̂ = [0.01884302169822449, 1.189794202726402, -0.08232242246705707, -0.21005706997299373, 0.01
0079953778259737, 0.008666986713640645]
ŝ = 0.2886297752219675
```

Out[20]:

In [21]:

```
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:19)...)
plot!(ylim=(-2.5, 2.5))
```

```
n = 30
r = 20
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.228138851986274
β̂ = [0.22836295877493937, -0.5021873824551958, -2.6212933790765356, 9.849384435900465, 5.5486
93347612864, -17.736161230346553, -5.233019722764845, 14.423512848551033, 2.700964781883039,
-6.417649037368562, -0.8272203631473101, 1.6911805172023053, 0.15355216321584136, -0.27113858
848819544, -0.016872512413894734, 0.025970494212589505, 0.0010054479966396946, -0.00136589508
49738901, -2.4957119792605243e-5, 3.033475944591812e-5]
ŝ = 0.2666954706632545
```

Out[21]:

サンプルサイズ $n = 30$ のときに, $r - 1 = 19$ 次の多項式函数で推定された回帰函数はぐちゃぐちゃになってしまっており, 左右の端の方で信頼区間と予測区間の幅が爆発している.

これは **過剰適合** (オーバーフィッティング, overfitting)の典型例になっている.

パラメータが多いモデルで推定を行うと過剰適合が起こり易い.

この問題に対処するための方法に, パラメータの動き方に制限を付ける正則化やモデルの適切な複雑さを知るための赤池情報量規準 (AIC)やベイズ統計でのWAICなどがある. 以下はそれらの検索のためのリンクである.

- 正則化 回帰 (https://www.google.com/search?q=%E6%AD%A3%E5%89%87%E5%8C%96+%E5%9B%9E%E5%B8%B0%E3%83%A2%E3%83%87%E3%83%AB)
- 赤池情報量規準 (https://www.google.com/search?q=%E8%B5%A4%E6%B1%A0%E6%83%85%E5%A0%B1%E9%87%8F%E8%A6%8F%E6%BA%96)
- WAIC ベイズ (https://www.google.com/search?q=WAIC+%E3%83%99%E3%82%A4%E3%82%BA)

3.2 信頼区間と予測区間に対応するP値函数のプロット

In [22]:

```
function plot_linreg_pvalue_functions(x, y, F...;
        α = 0.05, xstars=nothing, ystars=nothing, kwargs...)
    n = length(x)
    r = length(F)
    X = [fⱼ(xᵢ) for xᵢ in x, fⱼ in F] # design matrix
    β̂ = X \ y # equivalent to β̂ = X*(X'X)\X'y
    ŷ = X * β̂ # orthogonal projection of y onto XRʳ
    ŝ = norm(y - ŷ)/√(n - r) # ŝ² is the unbiased estimator of σ²
    @show n r α
    m = @show quantile(TDist(n - r), 1-α/2)
    @show β̂ ŝ

    f̂(xstar) = sum(β̂ⱼ * fⱼ(xstar) for (β̂ⱼ, fⱼ) in zip(β̂, F))
    f(xstar) = [fⱼ(xstar) for fⱼ in F]
    invXX = inv(X'X)
    g(xstar) = ŝ * √(    f(xstar)' * invXX * f(xstar)) # for CI
    h(xstar) = ŝ * √(1 + f(xstar)' * invXX * f(xstar)) # for PI
    tdist = TDist(n - r)
    G(xstar, ystar) = 2ccdf(tdist, abs(ystar - f̂(xstar))/g(xstar)) # pval of CI
    H(xstar, ystar) = 2ccdf(tdist, abs(ystar - f̂(xstar))/h(xstar)) # pval of PI

    if isnothing(xstars)
        a, b = extrema(x)
        a, b = a - 0.05(b-a), b + 0.05(b-a)
        xstars = range(a, b, 400)
    end
    a, b = extrema(xstars)
    if isnothing(ystars)
        c, d = extrema(y)
        c, d = c - 0.1m*(d-c), d + 0.1m*(d-c)
        ystars = range(c, d, 400)
    end
    c, d = extrema(ystars)

    P = plot(; legend=:topleft, colorbar=false)
    heatmap!(xstars, ystars, sqrt∘G)
    scatter!(x, y; label="", c=:cyan, msc=1)
    plot!(xstar -> f̂(xstar) - m*g(xstar), a, b;
        label="$(100(1-α))% CI", c=:orange, ls=:dash)
    plot!(xstar -> f̂(xstar) + m*g(xstar), a, b;
        label="", c=:orange, ls=:dash)
    plot!(; xlim=(a, b), ylim=(c, d))
    title!("√P-value functon of CI")

    Q = plot(; legend=:topleft, colorbar=false)
    heatmap!(xstars, ystars, sqrt∘H)
    scatter!(x, y; label="", c=:cyan, msc=1)
    plot!(xstar -> f̂(xstar) - m*h(xstar), a, b;
        label="$(100(1-α))% PI", c=:lightgreen, ls=:dash)
    plot!(xstar -> f̂(xstar) + m*h(xstar), a, b;
        label="", c=:lightgreen, ls=:dash)
    plot!(; xlim=(a, b), ylim=(c, d))
    title!("√P-value function of PI")

    plot(P, Q; size=(800, 300), layout=(1, 2), kwargs...)
end
```

Out[22]: plot_linreg_pvalue_functions (generic function with 1 method)

3.2.1 信頼区間と予測区間に対応するP値函数のテストプロット

(回帰函数の値または回帰直線上の値の)信頼区間(confience interval)を CI と略し, 予測区間(prediction interval)を PI と略すことにする.

In [23]:

```
Random.seed!(464937345105963)
x = 0:20
e = rand(Normal(0, 5), length(x))
y = @. 1 + x + e
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:1)...;
    ystars=range(-17, 37, 400))
```

```
n = 21
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0930240544083096
β̂ = [2.5195937065195744, 0.8781138823583465]
ŝ = 3.9801432134453814
```

Out[23]:

3.2.2 回帰直線の信頼区間と予測区間に対応するP値函数

In [24]:

```
Random.seed!(4649373)
n = 40
β₀, β₁ = β = [-1, 2]
@show β
f_true(x) = β₀ + β₁*x
x = rand(Normal(1, 1), n)
y = f_true.(x) + rand(Normal(0, 2), n)

plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:0)...;
    ystars=range(-8, 9, 400))
```

```
β = [-1, 2]
n = 40
r = 1
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0226909200367604
β̂ = [1.1019327498267735]
ŝ = 3.418537006157765
```

Out[24]:

In [25]:

```
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:1)...;
    ystars=range(-8, 9, 400))
```

```
n = 40
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0243941639119694
β̂ = [-1.0999950892202406, 2.4189615549368337]
ŝ = 2.202805671106067
```

Out[25]:

3.2.3 回帰直線の信頼区間に対応するP値函数の動画

In [26]:

```
function plot_linreg_confint_pvalfunc_3d(x, y, F...;
        α = 0.05, camera=(60, 60), xstars=nothing, ystars=nothing, kwargs...)
    n = length(x)
    r = length(F)
    X = [fⱼ(xᵢ) for xᵢ in x, fⱼ in F] # design matrix
    β̂ = X \ y # equivalent to β̂ = X*(X'X)\X'y
    ŷ = X * β̂ # orthogonal projection of y onto XℝʳX
    ŝ = norm(y - ŷ)/√(n - r) # ŝ² is the unbiased estimator of σ²
    tdist = TDist(n - r)
    m = quantile(tdist, 1 - α/2)

    f̂(xstar) = sum(β̂ⱼ * fⱼ(xstar) for (β̂ⱼ, fⱼ) in zip(β̂, F))
    f(xstar) = [fⱼ(xstar) for fⱼ in F]
    invXX = inv(X'X)
    g(xstar) = ŝ * √(f(xstar)' * invXX * f(xstar))
    G(xstar, ystar) = 2ccdf(tdist, abs(ystar - f̂(xstar))/g(xstar))

    if isnothing(xstars)
        a, b = extrema(x)
        a, b = a - 0.05(b-a), b + 0.05(b-a)
        xstars = range(a, b, 400)
    end
    a, b = extrema(xstars)
    if isnothing(ystars)
        c, d = extrema(y)
        c, d = c - 0.1m*(d-c), d + 0.1m*(d-c)
        ystars = range(c, d, 400)
    end
    c, d = extrema(ystars)

    P = plot(; colorbar=false)
    surface!(xstars, ystars, G; camera, alpha=0.9)
    scatter3d!(x, y, zeros(length(x)); label="", c=:cyan, msc=:auto)
    plot!(size=(600, 500))
    plot!(; kwargs...)
end

Random.seed!(4649373)
n = 40
β₀, β₁ = β = [-1, 2]
@show β
f_true(x) = β₀ + β₁*x
x = rand(Normal(1, 1), n)
y = f_true.(x) + rand(Normal(0, 2), n)

plot_linreg_confint_pvalfunc_3d(x, y, (x -> x^k for k in 0:1)...)
```

```
β = [-1, 2]
```

Out[26]:

In [27]:

```
anim = @animate for t in 0:5:359
    plot_linreg_confint_pvalfunc_3d(x, y, (x->x^k for k in 0:1)...; camera=(t, 60))
end
gif(anim, "images/anim_linreg_confint_pvalfunc_3d.gif")
```

```
[ Info: Saved animation to D:\OneDrive\work\Statistics\2022\images\anim_linreg_confint_pvalfunc_3d.gif
```

Out[27]:

PDFファイルではこのアニメーションは動かない. 動いている様子を見たい人は以下のリンク先を参照せよ.

- 線形回帰の信頼区間に対応するP値函数 (https://github.com/genkuroki/Statistics/blob/master/2022/images/anim_linreg_confint_pvalfunc_3d.gif)

3.2.4 多項式回帰の信頼区間と予測区間に対応するP値函数

In [28]:

```
Random.seed!(4649373)

n = 30
f_true(x) = sin(x)
x = range(-π, π, n)
y = f_true.(x) + rand(Normal(0, 0.3), n)
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:0)...;
    ystars=range(-2.5, 2.5, 400))
```

```
n = 30
r = 1
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.045229642132704
β̂ = [-0.04660146725463895]
ŝ = 0.8308323950167983
```

Out[28]:

In [29]:

```
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:1)...;
    ystars=range(-2.5, 2.5, 400))
```

```
n = 30
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0484071417952445
β̂ = [-0.04660146725463893, 0.2736800464953368]
ŝ = 0.6578091550765395
```

Out[29]:

In [30]:

```
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:3)...;
    ystars=range(-2.5, 2.5, 400))
```

```
n = 30
r = 4
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0555294386428726
β̂ = [-0.07647096737236962, 0.9635052707437338, 0.008493481944703464, -0.1091362362032411]
ŝ = 0.317473403472657
```

Out[30]:

In [31]:

```
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:5)...;
    ystars=range(-2.5, 2.5, 400))
```

```
n = 30
r = 6
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.063898561628025
β̂ = [0.01884302169822449, 1.189794202726402, -0.08232242246705707, -0.21005706997299373, 0.01
0079953778259737, 0.008666986713640645]
ŝ = 0.2886297752219675
```

Out[31]:

In [32]:

```
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:19)...;
    ystars=range(-2.5, 2.5, 400))
```

```
n = 30
r = 20
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.228138851986274
β̂ = [0.22836295877493937, -0.5021873824551958, -2.6212933790765356, 9.849384435900465, 5.5486
93347612864, -17.736161230346553, -5.233019722764845, 14.423512848551033, 2.700964781883039,
-6.417649037368562, -0.8272203631473101, 1.6911805172023053, 0.15355216321584136, -0.27113858
848819544, -0.016872512413894734, 0.025970494212589505, 0.0010054479966396946, -0.00136589508
49738901, -2.4957119792605243e-5, 3.033475944591812e-5]
ŝ = 0.2666954706632545
```

Out[32]:

4 ロジスティック回帰

In [33]:

```julia
"""ロジスティック回帰のunlink函数"""
p_(xᵢ, β) = logistic(β[1] + β[2]*xᵢ)

"""ロジスティック回帰のunlink函数の導函数"""
function dp_(xᵢ, β)
    pᵢ = logistic(β[1] + β[2]*xᵢ)
    pᵢ * (1 - pᵢ)
end

"""ロジスティック回帰モデルの尤度函数の-1倍"""
function logistic_negloglik(y, x, β)
    -sum(logpdf(Bernoulli(p_(xᵢ, β)), yᵢ) for (xᵢ, yᵢ) in zip(x, y))
end

"""ロジスティック回帰のスコア統計量A"""
function score_A(y, x, β)
    sum(p_(xᵢ, β) - yᵢ for (xᵢ, yᵢ) in zip(x, y))
end

"""ロジスティック回帰のスコア統計量B"""
function score_B(y, x, β)
    sum(xᵢ * (p_(xᵢ, β) - yᵢ) for (xᵢ, yᵢ) in zip(x, y))
end

"""ロジスティック回帰のフィッシャー情報量行列の成分a"""
function fisherinfo_a(x, β)
    sum(dp_(xᵢ, β) for xᵢ in x)
end

"""ロジスティック回帰のフィッシャー情報量行列の成分b"""
function fisherinfo_b(x, β)
    sum(xᵢ * dp_(xᵢ, β) for xᵢ in x)
end

"""ロジスティック回帰のフィッシャー情報量行列の成分c"""
function fisherinfo_c(x, β)
    sum(xᵢ^2 * dp_(xᵢ, β) for xᵢ in x)
end

"""ロジスティック回帰のフィッシャー情報量行列のすべての成分"""
function fisherinfo(x, β)
    a = fisherinfo_a(x, β)
    b = fisherinfo_b(x, β)
    c = fisherinfo_c(x, β)
    a, b, c
end

"""ロジスティック回帰モデルの最尤法"""
function logistic_mle(y, x; alg=LBFGS())
    f(β) = logistic_negloglik(y, x, β)
    o = optimize(f, MVector(0.0, 0.0), alg)
    β̂ = o.minimizer
end

"""ロジスティック回帰モデルの乱数"""
function logistic_rand(x, β)
    rand.(Bernoulli.(p_.(x, Ref(β))))
end
```

Out[33]: logistic_rand

In [34]:

```julia
"""β̂とSEhatの計算"""
function betahat_sehat_star(y, x, xstar; alg=LBFGS())
    β̂ = logistic_mle(y, x; alg)
    â, b̂, ĉ = fisherinfo(x, β̂)
    SEhat = √max(0, safediv(ĉ - 2b̂*xstar + â*xstar^2, â*ĉ - b̂^2))
    β̂, SEhat
end

"""β̂とSEhatとẑの計算"""
function betahat_sehat_zhat_star(y, x, xstar, tstar; alg=LBFGS())
    β̂, SEhat = betahat_sehat_star(y, x, xstar; alg)
    ẑ = safediv(β̂[1] + β̂[2]*xstar - tstar, SEhat)
    β̂, SEhat, ẑ
end

"""ロジスティック回帰のMonte Carloシミュレーション(xstar版)"""
function sim_logistic_regression_star(;
        β=[-4, 2], xstar=1, n=200,
        x=range(-β[1]/β[2]-2, -β[1]/β[2]+2, n),
        L=10^4, alg=LBFGS()
    )
    betahat = Vector{MVector{2, Float64}}(undef, L)
    sehat = Vector{Float64}(undef, L)
    zhat = Vector{Float64}(undef, L)
    tstar = β[1] + β[2]*xstar
    @threads for i in 1:L
        y = logistic_rand(x, β)
        β̂, SEhat, ẑ = betahat_sehat_zhat_star(y, x, xstar, tstar; alg)
        betahat[i] = β̂
        sehat[i] = SEhat
        zhat[i] = ẑ
    end
    betahat, sehat, zhat
end
```

Out[34]: sim_logistic_regression_star

In [35]:

```julia
"""tstarに関するWald型P値函数"""
function pvalue_tstar_wald(y, x, xstar, tstar; alg=LBFGS())
    ẑ = betahat_sehat_zhat_star(y, x, xstar, tstar; alg)[3]
    2ccdf(Normal(), abs(ẑ))
end

"""pstarに関するWald型P値函数"""
function pvalue_pstar_wald(y, x, xstar, pstar; alg=LBFGS())
    tstar = logit(pstar)
    pvalue_tstar_wald(y, x, xstar, tstar; alg)
end

"""tstarに関する信頼区間"""
function confint_tstar_wald(y, x, xstar; α = 0.05, alg=LBFGS())
    z = quantile(Normal(), 1-α/2)
    β̂, SEhat = betahat_sehat_star(y, x, xstar; alg)
    m = β̂[1] + β̂[2]*xstar
    m - z*SEhat, m + z*SEhat
end

"""pstarに関する信頼区間"""
function confint_pstar_wald(y, x, xstar; α = 0.05, alg=LBFGS())
    tL, tU = confint_tstar_wald(y, x, xstar; α, alg)
    logistic(tL), logistic(tU)
end
```

Out[35]: confint_pstar_wald

In [36]:

```julia
function betahat_sehat_beta1(y, x; alg=LBFGS())
    β̂ = logistic_mle(y, x; alg)
    â, b̂, ĉ = fisherinfo(x, β̂)
    SEhat = √max(0, safediv(â, â*ĉ - b̂^2))
    β̂, SEhat
end

function betahat_sehat_zhat_beta1(y, x, β₁; alg=LBFGS())
    β̂, SEhat = betahat_sehat_beta1(y, x; alg)
    β̂₁ = β̂[2]
    zhat = safediv(β̂₁ - β₁, SEhat)
    β̂, SEhat, zhat
end

"""ロジスティック回帰のMonte Carloシミュレーション(β₁版)"""
function sim_logistic_regression_beta1(;
        β=[-4, 2], n=200,
        x=range(-β[1]/β[2]-2, -β[1]/β[2]+2, n),
        L=10^4, alg=LBFGS()
    )
    betahat = Vector{MVector{2, Float64}}(undef, L)
    sehat = Vector{Float64}(undef, L)
    zhat = Vector{Float64}(undef, L)
    @threads for i in 1:L
        y = logistic_rand(x, β)
        β̂, SEhat, ẑ = betahat_sehat_zhat_beta1(y, x, β[2]; alg)
        betahat[i] = β̂
        sehat[i] = SEhat
        zhat[i] = ẑ
    end
    betahat, sehat, zhat
end
```

Out[36]: sim_logistic_regression_beta1

In [37]:

```julia
"""β₁のWald型のP値"""
function pvalue_beta1_wald(y, x, β₁; alg=LBFGS())
    zhat = betahat_sehat_zhat_beta1(y, x, β₁; alg)[3]
    2ccdf(Normal(), abs(zhat))
end

"""β₁のWald型信頼区間"""
function confint_beta1_wald(y, x; α=0.05, alg=LBFGS())
    z = quantile(Normal(), 1-α/2)
    β̂, SEhat = betahat_sehat_beta1(y, x; alg)
    β̂₁ = β̂[2]
    [β̂₁ - z*SEhat, β̂₁ + z*SEhat]
end
```

Out[37]: confint_beta1_wald

4.1 ロジスティック函数とロジット函数

ロジスティック函数 を次のように定める:

$$\mathrm{logistic}(t) = \frac{1}{1+e^{-t}} \qquad (t \in \mathbb{R})$$

ロジスティック函数 $f(t) = \mathrm{logistic}(t)$ は微分方程式

$$f'(t) = f(t)(1-f(t)), \qquad f(0) = \frac{1}{2}$$

を満たしており(自分で確認せよ), この条件を満たす函数 $f(t)$ はロジスティック函数になる.

ロジスティック函数の微分が出て来る計算は上の微分方程式を使うと楽にできる.

ロジスティック函数は狭義単調増加函数で, $t \to -\infty$ のとき 0 に収束し, $t \to \infty$ で 1 に収束する.

ロジスティック函数 $p = \mathrm{logistic}(t)$ $(t \in \mathbb{R})$ の値域は $0 < p < 1$ になる.

ロジスティック函数 $p = \mathrm{logistic}(t)$ の逆函数は **ロジット函数** と呼ばれている. ロジット函数は対数オッズ函数であり, 次のように表される:

$$\mathrm{logit}(p) = \log\frac{p}{1-p} \qquad (0 < t < 1).$$

In [38]:

```
P1 = plot(logistic, -8, 8;
    label="", title = "p = logistic(t)",
    xguide="t", yguide="p", legend=:topleft,
    xtick=-10:2:10, ytick=0:0.1:1)
P2 = plot(logit, 0, 1;
    label="", title="t = logit(p)",
    ytick=-10:2:10, xtick=0:0.1:1,
    xguide="p", yguide="t", legend=:topleft,
    ylim=(-8.4, 8.4), bottommargin=4Plots.mm)
plot(P1, P2; size=(600, 300), tickfontsize=7)
```

Out[38]:

4.2 ロジスティック回帰のデータの形

データは以下の形で得られると仮定する:

$$(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R} \times \{1, 0\}.$$

x_i 達は実数で y_i 達は 1 または 0 であるとする.

このデータは各 x_i ごとに x_i の値に応じである確率で y_i の値がランダムに 1 または 0 の値になるのように解釈できると仮定する.

この形のデータを2つのベクトルで表す:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \{1, 0\}^n.$$

4.3 ロジスティック回帰でのリンク函数

ベクトル値の回帰係数パラメータ

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \in \mathbb{R}^2$$

を考え, 確率パラメータ達 $0 < p_i < 1$ が次のように表されていると仮定する:

$$p_i = \mathrm{logistic}(\beta_0 + \beta_1 x_i).$$

すなわち,

$$\mathrm{logit}(p_i) = \log \frac{p_i}{1 - p_i} = \beta_0 + \beta_1 x_i.$$

このときロジット函数を **リンク函数** (link function)と呼び, ロジスティック函数を **アンリンク函数** (unlink function)と呼ぶことがある.

注意: p_i 達のリンク函数として対数函数を使った場合には

$$\log p_i = \beta_0 + \beta_1 x_i$$

による回帰を考え, リンク函数として恒等函数を使った場合には

$$p_i = \beta_0 + \beta_1 x_i$$

による回帰を考えることになる. どれを使うのが適切であるかは目的によって変わる. **注意終**

4.4 ロジスティック回帰の統計モデル

前節までの記号の下で, 統計モデルとして $y \in \{1,0\}^n$ に関する以下の確率質量函数を採用する:

$$
\begin{aligned}
P(y|x,\beta) &= \prod_{i=1}^{n} \left(p_i^{y_i} (1-p_i)^{1-y_i} \right) \\
&= \prod_{i=1}^{n} \left(\mathrm{logistic}(\beta_0 + \beta_1 x_i)^{y_i} (1 - \mathrm{logistic}(\beta_0 + \beta_1 x_i))^{1-y_i} \right)
\end{aligned}
$$

このモデルの確率分布を $\mathrm{LogisticModel}(x,\beta)$ と書くと,

$$
\mathrm{LogisticModel}(x,\beta) = \prod_{i=1}^{n} \mathrm{Bernoulli}(\mathrm{logistic}(\beta_0 + \beta_1 x_i)).
$$

これは最も簡単な場合でもっと複雑なモデルを考えることもできる. (例えば $\beta_0 + \beta_1 x_i$ の部分をもっと複雑にできる).

$Y = [Y_i]_{i=1}^n$ をこの統計モデルに従う確率変数であるとする.

このとき, 各 Y_i の値は次のようにしてランダムに決まると考えることができる:

(1) x_i の値に対する確率パラメータ p_i の値が $p_i = \mathrm{logistic}(\beta_0 + \beta_1 x_i)$ で決まる.

(2) Y_i の値はランダムに決まり, 確率 p_i で 1 になり, 確率 $1 - p_i$ で 0 になる.

x_i 達から $Y_i = 1, 0$ 達の値を確率的に決定する確率パラメータ達を

$$
p_i = \mathrm{logistic}(\beta_0 + \beta_1 x_i) \qquad \left(\iff \mathrm{logit}(p_i) = \log \frac{p_i}{1-p_i} = \beta_0 + \beta_1 x_i \right)
$$

の形式で推定することが, **ロジスティック回帰** と呼ばれている.

In [50]:

```
# ロジスティック回帰モデルのサンプル(データ)の生成
#Random.seed!(4649373)
β₀, β₁ = β = [-4, 2]
n = 51
x = range(-β₀/β₁-2, -β₀/β₁+2, n)
#y = logistic_rand(x, β)
y = Bool[
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1]

# モデルとデータの視覚化
scatter(x, y; label="sample", msc=:auto, alpha=0.7)
plot!(xstar → logistic(β[1] + β[2]*xstar);
    label="p = logistic($(β₀)+$(β₁)x)", c=:blue, ls=:dot)
plot!(ytick=0:0.1:1, yguide="y, p", xguide="x")
title!("size-$n sample of logistic model with β = $β")
plot!(legend=:outertopright, size=(800, 250),
    leftmargin=4Plots.mm, bottommargin=4Plots.mm)
```

Out[50]:

4.5 最尤法

データの数値 $x \in \mathbb{R}^n$, $y \in \{1,0\}^n$ に対して, 前節で定義した統計モデルの尤度函数の対数の -1 倍 $L = L(\beta)$ は次の形になる:

$$
L = L(\beta) = -\log p(y|x,\beta) = -\sum_{i=1}^{n} \left(y_i \log p_i + (1-y_i) \log(1-p_i) \right).
$$

ここで,

$$p_i = \mathrm{logistic}(\beta_0 + \beta_1 x_i) \quad (i = 1, \dots, n).$$

対数尤度函数の -1 倍 $L(\beta)$ を最小化する β を $\hat{\beta} = [\hat{\beta}_0, \hat{\beta}_1]^T$ と書き, **最尤法の解** (さいゆうほうの解)と呼ぶことにする.

特別な場合を除いて, 最尤法の計算はコンピュータによる最適化の数値計算によって行うことになる. (その手法も色々ある.)

In [51]:

```
# ロジスティック回帰の実行例
# 上の方で定義されている logistic_mle(y, x) 函数の定義の中で,
# Optim.jl パッケージの optimize 函数が使われている.
β̂₀, β̂₁ = β̂ = logistic_mle(y, x)
@show β₀, β₁
@show β̂₀, β̂₁

# ロジスティック回帰の実行結果の視覚化
scatter(x, y; label="sample", msc=:auto, alpha=0.7)
plot!(xstar → logistic(β[1] + β[2]*xstar);
    label="p = logistic($(β₀)+$(β₁)x)", c=:blue, ls=:dot)
plot!(xstar → logistic(β̂₀ + β̂₁*xstar); label="MLE", c=2)
plot!(ytick=0:0.1:1, yguide="y, p", xguide="x")
title!("size-$n sample of logistic model with β = $β")
plot!(legend=:outertopright, size=(800, 250), leftmargin=2Plots.mm)
```

```
(β₀, β₁) = (-4, 2)
(β̂₀, β̂₁) = (-3.377183684137118, 1.5218694633641356)
```

Out[51]:

4.6 スコア統計量とFisher情報量行列

この節では $y = [y_i]_{i=1}^n$ はロジスティック回帰の統計モデルに従う確率変数であるとする. (「ロジスティック回帰の統計モデル」の節ではこれを Y と書いていたが, 前節の記号をそのまま使いたいので, $y = [y_i]_{i=1}^n$ と書くことにする. 一種の手抜き.)

以下では, ロジスティック函数が満たす微分方程式から得られる次の公式を自由に用いる:

$$\frac{\partial p_i}{\partial \beta_0} = p_i(1 - p_i), \qquad \frac{\partial p_i}{\partial \beta_1} = x_i p_i(1 - p_i).$$

スコア統計量 $A = A(\beta)$, $B = B(\beta)$ を次のように定める:

$$A = A(\beta) = \frac{\partial L}{\partial \beta_0} = -\sum_{i=1}^n (y_i(1 - p_i) - (1 - y_i)p_i) = \sum_{i=1}^n (p_i - y_i),$$
$$B = B(\beta) = \frac{\partial L}{\partial \beta_1} = -\sum_{i=1}^n (x_i y_i(1 - p_i) - x_i(1 - y_i)p_i) = \sum_{i=1}^n x_i(p_i - y_i).$$

β は $p_i = \mathrm{logistic}(\beta_0 + \beta_1 x_i)$ の中に含まれている.

y_i は確率 p_i で 1 になり, 確率 $1 - p_i$ で 0 になるので, $E[A] = E[B] = 0$ となる.

最尤法の解 $\hat{\beta}$ は $L = L(\beta)$ を最小化するので,

$$A(\hat{\beta}) = B(\hat{\beta}) = 0$$

を満たしている. ベクトル値確率変数 $U = U(\beta)$ を次のように定める:

$$U = U(\beta) = \begin{bmatrix} A(\beta) \\ B(\beta) \end{bmatrix}$$

この U をも **スコア統計量** と呼ぶことにする.

Fisher情報量行列 の成分 $a = a(\beta)$, $b = b(\beta)$, $c = c(\beta)$ を以下のように定める:

$$
\begin{aligned}
a = a(\beta) &= \frac{\partial^2 L}{\partial \beta_0^2} = \sum_{i=1}^n p_i(1-p_i), \\
b = b(\beta) &= \frac{\partial^2 L}{\partial \beta_0 \partial \beta_1} = \sum_{i=1}^n x_i p_i(1-p_i), \\
c = c(\beta) &= \frac{\partial^2 L}{\partial \beta_1^2} = \sum_{i=1}^n x_i^2 p_i(1-p_i).
\end{aligned}
$$

$E[(y_i - p_i)^2] = E[y_i^2] - E[y_i]^2 = E[y_i] - E[y_i]^2 = p_i - p_i^2 = p_i(1-p_i)$ を使った直接的な計算によって, a, c はそれぞれスコア統計量 A, B の分散に等しく, b はスコア統計量 A, B の共分散に等しいことを示せる(実はこのようなことは一般的に成立している):

$$
a = E[A^2], \quad b = E[AB], \quad c = E[B^2].
$$

Fisher情報量行列 $I(\beta)$ を次のように定める:

$$
I(\beta) = \begin{bmatrix} a(\beta) & b(\beta) \\ b(\beta) & c(\beta) \end{bmatrix}.
$$

Fisher情報量行列 $I(\beta)$ はスコア統計量 $U(\beta) = [A(\beta), B(\beta)]^T$ の分散共分散行列になっている.

注意: Fisher情報量行列を $I(\beta)$ と単位行列を混同しないこと!

注意: 「Fisher情報量行列」のような何か非常に高級で難しそうな名前が付いているという理由で「これは難しいものだ」と判断してはいけない. 上の場合には, その成分は対数尤度函数の -1 倍 $L = L(\beta)$ を2回偏微分しただけのものに過ぎず, $L = L(\beta)$ を1回偏微分して得られるスコア統計量の分散共分散行列を与える量になっている行列であるに過ぎない. 付けられた名前の権威に負けて恐れを持ってはいけない. 恐れを持たずに数学的な内容だけを見ればよい.

注意: Fisher情報量行列の一般的な場合の定義では, 成分を定義するときには2回偏微分した後に期待値を取る. 上でそうなっていない理由は期待値を取る前の2階の偏微分の段階ですでに定数になってしまっているからである.

4.7 問題: 一般の場合のスコア統計量とFisher情報量行列

一般のパラメータ $\theta = (\theta_1, \ldots, \theta_d)$ を持つ統計モデル $p(y|\theta)$ とそれに従う確率変数 y が与えられていて, y の対数尤度函数の -1 倍を $L(\theta) = -\log p(y|\theta)$ と書くとき(これは確率変数 y に依存するので確率変数になる), **スコア統計量** $U = [U_i]_{i=1}^n$ が

$$
U_i = \frac{\partial}{\partial \theta_i} L(\theta)
$$

と定義され, **Fisher情報量行列** $I(\theta)$ の (i, j) 成分 $I_{ij}(\theta)$ が

$$
I_{ij}(\theta) = E\left[\frac{\partial^2}{\partial \theta_i \partial \theta_j} L(\theta)\right]
$$

と定義される. ここで $E[\ ]$ は確率変数 y に関する期待値を取る操作である. 次が成立することを示せ:

$$
E[U_i] = 0, \quad E[U_i U_j] = I_{ij}(\theta).
$$

解答例: $E[U_i] = 0$ を示そう. $p(y|\theta)$ の θ_i に関する偏導函数を $p_{\theta_i}(y|\theta)$ と書くと,

$$
\begin{aligned}
E[U_i] &= E\left[\frac{\partial}{\partial \theta_i} L(\theta)\right] = -E\left[\frac{\partial}{\partial \theta_i} \log p(y|\theta)\right] = -E\left[\frac{p_{\theta_i}(y|\theta)}{p(y|\theta)}\right] \\
&= -\int \frac{p_{\theta_i}(y|\theta)}{p(y|\theta)} p(y|\theta)\, dy = -\int p_{\theta_i}(y|\theta)\, dy \\
&= -\frac{\partial}{\partial \theta_i} \int p(y|\theta)\, dy = -\frac{\partial}{\partial \theta_i} 1 = 0.
\end{aligned}
$$

$I_{ij}(\theta) = E[U_i U_j]$ を示そう. $p(y|\theta)$ の2階の偏導函数を $p_{\theta_i \theta_j}(y|\theta)$ と書くと, 上と同様にして,

$$
E\left[\frac{p_{\theta_i \theta_j}(y|\theta)}{p(y|\theta)}\right] = \int p_{\theta_i \theta_j}(y|\theta)\, dy = \frac{\partial^2}{\partial \theta_i \partial \theta_j} \int p(y|\theta)\, dy = 0.
$$

これを使うと,

$$
\begin{aligned}
I_{ij}(\theta) &= E\left[\frac{\partial^2}{\partial \theta_i \partial \theta_j} L(\theta)\right] = -E\left[\frac{\partial^2}{\partial \theta_i \partial \theta_j} \log p(y|\theta)\right] \\
&= -E\left[\frac{p_{\theta_i \theta_j}(y|\theta)}{p(y|\theta)} - \frac{p_{\theta_i}(y|\theta) p_{\theta_j}(y|\theta)}{p(y|\theta)^2}\right] \\
&= E\left[\frac{p_{\theta_i}(y|\theta)}{p(y|\theta)^2} \frac{p_{\theta_j}(y|\theta)}{p(y|\theta)^2}\right] = E[U_i U_j].
\end{aligned}
$$

解答終

注意: 前節の最後の注意で「Fisher情報量行列という名前に権威を感じて恐れを持つな」と述べたが, ここまで来れた人は数理統計学の面白い部分まで十分に到達できていると考えられるので, 自分自身を誇って良いと思われる.

4.8 βの最尤推定量の分布の正規分布近似

スコア統計量 $U(\beta) = [A(\beta), B(\beta)]^T$ は次のように多変量正規分布に近似的に従っていると仮定する:

$$U(\beta) = \begin{bmatrix} A(\beta) \\ B(\beta) \end{bmatrix} \sim \mathrm{MvNormal}(0, I(\beta)), \text{ approximately.}$$

最尤法の解で得られる β の推定量 $\hat{\beta}$ を使って, 統計量 $\hat{a}, \hat{b}, \hat{c}$ を次のように定める:

$$\hat{a} = a(\hat{\beta}), \quad \hat{b} = b(\hat{\beta}), \quad \hat{c} = c(\hat{\beta}).$$

このとき, $A(\hat{\beta}) = B(\hat{\beta}) = 0$ でかつ

$$a = \frac{\partial A}{\partial \beta_0}, \quad b = \frac{\partial A}{\partial \beta_1} = \frac{\partial B}{\partial \beta_0}, \quad c = \frac{\partial B}{\partial \beta_1}$$

なので, 次の一次近似が成立している(1次までのTaylor展開):

$$\begin{aligned} A(\beta) &\approx A(\hat{\beta}) + \frac{\partial A}{\partial \beta_0}(\hat{\beta})(\beta_0 - \hat{\beta}_0) + \frac{\partial A}{\partial \beta_1}(\hat{\beta})(\beta_1 - \hat{\beta}_1) = \hat{a}(\beta_0 - \hat{\beta}_0) + \hat{b}(\beta_1 - \hat{\beta}_1), \\ B(\beta) &\approx B(\hat{\beta}) + \frac{\partial B}{\partial \beta_0}(\hat{\beta})(\beta_0 - \hat{\beta}_0) + \frac{\partial B}{\partial \beta_1}(\hat{\beta})(\beta_1 - \hat{\beta}_1) = \hat{b}(\beta_0 - \hat{\beta}_0) + \hat{c}(\beta_1 - \hat{\beta}_1). \end{aligned}$$

すなわち,

$$\begin{bmatrix} A \\ B \end{bmatrix} \approx \begin{bmatrix} \hat{a} & \hat{b} \\ \hat{b} & \hat{c} \end{bmatrix} \begin{bmatrix} \beta_0 - \hat{\beta}_0 \\ \beta_1 - \hat{\beta}_1 \end{bmatrix}.$$

ゆえに,

$$\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix} \approx \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} - \begin{bmatrix} \hat{a} & \hat{b} \\ \hat{b} & \hat{c} \end{bmatrix}^{-1} \begin{bmatrix} A \\ B \end{bmatrix}.$$

これより,

$$\hat{\beta} \sim \mathrm{MvNormal}(\beta,\ I(\hat{\beta})^{-1} I(\beta) I(\hat{\beta})^{-1}), \text{ approximately.}$$

さらに次の近似も成立していると仮定する:

$$I(\hat{\beta}) \approx I(\beta).$$

この仮定のもとで

$$\hat{\beta} \sim \mathrm{MvNormal}(\beta,\ I(\hat{\beta})^{-1}), \text{ approximately.}$$

In [41]:

```
# ẑの分布が標準正規分布で近似されることの確認
# β̂の分布の視覚化
PP, QQ = [], []
for n in (100, 400)
    @time betahat, sehat, zhat = sim_logistic_regression_star(; n)
    P = stephist(zhat; norm=true, label="zhat")
    plot!(Normal(), -5, 5; label="Normal(0,1)")
    title!("n = $n")
    push!(PP, P)
    Q = scatter(first.(betahat), last.(betahat);
        label="betahat", msc=:auto, alpha=0.5, ms=1)
    title!("n = $n")
    plot!(xlim=(-8, -1), ylim=(0.5, 4.5), xtick=-10:10)
    push!(QQ, Q)
end
plot(PP...; size=(800, 250), layout=(1, 2)) ▷ display
plot(QQ...; size=(800, 250), layout=(1, 2)) ▷ display
```

```
1.427043 seconds (3.68 M allocations: 149.269 MiB, 4.45% gc time, 453.61% compilation time)
3.874268 seconds (3.42 M allocations: 125.876 MiB, 1.86% gc time)
```

4.9 ロジスティック回帰における $\beta_0+\beta_1 x$ に関するWald型のP値函数と信頼区間

一般にベクトル値確率変数 $V = [V_i]_{i=1}^n \sim \mathrm{MvNormal}(\mu, \Sigma)$ と $c \in \mathbb{R}^n$ について,

$$
\begin{aligned}
E[V^T c] &= \mu^T c, \\
\mathrm{var}(V^T c) &= E[(V^T c - \mu^T c)^2] = E[((V-\mu)^T c)^2] = E[((V-\mu)^T c)^T ((V-\mu)^T c)] \\
&= E[(c^T (V-\mu))(V-\mu)^T c] = c^T E[(V-\mu))(V-\mu)^T] c = c^T \Sigma c.
\end{aligned}
$$

$x_* \in \mathbb{R}$ を任意に取って, この結果を $V = \hat{\beta}$, $c = [1, x_*]^T$ と

$$
\Sigma = I(\beta)^{-1} = \frac{1}{ac - b^2} \begin{bmatrix} c & -b \\ -b & a \end{bmatrix}
$$

に適用すると,

$$
V^T c = \hat{\beta}_0 + \hat{\beta}_1 x_*, \quad c^T \Sigma c = \frac{c - 2bx_* + ax_*^2}{ac - b^2}
$$

なので,

$$
\widehat{\mathrm{SE}}_{\hat{\beta}_0 + \hat{\beta}_1 x_*} = \sqrt{\frac{\hat{c} - 2\hat{b}x_* + \hat{a}x_*^2}{\hat{a}\hat{c} - \hat{b}^2}}
$$

とおくと,

$$\hat{\beta}_0 + \hat{\beta}_1 x_* \sim \mathrm{Normal}\left(\beta_0 + \beta_1 x_*,\ \widehat{\mathrm{SE}}_{\hat{\beta}_0+\hat{\beta}_1 x_*}\right),\ \text{approximately.}$$

以上の結果を使うと,「x_* に対応する y_* の値が 1 になる確率が $p_* = \mathrm{logistic}(\beta_0 + \beta_1 x_*)$ である」という仮説のWald型のP値を次のように定めることができる:

$$\mathrm{pvalue}_{\mathrm{Wald}}(y|x, x_*, \beta) = 2(1 - \mathrm{cdf}(\mathrm{Normal}(0, 1), \hat{z}(x_*, \beta)).$$

ここで,

$$\hat{z}(x_*, \beta) = \frac{(\hat{\beta}_0 + \hat{\beta}_1 x_*) - (\beta_0 + \beta_1 x_*)}{\widehat{\mathrm{SE}}_{\hat{\beta}_0+\hat{\beta}_1 x_*}}.$$

有意水準 $0 < \alpha < 1$ を任意に取り, $z_{\alpha/2}$ を次のように定める:

$$z_{\alpha/2} = \mathrm{quantile}(\mathrm{Normal}(0, 1), 1 - \alpha/2).$$

Wald型P値函数には $t_* = \beta_0 + \beta_1 x_*$ に関する次の信頼区間が対応している:

$$\mathrm{confint}_{\mathrm{Wald}}^{\beta_0+\beta_1 x_*}(y|x, x_*, \alpha) = \left[\hat{\beta}_0 + \hat{\beta}_1 x_* - z_{\alpha/2}\widehat{\mathrm{SE}}_{\hat{\beta}_0+\hat{\beta}_1 x_*},\ \hat{\beta}_0 + \hat{\beta}_1 x_* + z_{\alpha/2}\widehat{\mathrm{SE}}_{\hat{\beta}_0+\hat{\beta}_1 x_*}\right].$$

これを $p_* = \mathrm{logistic}(\beta_0 + \beta_1 x_*)$ に関する信頼区間に書き直すと,

$$\begin{aligned}&\mathrm{confint}_{\mathrm{Wald}}^{p_*}(y|x, x_*, \alpha)\\ &= \left[\mathrm{logistic}\left(\hat{\beta}_0 + \hat{\beta}_1 x_* - z_{\alpha/2}\widehat{\mathrm{SE}}_{\hat{\beta}_0+\hat{\beta}_1 x_*}\right),\ \mathrm{logistic}\left(\hat{\beta}_0 + \hat{\beta}_1 x_* + z_{\alpha/2}\widehat{\mathrm{SE}}_{\hat{\beta}_0+\hat{\beta}_1 x_*}\right)\right].\end{aligned}$$

In [42]:

```julia
# ロジスティック回帰モデルのサンプル(データ)の生成
#Random.seed!(4649373)
β₀, β₁ = β = [-4, 2]
n = 51
x = range(-β₀/β₁-2, -β₀/β₁+2, n)
#y = logistic_rand(x, β)
y = Bool[
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1]

# ロジスティック回帰
β̂₀, β̂₁ = β̂ = logistic_mle(y, x)
@show β₀, β₁
@show β̂₀, β̂₁
println()

# P値の例
for pstar in 0.2:0.1:0.6
    @eval @show pvalue_pstar_wald(y, x, 2, $pstar)
end

# 95%信頼区間の視覚化
α = 0.05
scatter(x, (yᵢ -> yᵢ==0 ? -0.05 : 1.05).(y); label="sample", msc=:auto, alpha=0.7)
plot!(xstar -> logistic(β₀ + β₁*xstar);
    label="true model", c=:blue, ls=:dot)
plot!(xstar -> logistic(β̂₀ + β̂₁*xstar); label="MLE", c=2)
plot!(xstar -> confint_pstar_wald(y, x, xstar; α)[1];
    fillrange = xstar -> confint_pstar_wald(y, x, xstar; α)[2],
    label="$(100(1-α))% CI", c=:red, ls=:dash, fillalpha=0.2)
plot!(xstar -> confint_pstar_wald(y, x, xstar; α)[2];
    label="", c=:red, ls=:dash)
plot!(ytick=0:0.1:1, yguide="p", xguide="x")
title!("size-$n sample of logistic model with β = $β")
plot!(legend=:outertopright, size=(800, 250),
    leftmargin=4Plots.mm, bottommargin=4Plots.mm)
```

```
(β₀, β₁) = (-4, 2)
(β̂₀, β̂₁) = (-3.377183684137118, 1.5218694633641356)

pvalue_pstar_wald(y, x, 2, 0.2) = 0.004572848441716502
pvalue_pstar_wald(y, x, 2, 0.3) = 0.16636478531964694
pvalue_pstar_wald(y, x, 2, 0.4) = 0.8461954256075008
pvalue_pstar_wald(y, x, 2, 0.5) = 0.3691434653598818
pvalue_pstar_wald(y, x, 2, 0.6) = 0.04657629503341515
```

Out[42]:

In [43]:

```
# 99%信頼区間の視覚化
α = 0.01
scatter(x, (yᵢ → yᵢ==0 ? -0.05 : 1.05).(y); label="sample", msc=:auto, alpha=0.7)
plot!(xstar → logistic(β₀ + β₁*xstar);
    label="true model", c=:blue, ls=:dot)
plot!(xstar → logistic(β̂₀ + β̂₁*xstar); label="MLE", c=2)
plot!(xstar → confint_pstar_wald(y, x, xstar; α)[1];
    fillrange = xstar → confint_pstar_wald(y, x, xstar; α)[2],
    label="$(100(1-α))% CI", c=:red, ls=:dash, fillalpha=0.2)
plot!(xstar → confint_pstar_wald(y, x, xstar; α)[2];
    label="", c=:red, ls=:dash)
plot!(ytick=0:0.1:1, yguide="p", xguide="x")
title!("size-$n sample of logistic model with β = $β")
plot!(legend=:outertopright, size=(800, 250),
    leftmargin=4Plots.mm, bottommargin=4Plots.mm)
```

Out[43]:

In [44]:

```
# P値函数の視覚化

xstars = range(0, 4, 400)
pstars = range(0, 1, 200)
heatmap(xstars, pstars, (xstar, pstar) → pvalue_pstar_wald(y, x, xstar, pstar);
    clim=(-0.1, 1), colorbar=false)
scatter!(x, (yᵢ → yᵢ==0 ? -0.05 : 1.05).(y); label="", msc=:auto, alpha=0.7, c=1)
plot!(ytick=0:0.1:1, yguide="p", xguide="x")
plot!(xlim=(-0.1, 4.1))
plot!(size=(630, 270), leftmargin=4Plots.mm, bottommargin=4Plots.mm)
```

Out[44]:

横軸の値 x_* ごとにパラメータ値 p_* にP値を対応させるP値函数が決まっている.

上のヒートマップで明るい所ほどP値が大きい.

P値が「暗過ぎない部分」が信頼区間になっている.

4.10 ロジスティック回帰における β_1 に関するWald型のP値函数と信頼区間

$\beta_0 + \beta_1 x_*$ に関するP値函数と信頼区間で $x_* = 0$ の場合を考えれば β_0 に関するP値函数と信頼区間が得られる. 以下では β_1 に関するP値函数と信頼区間を構成しよう.

この節でも次の近似が使えると仮定する:

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix} \sim \mathrm{MvNormal}(\beta,\ I(\hat{\beta})^{-1}),\ \text{approximately}.$$

このとき,

$$\widehat{\mathrm{SE}}_{\hat{\beta}_1} = \sqrt{\frac{\hat{a}}{\hat{a}\hat{c} - \hat{b}^2}} = \left(I(\hat{\beta})^{-1} \text{ の } (2,2) \text{ 成分の平方根}\right)$$

とおくと,

$$\hat{\beta}_1 \sim \mathrm{Normal}\left(\beta_1, \widehat{\mathrm{SE}}_{\hat{\beta}_1}\right), \text{ approximately.}$$

これにより,「一次の項の係数が β_1 である」という仮説のWald型のP値を次のように定めることができる:

$$\mathrm{pvalue}_{\mathrm{Wald}}(y|x, \beta_1) = 2(1 - \mathrm{cdf}(\mathrm{Normal}(0,1), |\hat{z}(\beta_1)|)).$$

ここで,

$$\hat{z}(\beta_1) = \frac{\hat{\beta}_1 - \beta_1}{\widehat{\mathrm{SE}}_{\hat{\beta}_1}}.$$

これに対応する β_1 に関する信頼区間は次のようになる:

$$\mathrm{confint}^{\beta_1}_{\mathrm{Wald}}(y|x, \alpha) = \left[\hat{\beta}_1 - z_{\alpha/2}\widehat{\mathrm{SE}}_{\hat{\beta}_1},\ \hat{\beta}_1 + z_{\alpha/2}\widehat{\mathrm{SE}}_{\hat{\beta}_1}\right].$$

ここで, $z_{\alpha/2} = \mathrm{quantile}(\mathrm{Normal}(0,1), 1 - \alpha/2)$.

In [45]:

```
# ẑ(β₁)が近似的に標準正規分布に従うことの確認
β₀, β₁ = β = [-4, 2]
PP = []
for n in (100, 400)
    @time betahat, sehat, zhat = sim_logistic_regression_beta1(; β, n)
    P = stephist(zhat; norm=true, label="zhat")
    plot!(Normal(), -5, 5; label="Normal(0,1)")
    title!("n = $n")
    push!(PP, P)
end
plot(PP...; size=(800, 250), layout=(1, 2))
```

```
0.814003 seconds (2.86 M allocations: 107.876 MiB, 4.06% gc time, 77.46% compilation time)
3.344584 seconds (3.41 M allocations: 125.591 MiB, 0.65% gc time)
```

Out[45]:

In [46]:

```
# ロジスティック回帰モデルのサンプル(データ)の生成
#Random.seed!(4649373)
β₀, β₁ = β = [-4, 2]
n = 51
x = range(-β₀/β₁-2, -β₀/β₁+2, n)
#y = logistic_rand(x, β)
y = Bool[
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1]
@show n

# ロジスティック回帰
β̂₀, β̂₁ = β̂ = logistic_mle(y, x)
@show β₀, β₁
@show β̂₀, β̂₁

# P値の例
@show pvalue_beta1_wald(y, x, β₁)

# 信頼区間
@show ci = confint_beta1_wald(y, x)

# P値函数と信頼区間の視覚化
xlim = confint_beta1_wald(y, x; α=0.001)
plot(β₁ -> pvalue_beta1_wald(y, x, β₁), xlim...; label="")
plot!(ytick=0:0.1:1, yguide="P-value", xguide="β₁")
plot!(ci, fill(0.05, 2); label="95% CI")
vline!([β̂₁]; label="MLE of β₁", ls=:dash) # maximum likelihood estimate of β₁
vline!([β₁]; label="true β₁", ls=:dot)
title!("n = $n")
```

```
n = 51
(β₀, β₁) = (-4, 2)
(β̂₀, β̂₁) = (-3.377183684137118, 1.5218694633641356)
pvalue_beta1_wald(y, x, β₁) = 0.23754555168228686
ci = confint_beta1_wald(y, x) = [0.7284702769069128, 2.3152686498213586]
```

Out[46]:

青の線は β_1 のP値函数であり, 橙色の線分は β_1 の 95% 信頼区間である.

信頼区間はP値函数を有意水準の高さで切断すると得られる.

緑の縦破線は β_1 の最尤推定値であり, 紫色の縦点線はテストデータを生成するために使ったパラメータの β_1 である.

注意: 「真のパラメータ値」(true parameter value)の意味は「現実における真の値」という意味ではなく, 「テストデータを生成するために使ったパラメータの値」という意味であることが多い点に注意を払わなければいけない.

In [47]:

```
# ロジスティック回帰モデルのサンプル(データ)の生成
#Random.seed!(4649373837)
β₀, β₁ = β = [-4, 2]
n = 51
x = range(-β₀/β₁-2, -β₀/β₁+2, n)
#y = logistic_rand(x, β)
y = Bool[
    0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
    1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1,
    1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1]
@show n

# ロジスティック回帰
β̂₀, β̂₁ = β̂ = logistic_mle(y, x)
@show β₀, β₁
@show β̂₀, β̂₁

# P値の例
@show pvalue_beta1_wald(y, x, β₁)

# 信頼区間
@show ci = confint_beta1_wald(y, x)

# P値函数と信頼区間の視覚化
xlim = confint_beta1_wald(y, x; α=0.001)
plot(β₁ -> pvalue_beta1_wald(y, x, β₁), xlim...; label="")
plot!(ytick=0:0.1:1, yguide="P-value", xguide="β₁")
plot!(ci, fill(0.05, 2); label="95% CI")
vline!([β̂₁]; label="MLE of β₁", ls=:dash) # maximum likelihood estimate of β₁
vline!([β₁]; label="true β₁", ls=:dot)
title!("n = $n")
```

```
n = 51
(β₀, β₁) = (-4, 2)
(β̂₀, β̂₁) = (-2.575270537221982, 1.2582835733046909)
pvalue_beta1_wald(y, x, β₁) = 0.03492292470061806
ci = confint_beta1_wald(y, x) = [0.5690637954319593, 1.9475033511774225]
```

Out[47]:

上の数値例では, データを生成するために使ったパラメータ値の $\beta_1 = 1$ が 95% 信頼区間に含まれていない.

4.11 ロジスティック回帰における β_1 に関するWald型のP値函数と信頼区間の動画

In [48]:

```
# ロジスティック回帰モデルのサンプル(データ)の生成
Random.seed!(46493735964)
β₀, β₁ = β = [-4, 2]
n = 1000
x = rand(Uniform(-β₀/β₁-2, -β₀/β₁+2), n)
y = logistic_rand(x, β)

# P値函数と信頼区間の動画
ran = [fill(50, 20); 50:100; 100:2:200; 200:4:400; 600:8:n; fill(n, 20)]
@time anim = @animate for m in ran
    @views X, Y = x[1:m], y[1:m]
    β̂₀, β̂₁ = β̂ = logistic_mle(Y, X)
    ci = confint_beta1_wald(Y, X)
    plot(β₁ → pvalue_beta1_wald(Y, X, β₁), 0, 5; label="")
    plot!(yguide="P-value", xguide="β₁")
    plot!(ytick=0:0.1:1, ylim=(-0.02, 1.02))
    plot!(ci, fill(0.05, 2); label="95% CI")
    vline!([β̂₁]; label="MLE of β₁", ls=:dash) # maximum likelihood estimate of β₁
    vline!([β₁]; label="true β₁", ls=:dot)
    title!("n = $m")
end

gif(anim, "images/anim_logisticreg_beta1_confint_pvalfunc.gif")
```

```
78.525101 seconds (20.83 M allocations: 708.353 MiB, 0.12% gc time, 0.46% compilation time)
```

```
[ Info: Saved animation to D:\OneDrive\work\Statistics\2022\images\anim_logisticreg_beta1_confint_pvalfunc.gif
```

Out[48]:

PDFファイルではこのアニメーションは動かない. 動いている様子を見たい人は以下のリンク先を参照せよ.

- [ロジスティック回帰のβ_1の信頼区間に対応するP値函数](https://github.com/genkuroki/Statistics/blob/master/2022/images/anim_logisticreg_beta1_confint_pvalfunc.gif) (https://github.com/genkuroki/Statistics/blob/master/2022/images/anim_logisticreg_beta1_confint_pvalfunc.gif)

5 x_i 達の値も1または0の場合のロジスティック回帰

x_i 達の値も1または0の場合のロジスティック回帰によって,

- [P値と仮説検定と信頼区間: 母比率の違い](https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/11%20Hypothesis%20testing%20and%20confidence%20inter%20Two%20proportions.ipynb) (https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/11%20Hypothesis%20testing%20and%20confidence%20inter%20Two%20proportions.ipynb)

におけるオッズ比パラメータに関する検定や信頼区間に関する結果を再現できることを説明しよう.

すなわち, ロジスティック回帰は「比率の比較」を含んでいる.

このことは, 線形回帰が「平均の推定」を含んでいることに似ている.

以下の説明は **スコア統計量** や **Fisher情報量行列** などを使う統計分析の方法への入門的解説にもなっている.

5.1 x_i 達の値も1または0の場合にロジスティック回帰モデルは2つの二項分布モデルに等しい

$x = [x_i]_{i=1}^n \in \{1, 0\}^n$ であると仮定する.

$X, Y \in \{1, 0\}$ について, n_{XY} を $(x_i, y_i) = (X, Y)$ となる i の個数と定め, n_X を $x_i = X$ となる i の個数と定める. これによって次の 2×2 の分割表が得られる:

	$Y = 1$	$Y = 0$	
$X = 1$	n_{11}	n_{10}	n_1
$X = 0$	n_{01}	n_{00}	n_0

$$\left(\sum_{X,Y} n_{X,Y} = \sum_X n_X = n \right).$$

行列 N を次のように定める:

$$N = \begin{bmatrix} n_{11} & n_{10} \\ n_{01} & n_{00} \end{bmatrix}$$

さらに, $p = p(\beta)$ と $q = q(\beta)$ を次のように定める:

$$p = p(\beta) = \mathrm{logistic}(\beta_0 + \beta_1), \quad q = q(\beta) = \mathrm{logistic}(\beta_0).$$

これは

$$\log \frac{p}{1-p} = \beta_0 + \beta_1, \quad \log \frac{q}{1-q} = \beta_0$$

と同値なので,

$$\beta_1 = \log \frac{p(1-q)}{(1-p)q} = (\text{パラメータの対数オッズ比}).$$

ロジスティック回帰の統計モデルの確率質量函数は N に関する次の確率質量函数に書き直される:

$$P(N|n_1, n_0, p, q) = \binom{n_1}{n_{11}} p^{n_{11}} (1-p)^{n_{10}} \cdot \binom{n_0}{n_{01}} q^{n_{01}} (1-q)^{n_{00}}.$$

5.2 x_i 達の値も1または0の場合のスコア統計量とFisher情報量行列

このとき, スコア統計量は次のようになる:

$$\begin{aligned} A &= A(\beta) = -n_{11}(1-p) + n_{10}p - n_{01}(1-q) + n_{00}q = n_1 p - n_{11} + n_0 q - n_{01}, \\ B &= B(\beta) = -n_{11}(1-p) + n_{10}p = n_1 p - n_{11}. \end{aligned}$$

最尤法の解 $\hat{\beta}$ に対応する $\hat{p} = p(\hat{\beta})$, $\hat{q} = q(\hat{\beta})$ は $A = 0$, $B = 0$ の解になっているので,

$$\hat{p} = \frac{n_{11}}{n_1} = \frac{n_{11}}{n_{11} + n_{10}}, \quad \hat{q} = \frac{n_{01}}{n_0} = \frac{n_{01}}{n_{01} + n_{00}}.$$

Fisher情報量行列の成分達は次のようになる:

$$\begin{aligned} a &= a(\beta) = n_1 p(1-p) + n_0 q(1-q), \\ b &= b(\beta) = n_1 p(1-p), \\ c &= c(\beta) = n_1 p(1-p). \end{aligned}$$

ゆえに, $\hat{a} = a(\hat{\beta})$, $\hat{b} = b(\hat{\beta})$, $\hat{c} = c(\hat{\beta})$ は次のようになる:

$$\hat{a} = \frac{n_{11} n_{10}}{n_1} + \frac{n_{01} n_{00}}{n_0}, \quad \hat{b} = \hat{c} = \frac{n_{11} n_{10}}{n_1}.$$

5.3 x_i 達の値も1または0の場合のWald型のP値函数と信頼区間

したがって,

$$\widehat{\mathrm{SE}}_{\hat{\beta}_1} = \sqrt{\frac{\hat{a}}{\hat{a}\hat{c} - \hat{b}^2}} = \sqrt{\frac{1}{n_{11}} + \frac{1}{n_{10}} + \frac{1}{n_{01}} + \frac{1}{n_{00}}}.$$

ゆえに

$$\hat{z}(\beta_1) = \frac{\hat{\beta}_1 - \beta_1}{\widehat{\mathrm{SE}}_{\hat{\beta}_1}}.$$

の分母の具体系がわかったので, パラメータの対数オッズ比が β_1 であるという仮説のWald型P値函数

$$\mathrm{pvalue}_{\mathrm{Wald}}(N|\beta_1) = 2(1 - \mathrm{cdf}(\mathrm{Normal}(0, 1), |\hat{z}(\beta_1)|))$$

や対数オッズ比 β_1 に関するWald型信頼区間

$$\mathrm{confint}^{\beta_1}_{\mathrm{Wald}}(N|\alpha) = \left[\hat{\beta}_1 - z_{\alpha/2}\widehat{\mathrm{SE}}_{\hat{\beta}_1},\ \hat{\beta}_1 + z_{\alpha/2}\widehat{\mathrm{SE}}_{\hat{\beta}_1}\right]$$

の計算の仕方がよく分かったことになる.

以上の結果は

- 「検定と信頼区間: 比率の比較」のノート (https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/11%20Hypothesis%20testing%20and%20confidence%20inter%20Two%20proportions.ipynb)

における「Wald版のオッズ比に関するP値と信頼区間」の再現になっている.

5.4 x_i 達の値も1または0の場合のWilson型のP値函数と信頼区間

5.4.1 A=0で定まる条件付き確率分布の正規分布近似

この節ではスコア統計量の分布の正規分布近似に戻って考え直す:

$$U(\beta) = \begin{bmatrix} A(\beta) \\ B(\beta) \end{bmatrix} \sim \mathrm{MvNormal}(0, I(\beta)),\ \text{approximately}.$$

$I(\beta)$ の成分は $a(\beta), b(\beta), c(\beta)$ と表されているのであった.

回帰係数パラメータの片方の β_1 (対数オッズ比パラメータ)を任意に固定して, β_0 のみを動かして得られる最尤法(尤度最大化)の解を $\tilde{\beta}_0$ と書き,

$$\tilde{\beta} = \begin{bmatrix} \tilde{\beta}_0 \\ \beta_1 \end{bmatrix}$$

とおく. このとき, $\tilde{\beta}_0$ は, 与えられた β_1 に対する方程式

$$A = A(\beta) = A\left(\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}\right) = 0$$

の解になる. ゆえに, $U(\tilde{\beta})$ が従う分布は $U(\beta)$ が従う分布を $A(\beta) = 0$ の場合に制限して得られる条件付き確率分布になる. このことから, 次が成立することがわかる:

$$B(\tilde{\beta}) \sim \mathrm{Normal}\left(0, \sqrt{\frac{\tilde{a}\tilde{c} - \tilde{b}^2}{\tilde{a}}}\right),\ \text{approximately}.$$

ここで, $\tilde{a}, \tilde{b}, \tilde{c}$ は $\beta = \tilde{\beta}$ のときの a, b, c である:

$$\tilde{a} = a(\tilde{\beta}), \quad \tilde{b} = a(\tilde{\beta}), \quad \tilde{c} = a(\tilde{\beta}).$$

$\tilde{a}/(\tilde{a}\tilde{c} - \tilde{b}^2)$ が $I(\tilde{\beta})^{-1}$ の $(2,2)$ 成分に等しいことに注意せよ. (2変量正規分布の条件付き確率分布について自分で計算してみて, そうなる理由を確認せよ.)

この結果を有用にするためには $\tilde{\beta}_0$ を求め, $\tilde{a}, \tilde{b}, \tilde{c}, B(\tilde{\beta})$ の具体的な形を決定する必要がある. 以下ではそれを実行しよう.

5.4.2 与えられた対数オッズ比パラメータの値 β_1 に対する β_0 の推定量に関する公式

公式: $\tilde{\beta}_0$ と $\tilde{a}, \tilde{b}, \tilde{c}, B(\tilde{\beta})$ の具体的な形は次のようになる:

$$\begin{aligned}
\hat{\beta}_0 &= \log\frac{n_{01} + \tilde{\delta}}{n_{00} - \tilde{\delta}}, \\
\tilde{a} &= \frac{(n_{11} - \tilde{\delta})(n_{10} + \tilde{\delta})}{n_1} + \frac{(n_{01} + \tilde{\delta})(n_{00} - \tilde{\delta})}{n_0}, \\
\tilde{b} = \tilde{c} &= \frac{(n_{11} - \tilde{\delta})(n_{10} + \tilde{\delta})}{n_1}, \\
B(\tilde{\beta}) &= -\tilde{\delta}.
\end{aligned}$$

ただし, $\tilde{\delta}$ は

$$\omega = \exp(\beta_1), \quad A = 1 - \omega, \quad B = n_{11} + n_{00} + \omega(n_{10} + n_{01}), \quad C = n_{11}n_{00} - \omega n_{01}n_{10}$$

とおいて, 次のように定義される:

$$\tilde{\delta} = \frac{2C}{B + \sqrt{B^2 - 4AC}}.$$

証明: $A(\beta)$ の具体的な形は

$$\begin{aligned}
&A(\beta) = -n_{11}(1-p) + n_{10}p - n_{01}(1-q) + n_{00}q = n_1 p - n_{11} + n_0 q - n_{01}, \\
&\frac{q}{1-q} = \exp(\beta_0), \quad \frac{p(1-q)}{(1-p)q} = \exp(\beta_1)
\end{aligned}$$

だったので, 与えられた β_1 に対する β_0 に関する方程式 $A(\beta) = 0$ は次と同値である:

$$\delta := n_{11} - n_1 p = -(n_{01} - n_0 q).$$

このとき,

$$p = \frac{n_{11} - \delta}{n_1}, \quad 1 - p = \frac{n_{10} + \delta}{n_1}, \quad q = \frac{n_{01} + \delta}{n_0}, \quad 1 - q = \frac{n_{00} - \delta}{n_0}$$

なので, 条件 $((p(1-q))/((1-p)q)) = \exp(\beta_1)$ は次のように書き直される:

$$\frac{(n_{11} - \delta)(n_{00} - \delta)}{(n_{10} + \delta)(n_{00} + \delta)} = \exp(\beta_1).$$

これを δ に関する2次方程式に直してから解くと, その解は

$$\tilde{\delta} = \frac{2C}{B + \sqrt{B^2 - 4AC}}.$$

ここで,

$$\omega = \exp(\beta_1), \quad A = 1 - \omega, \quad B = n_{11} + n_{00} + \omega(n_{10} + n_{01}), \quad C = n_{11}n_{00} - \omega n_{01}n_{10}.$$

$\tilde{p}, \tilde{q}$ を次のように定める:

$$\tilde{p} = \frac{n_{11} - \tilde{\delta}}{n_1}, \quad \tilde{q} = \frac{n_{01} + \tilde{\delta}}{n_0}.$$

このとき, β_0 に関する方程式 $A(q) = 0$ は次のように解ける:

$$\hat{\beta}_0 = \log \frac{\tilde{q}}{1 - \tilde{q}} = \log \frac{n_{01} + \tilde{\delta}}{n_{00} - \tilde{\delta}}.$$

そして, このとき,

$$\begin{aligned}
&\tilde{a} = n_1 \tilde{p}(1 - \tilde{p}) + n_0 \tilde{q}(1 - \tilde{q}) = \frac{(n_{11} - \tilde{\delta})(n_{10} + \tilde{\delta})}{n_1} + \frac{(n_{01} + \tilde{\delta})(n_{00} - \tilde{\delta})}{n_0}, \\
&\tilde{b} = \tilde{c} = n_1 \tilde{p}(1 - \tilde{p}) = \frac{(n_{11} - \tilde{\delta})(n_{10} + \tilde{\delta})}{n_1}, \\
&B(\tilde{\beta}) = n_1 \tilde{p} - n_{11} = -\tilde{\delta}.
\end{aligned}$$

5.4.3 対数オッズ比パラメータ β_1 に関するWilson型のP値函数と信頼区間の構成

以上の結果と $n_{11} + n_{10} = n_1, n_{01} + n_{00} = n_0$ より,

$$\begin{aligned}
\frac{\tilde{a}}{\tilde{a}\tilde{c} - \tilde{b}^2} &= \frac{\dfrac{(n_{11} - \tilde{\delta})(n_{10} + \tilde{\delta})}{n_1} + \dfrac{(n_{01} + \tilde{\delta})(n_{00} - \tilde{\delta})}{n_0}}{\dfrac{(n_{11} - \tilde{\delta})(n_{10} + \tilde{\delta})}{n_1} \dfrac{(n_{01} + \tilde{\delta})(n_{00} - \tilde{\delta})}{n_0}} \\
&= \frac{n_1}{(n_{11} - \tilde{\delta})(n_{10} + \tilde{\delta})} + \frac{n_1}{(n_{01} + \tilde{\delta})(n_{00} - \tilde{\delta})} \\
&= \frac{1}{n_{11} - \tilde{\delta}} + \frac{1}{n_{10} + \tilde{\delta}} + \frac{1}{n_{01} + \tilde{\delta}} + \frac{1}{n_{00} - \tilde{\delta}}.
\end{aligned}$$

したがって,

$$B(\tilde{\beta}) \sim \mathrm{Normal}\left(0, \sqrt{\frac{\tilde{a}\tilde{c} - \tilde{b}^2}{\tilde{a}}}\right), \text{ approximately.}$$

という結果は次のように書き直される:

$$z(\beta_1) := \tilde{\delta}\ \widehat{\mathrm{SE}}_{\tilde{\delta}}^{-1} \sim \mathrm{Normal}(0,1),\ \text{approximately}.$$

ここで,

$$\widehat{\mathrm{SE}}_{\tilde{\delta}}^{-1} = \sqrt{\frac{\tilde{a}}{\tilde{a}\tilde{c} - \tilde{b}^2}} = \sqrt{\frac{1}{n_{11} - \tilde{\delta}} + \frac{1}{n_{10} + \tilde{\delta}} + \frac{1}{n_{01} + \tilde{\delta}} + \frac{1}{n_{00} - \tilde{\delta}}}.$$

さらにこれは, 次と同値である:

$$\chi^2(\beta_1) := z(\beta_1)^2 = \tilde{\delta}^2\ \widehat{\mathrm{SE}}_{\tilde{\delta}}^{-2} \sim \mathrm{Chisq}(1),\ \text{approximately}.$$

これより, 「対数オッズ比パラメータは β_1 である」という仮説のWilson型の(スコア検定の)P値函数が以下のように定義される:

$$\mathrm{pvalue}_{\mathrm{Wilson}}(N|\beta_1) = 2(1 - \mathrm{cdf}(\mathrm{Normal}(0,1), |z(\beta_1)|) = \mathrm{cdf}(\mathrm{Chisq}(1), \chi^2(\beta_1)).$$

さらに, Wilson型(スコア検定に付随する)信頼区間が次のように得られる:

$$\mathrm{confint}_{\mathrm{Wilson}}^{\beta_1}(N|\alpha) = \{\ \beta_1 \in \mathbb{R} \mid \mathrm{pvalue}_{\mathrm{Wilson}}(N|\beta_1) \geq \alpha\ \}.$$

以上の結果は

- 「検定と信頼区間: 比率の比較」のノート (https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/11%20Hypothesis%20testing%20and%20confidence%20inter%20Two%20proportions.ipynb)

における「Pearsonのχ²検定版のオッズ比に関するP値と信頼区間」の再現になっている. 信頼区間の計算法のより詳しい説明や計

5.5 x_i 達の値も1または0の場合にロジスティック回帰の一般化の役に立ち方

$x_i = (X_i, k) \in \{1, 0\} \times \{1, \dots, K\}$ が $X_i = 1, 0$ と $k = 1, 2, \dots, K$ の組である場合への以上の話の一般化は, K 個の統計分析の結果をまとめあげる **メタアナリシス** などでも役に立っている.

例えば, K 個の統計分析に関する共通オッズ比を推定するためのMantel-Haenszel統計量の話やMantel-Haenszel検定の話をインターネットで検索してみよ. 共通オッズ比のMantel-Haenszel検定はロジスティック回帰を使った検定と本質的に同値になっている.

- 共通オッズ比 Mantel-Haenszel統計量 (https://www.google.com/search?q=%E5%85%B1%E9%80%9A%E3%82%AA%E3%83%83%E3%82%BA%E6%AF%94+Mantel-Haenszel%E7%B5%B1%E8%A8%88%E9%87%8F)
- Mantel-Haenszel検定 (https://www.google.com/search?q=Mantel-Haenszel%E6%A4%9C%E5%AE%9A)

検索すれば, x_i 達が離散的な値を取る場合のロジスティック回帰は医療統計でよく使われていることがわかるだろう.

このように, ロジスティック回帰は機械学習の文脈で解説されることが多いが, 医療統計でも使われている.

将来, 機械学習の技術を利用しようと思っている人も, それとは毛色が違う医療統計についても学んでおけば, アイデアの幅が広がり, オリジナルな仕事をできるかもしれない.