回帰 (regression)

- 黒木玄

- 2022-07-13〜2022-07-13

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.

目次

In [1]:
```
ENV["LINES"], ENV["COLUMNS"] = 100, 100
using Base.Threads
using BenchmarkTools
using DataFrames
using Distributions
using LinearAlgebra
using Memoization
using Printf
using QuadGK
using RCall
@rimport stats as R
using Random
Random.seed!(4649373)
using Roots
using SpecialFunctions
using StaticArrays
using StatsBase
using StatsFuns
using StatsPlots
default(fmt = :png, size = (400, 250),
    titlefontsize = 10, plot_titlefontsize = 12)
using SymPy
```

In [2]:
```
# Override the Base.show definition of SymPy.jl:
# https://github.com/JuliaPy/SymPy.jl/blob/29c5bfd1d10ac53014fa7fef468bc8deccadc2fc/src/types.

@eval SymPy function Base.show(io::IO, ::MIME"text/latex", x::SymbolicObject)
    print(io, as_markdown("\\displaystyle " *
            sympy.latex(x, mode="plain", fold_short_frac=false)))
end
@eval SymPy function Base.show(io::IO, ::MIME"text/latex", x::AbstractArray{Sym})
    function toeqnarray(x::Vector{Sym})
        a = join(["\\displaystyle " *
                sympy.latex(x[i]) for i in 1:length(x)], "\\\\")
        """\\left[ \\begin{array}{r}$a\\end{array} \\right]"""
    end
    function toeqnarray(x::AbstractArray{Sym,2})
        sz = size(x)
        a = join([join("\\displaystyle " .* map(sympy.latex, x[i,:]), "&")
                for i in 1:sz[1]], "\\\\")
        "\\left[ \\begin{array}{" * repeat("r",sz[2]) * "}" * a * "\\end{array}\\right]"
    end
    print(io, as_markdown(toeqnarray(x)))
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.

1.1 回帰の超一般論

一般に, 回帰はデータの数値が $(x_1, y_1), \ldots, (x_n, y_n)$ の形式で各々の y_i が数値 x_i に依存して決まるという場合を, パラメータ $x = (x_1, \ldots, x_n), \theta = (\theta_1, \ldots, \theta_d)$ を持つ $y = (y_1, \ldots, y_n)$ に関する確率密度函数(または確率質量函数)

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

でモデル化することによって行われるパラメータ θ の推定のことである.

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

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

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.

平均の信頼区間の t 分布を使った信頼区間の計算で使われる統計モデルは, 正規分布の標本分布モデル

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

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 y にあたる変数がこの場合の y で, x にあたる変数はない.

x にあたる変数がモデルのパラメータとして登場することが, 回帰(regression)の特徴である.

2 線形回帰

線形回帰一般の良い解説が見当たらなかったので, 以下の節で解説することにする. 以下の節のスタイルでの線形回帰の理解は機械学習一般の理解のためにも有用である.

2.1 データ

データは以下の形で得られると仮定する:

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

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 モデルの構成要素

$r < n$ であると仮定する.

r 個の函数 $f_1, \ldots, f_r : \mathbb{R} \to \mathbb{R}$ 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 $\beta_1, \ldots, \beta_r \in \mathbb{R}$ が与えられているとし,

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

とおく. このとき,

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

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.

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,

$$
X\beta = \left[\sum_{j=1}^{r} \beta_j f_j(x_i) \right]_{i=1}^{n}.
$$

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. (数値的には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.

2.4 正規分布による統計モデルの記述

統計モデルとして, y 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, $\| \ \|$ は通常のEuclidノルム(成分の2乗の和の平方根)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とすると,

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

と書ける. ここで, ベクトル値確率変数 $\varepsilon = [\varepsilon_i]_{i=1}^n$ の成分達 $\varepsilon_1, \ldots, \varepsilon_n$ 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

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

が平均 0, 分散 σ^2 の正規分布に従ってランダムに決まっていると考える.

注意: 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 $(\beta, \sigma^2) \mapsto p(y|X, \beta, \sigma^2)$ 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 x から決まる.)

尤度函数の対数の -2 倍は次の形になる:

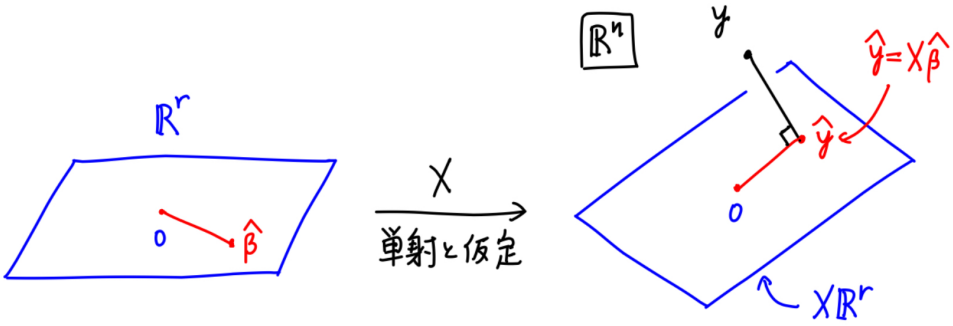
$$
L(\beta, \sigma^2) = -2 \log p(y|X, \beta, \sigma^2) = \frac{1}{\sigma^2} \|y - X\beta\|^2 + n \log \sigma^2 + n \log(2\pi).
$$

これを最小にする β, σ^2 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 $X\beta$ 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.



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

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

このとき,

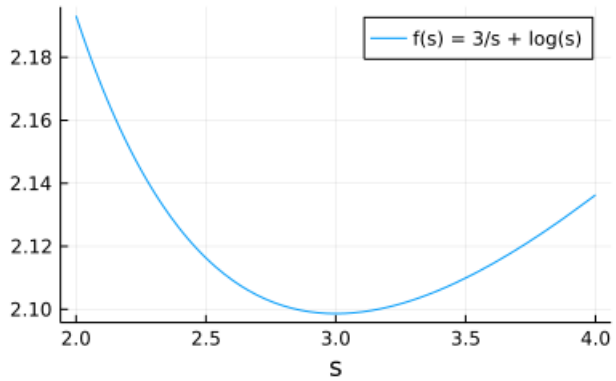
$$L(\hat{\beta}, \sigma^2) = \frac{n\hat{\sigma}^2}{\sigma^2} + n\log\sigma^2 + n\log(2\pi).$$

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, $f(s) = a/s + logs$ 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.)

In [4]:
```
1 plot(s -> 3/s + log(s), 2, 4; label="f(s) = 3/s + log(s)", xguide="s")
```

Out[4]:



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, $X^T X$ は可逆になるのであった. ゆえに

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

これと,

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

を合わせると, 尤度函数 $(\beta, \sigma^2) \mapsto p(y|X, \beta, \sigma^2)$ を最小化するパラメータ値が得られる.

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.

x, y から $\hat{\beta} = (X^T X)^{-1} X^T y$ を求めて, $\beta = \hat{\beta}$ とおくと, 二乗和 $\|y - X\beta\|$ が最小化されるので, 回帰係数 β をその $\hat{\beta}$ として求める方法を **最小二乗法** と呼ぶ.

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

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.

2.6 βとσ²の不偏推定量

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ゆる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 $E[ee^T] = \sigma^2 I$ の左辺は 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.)

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:

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

このとき,

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

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する.

ベクトル値確率変数 $\hat{\beta} = (X^T X)^{-1} X^T y$ の分散共分散行列は

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

より,

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

なので, 次のように計算される:

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

2つ目の等号で $E[ee^T] = \sigma^2 I$ を使った.

$X\hat{\beta}$ の分散共分散行列は,

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

より, 次のように計算される:

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

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

$X(X^T X)^{-1} X^T$ 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にも注意せよ.)

残差 $y - X\hat{\beta}$ の分散共分散行列は,

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

より, 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.

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

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

を使うと,

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

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.

2.7 例: 平均の推定の場合

線形回帰は平均の推定を含む.

$r = 1$ 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, 計画行列 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 の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:

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

この結果は不偏分散の話の線形回帰の理論を使った再現になっている.

2.8 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 達の不変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,

$$
\begin{aligned}
\beta_0 &= \bar{y} - \frac{\hat{\sigma}_{xy}}{\hat{\sigma}_x^2}\bar{x} = \bar{y} - \frac{s_{xy}}{s_x^2}\bar{x}, \quad \beta_1 = \frac{\hat{\sigma}_{xy}}{\hat{\sigma}_x^2} = \frac{s_{xy}}{s_x^2}, \\
\hat{\sigma}^2 &= \frac{n-1}{n}\frac{s_x^2s_y^2 - s_{xy}^2}{s_x^2}, \quad \hat{s}^2 = \frac{n-1}{n-2}\frac{s_x^2s_y^2 - s_{xy}^2}{s_x^2}.
\end{aligned}
$$

以上の公式は

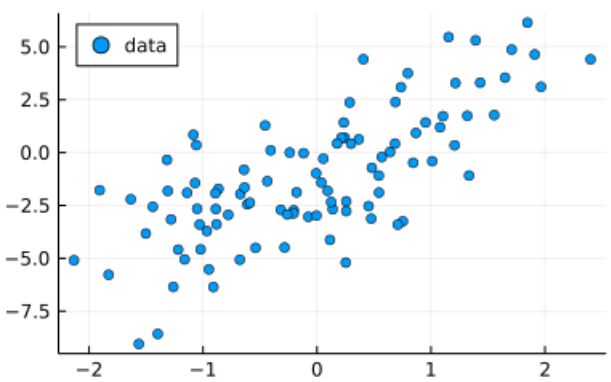
- 「標本分布について」のノート (https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/04%20Distribution%20of%20samples.ipynb)

の「最小二乗法による線形回帰」の節で得た公式に一致する.

2.9 Julia言語による回帰直線の計算の最も簡単な例

In [5]:
```
# 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);
y = @. -1 + 2x + 2e;
scatter(x, y; label="data", legend=:topleft)
```

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, :]
```

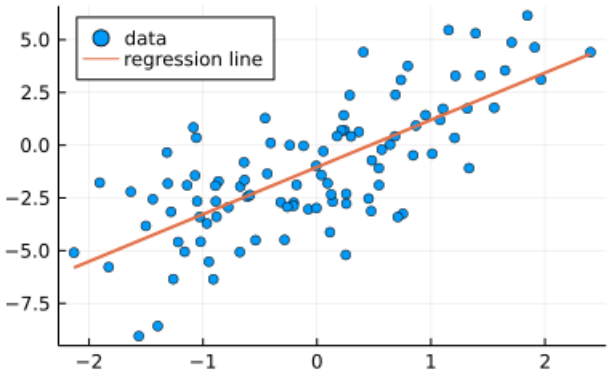
Out[6]:
```
5×2 Matrix{Float64}:
 1.0   0.843695
 1.0   0.286651
 1.0  -0.117361
 1.0   1.43215
 1.0  -1.39703
```

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}:
 -1.0404163851842536
  2.235682111246987
```

In [8]:
```
# 結果をプロット
scatter(x, y; label="data", legend=:topleft)
plot!(xstar -> β̂[1] + β̂[2]*xstar; label="regression line", lw=2)
```

Out[8]:



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.

2.10 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:

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

例えば, $\mu=(m,m,\dots,m)$, $\Sigma=\sigma^2 I$ (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.

2.11 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: 実対称行列 Σ は直交行列 U で対角化できる.

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

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

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

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,

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

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,

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

となるので, χ^2 は自由度 n のχ²分布に従う.

解答終

2.12 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.

2.13 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い, $\hat{s}^2 \approx \sigma^2$ 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,

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

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.

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:

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

真の回帰函数上の値 $f(x_*)^T\beta$ の信頼度 $1-\alpha$ 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:

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

2.14 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,

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

となることより,

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

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:

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:

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

真の回帰函数上の値 $f(x_*)^T\beta$ 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:

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

2.15 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,

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

ここで,

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

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 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:

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

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.

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)
- 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 [9]:
```
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 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

    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[9]:
```
plot_linreg_confint_predint (generic function with 1 method)
```

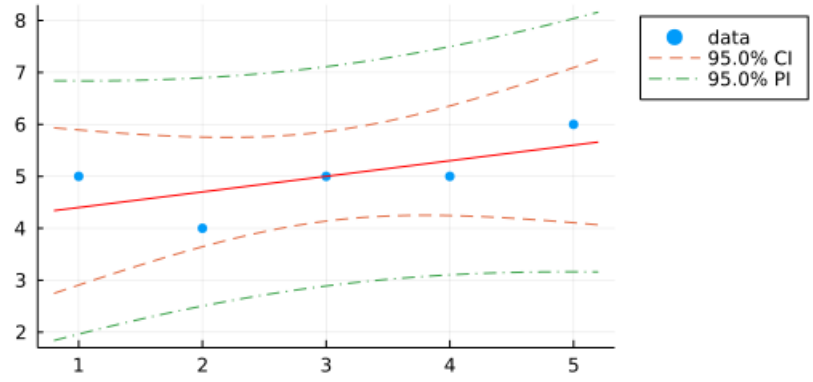
3.1.1 信頼区間と予測区間のテストプロット

In [10]:

```
x = 1:5
y = [5, 4, 5, 5, 6]
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:1)...)
plot!(ylim=(1.7, 8.3))
```

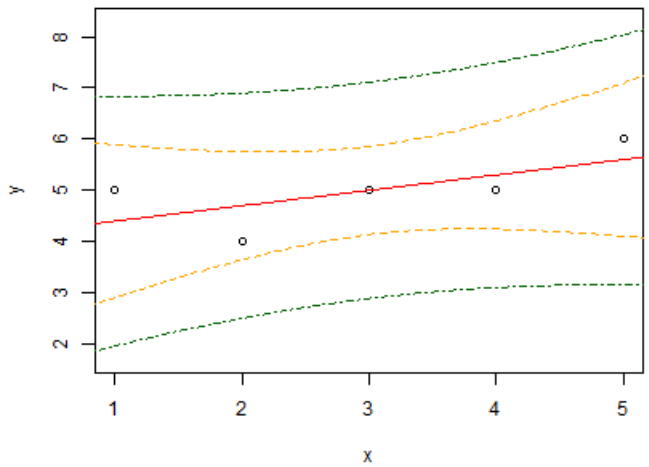
```
n = 5
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 3.1824463052837078
β̂ = [4.1, 0.29999999999999993]
ŝ = 0.6055300708194982
```

Out[10]:



In [11]:

```
@rput x y
reg = R"""reg = lm(y ~ x)"""
R"""
xstars = data.frame(x = seq(-0.5, 5.5, 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(1.7, 8.3), yaxp=c(2, 9, 7))
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[11]:

```
RObject{VecSxp}

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
        4.1          0.3
```

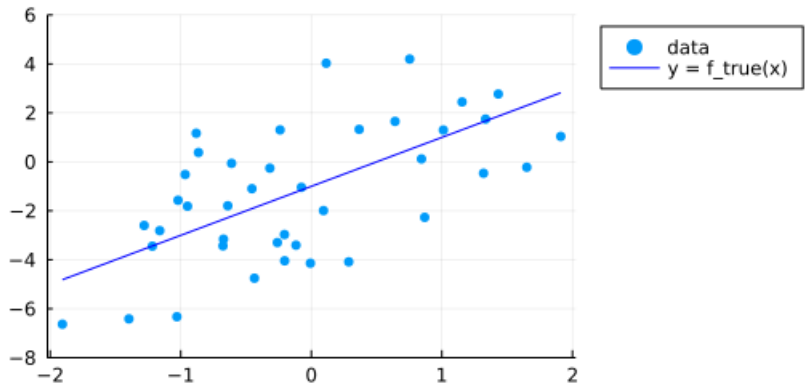
以上によって, このノートのコードの結果とRによる計算結果のプロットは一致していることがわかった.

3.1.2 回帰直線の信頼区間と予測区間

In [12]:

```
Random.seed!(4649373)

n = 40
f_true(x) = -1 + 2x
x = rand(Normal(0,1), 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, 6))
```

Out[12]:

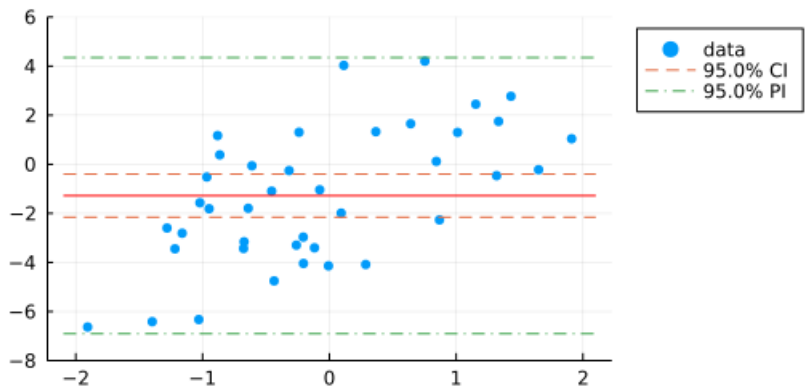


In [13]:

```
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:0)...)
plot!(ylim=(-8, 6))
```

```
n = 40
r = 1
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0226909200367604
β̂ = [-1.277501664364963]
ŝ = 2.745840198360994
```

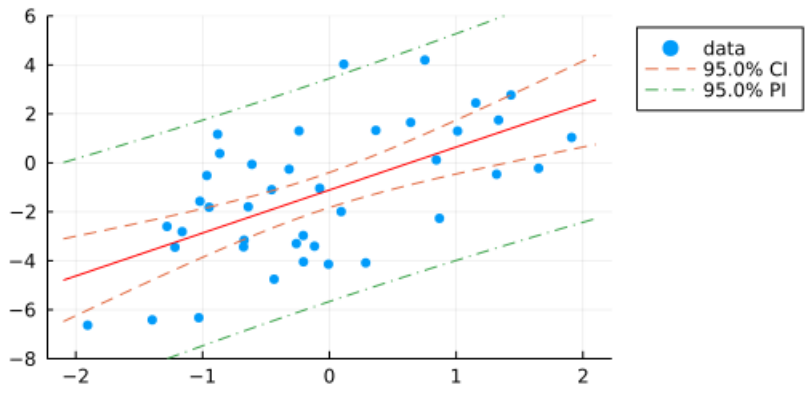
Out[13]:



In [14]:
```
plot_linreg_confint_predint(x, y, (x -> x^k for k in 0:1)...)
plot!(ylim=(-8, 6))
```

```
n = 40
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0243941639119694
β̂ = [-1.1102478114961913, 1.7549089886753286]
ŝ = 2.222022308287274
```

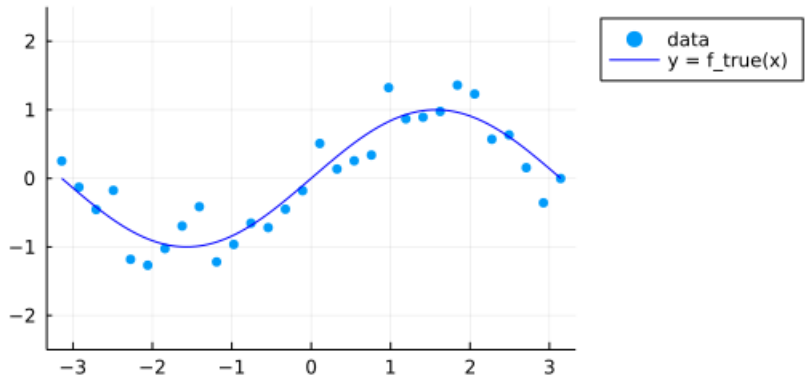
Out[14]:



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 [15]:
```
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))
```

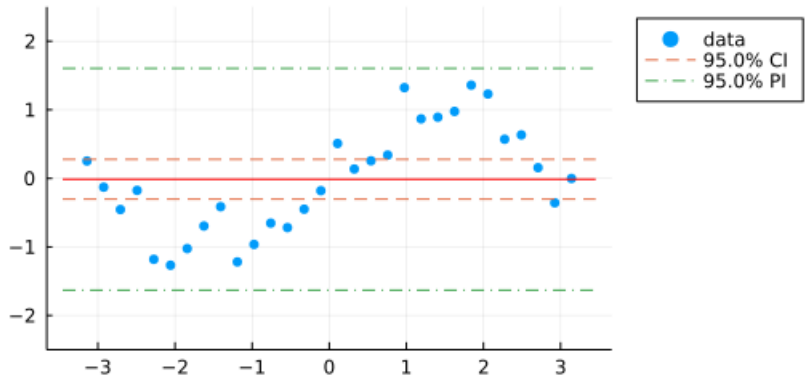
Out[15]:



In [16]:
```
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) = 2.0452296421327034
β̂ = [-0.012921569974331694]
ŝ = 0.7781229227974805
```

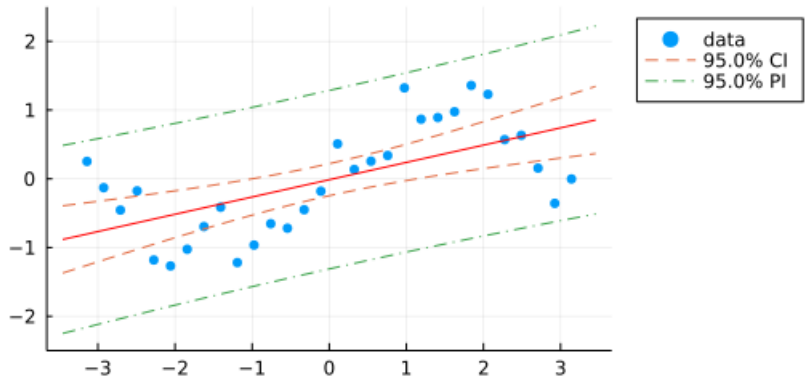
Out[16]:



In [17]:
```
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 = [-0.012921569974331675, 0.25146544548096655]
ŝ = 0.6235650870768796
```

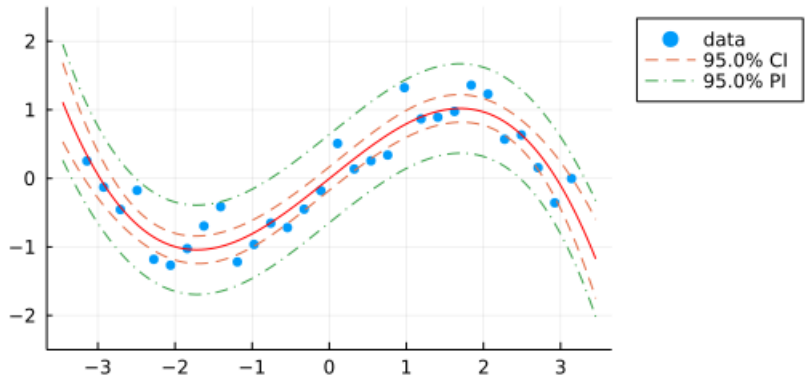
Out[17]:



In [18]:
```
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 = [-0.004868037850388715, 0.9055404522516881, -0.0022900460140340695, -0.10348024677024828]
ŝ = 0.3018043730527076
```

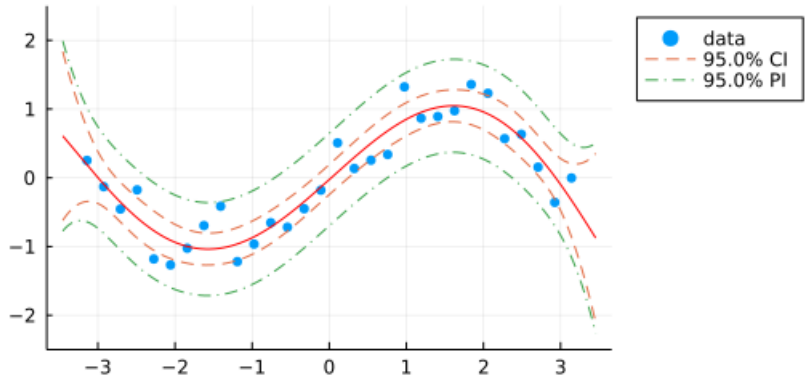
Out[18]:



In [19]:
```
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) = 2.0638985616280254
β̂ = [-0.026428141042331734, 1.0071582696511716, 0.0182525867063003, -0.14879998365691066, -0.002
2800938849420445, 0.003892016571707278]
ŝ = 0.3078837919499401
```

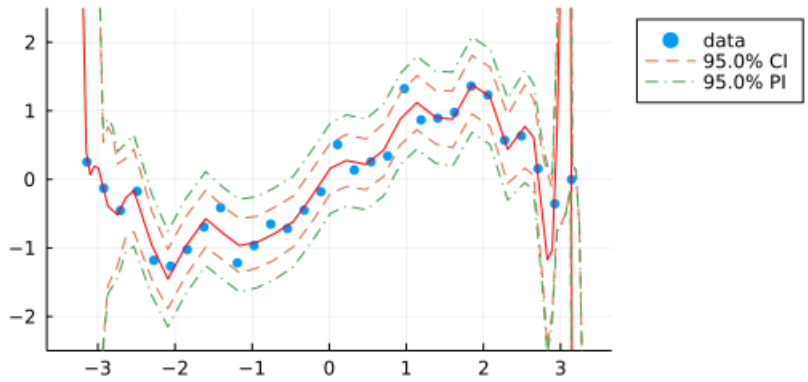
Out[19]:



In [20]:
```
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 = [0.14247028140681808, 1.2914469265448518, -2.862708649041329, -3.1780625037772205, 7.5180553
80487802, 7.790792095545583, -8.165629013870868, -8.17395305360277, 4.666047726772347, 4.3453609
37853182, -1.5269201557827485, -1.3012484041405847, 0.2944652989355421, 0.22929508226073686, -0.
03296579977722806, -0.023587239919145065, 0.001976582640073381, 0.0013103573390256058, -4.899517
106916385e-5, -3.036313618793969e-5]
ŝ = 0.24567191751997788
```

Out[20]:



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 [21]:
```
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
```

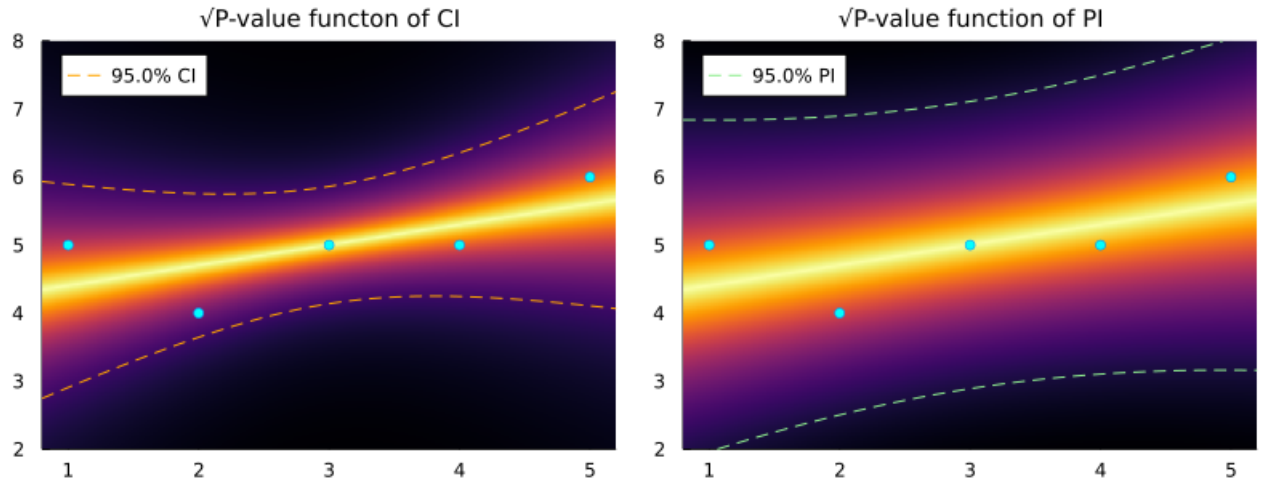
Out[21]:
```
plot_linreg_pvalue_functions (generic function with 1 method)
```

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

In [22]:
```
x = 1:5
y = [5, 4, 5, 5, 6]
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:1)...;
    ystars=range(2, 8, 400))
```

```
n = 5
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 3.1824463052837078
β̂ = [4.1, 0.29999999999999993]
ŝ = 0.6055300708194982
```

Out[22]:

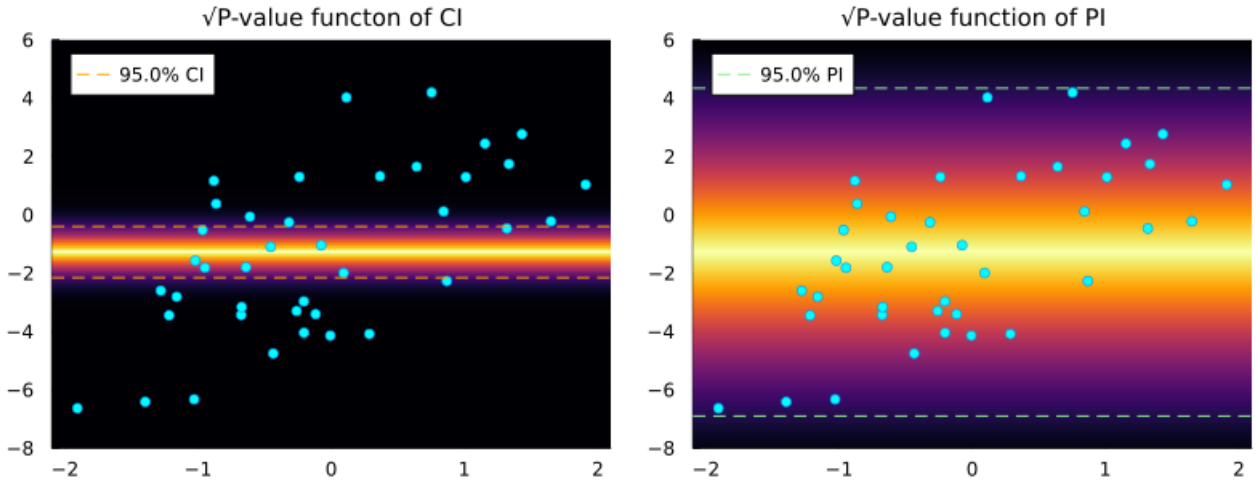


3.2.2 回帰直線の信頼区間と予測区間に対応するP値函数

In [23]:
```
Random.seed!(4649373)

n = 40
f_true(x) = -1 + 2x
x = rand(Normal(0,1), 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, 6, 400))
```

```
n = 40
r = 1
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0226909200367604
β̂ = [-1.277501664364963]
ŝ = 2.745840198360994
```

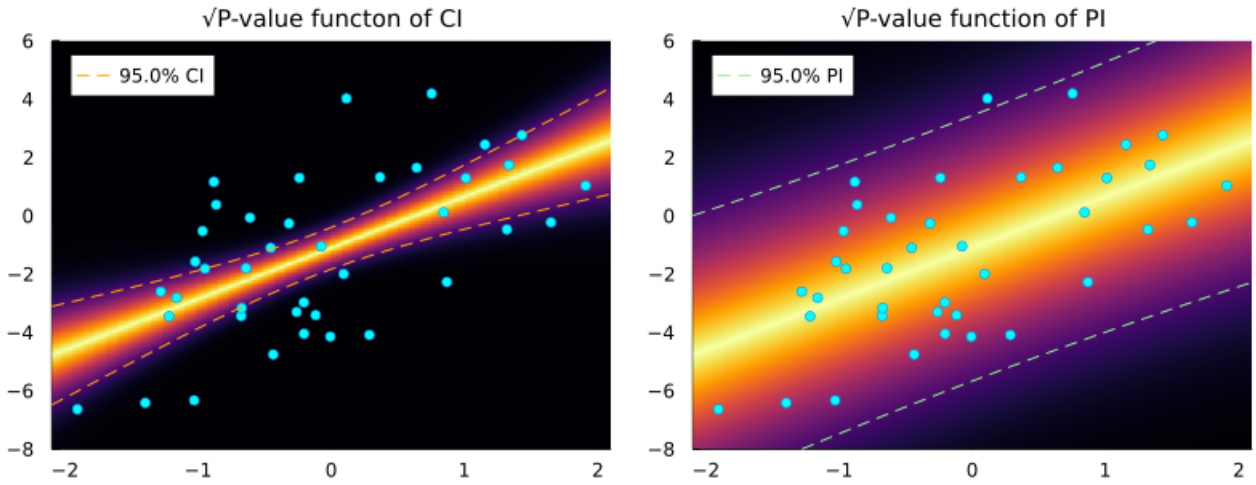
Out[23]:



In [24]:
```
plot_linreg_pvalue_functions(x, y, (x -> x^k for k in 0:1)...;
    ystars=range(-8, 6, 400))
```

```
n = 40
r = 2
α = 0.05
quantile(TDist(n - r), 1 - α / 2) = 2.0243941639119694
β̂ = [-1.1102478114961913, 1.7549089886753286]
ŝ = 2.222022308287274
```

Out[24]:

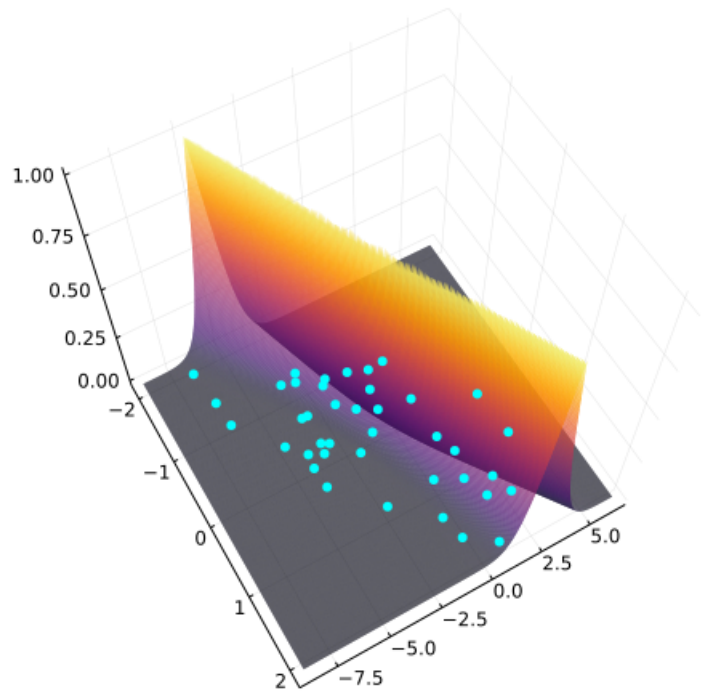


3.2.3 回帰直線の信頼区間に対応するP値函数の動画

In [25]:

```
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 X𝐑ʳ
    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
f_true(x) = -1 + 2x
x = rand(Normal(0,1), 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)...)
```

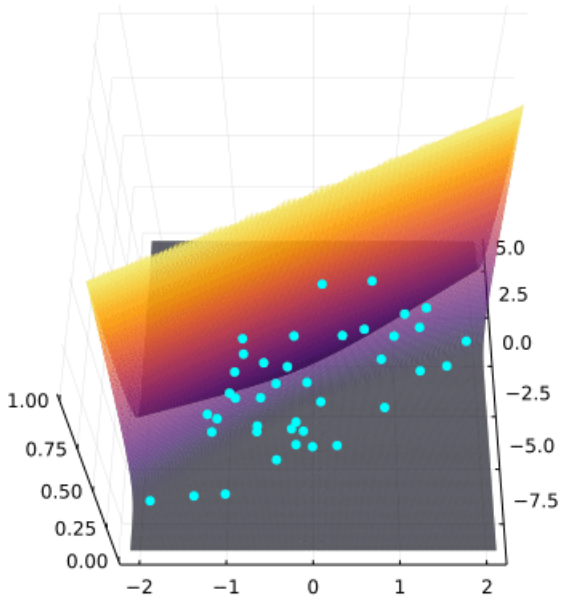
Out[25]:



In [26]:
```
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
│   fn = D:\OneDrive\work\Statistics\2022\images\anim_linreg_confint_pvalfunc_3d.gif
└ @ Plots D:\.julia\packages\Plots\MzlNY\src\animation.jl:130
```

Out[26]:



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)

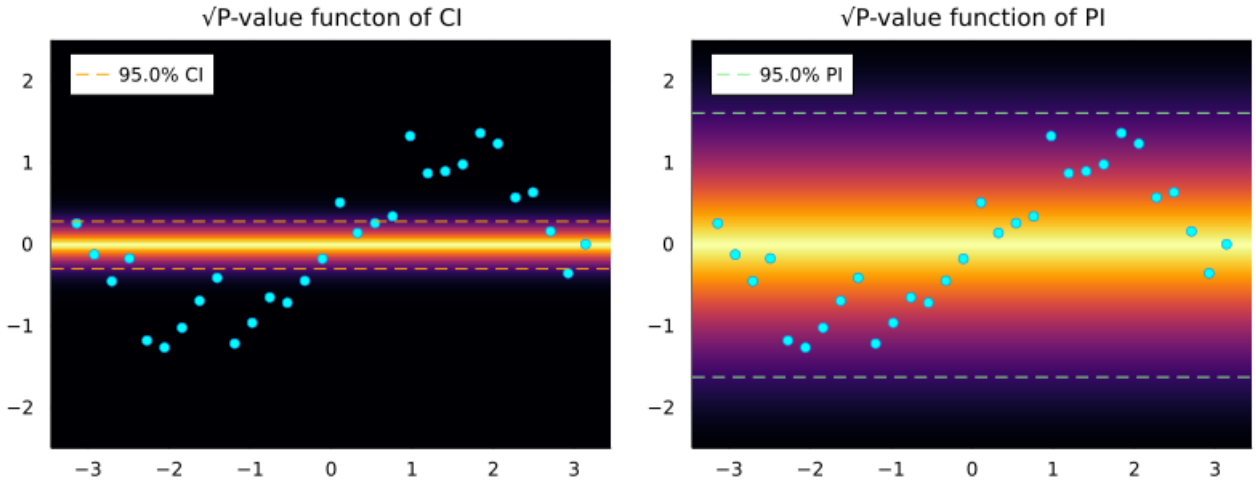
3.2.4 多項式回帰の信頼区間と予測区間に対応するP値函数

In [27]:

```
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) = 2.0452296421327034
β̂ = [-0.012921569974331694]
ŝ = 0.7781229227974805
```

Out[27]:

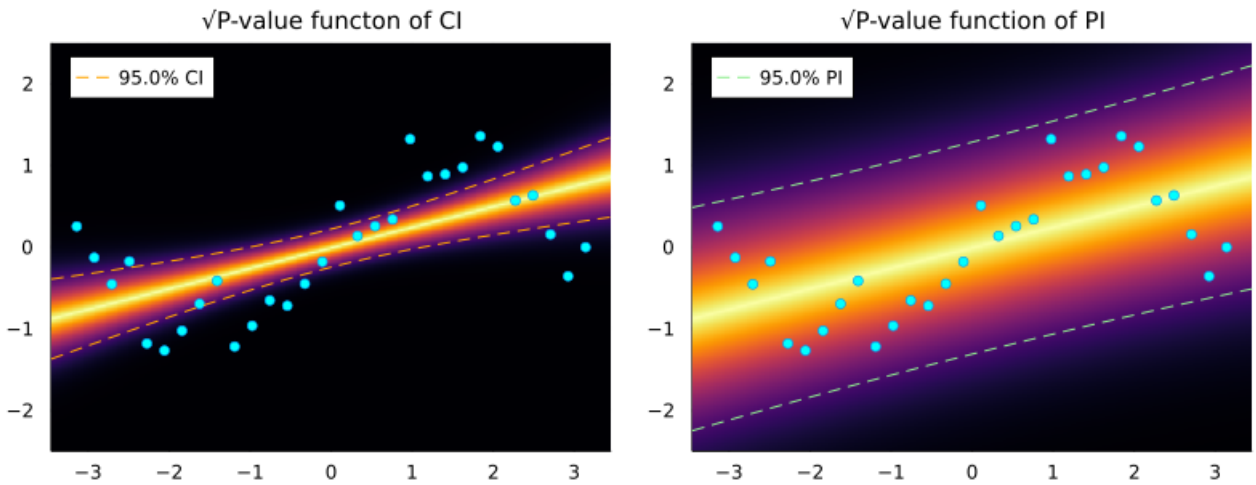


In [28]:

```
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 = [-0.012921569974331675, 0.25146544548096655]
ŝ = 0.6235650870768796
```

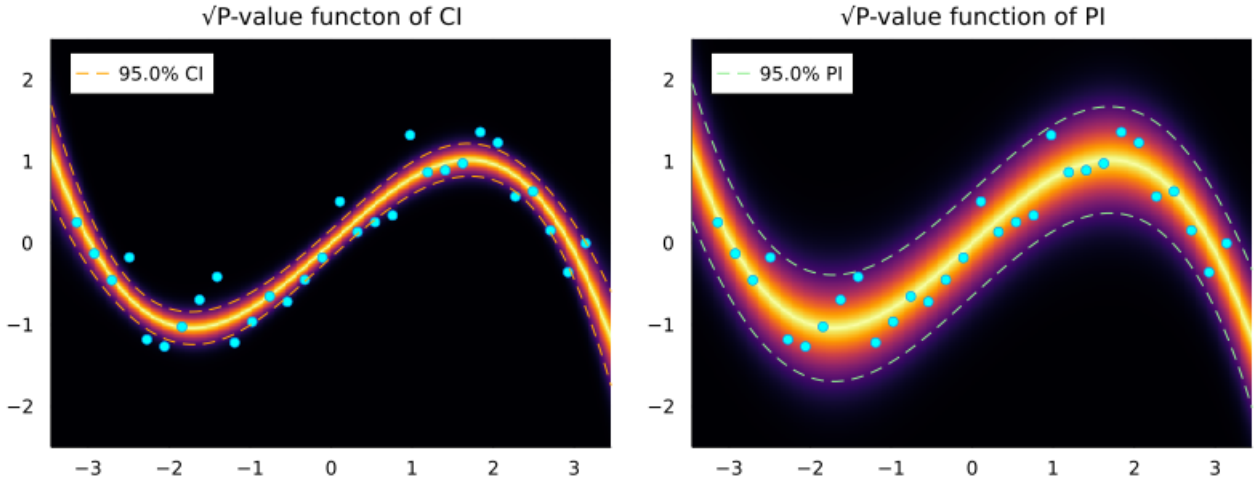
Out[28]:



In [29]:
```
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 = [-0.004868037850388715, 0.9055404522516881, -0.0022900460140340695, -0.10348024677024828]
ŝ = 0.3018043730527076
```

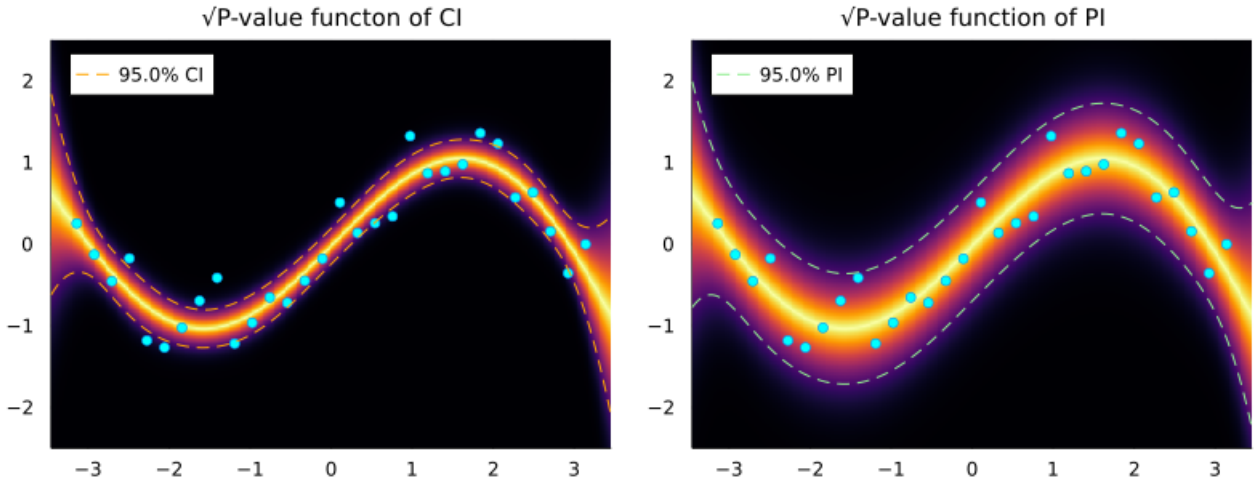
Out[29]:



In [30]:
```
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) = 2.0638985616280254
β̂ = [-0.026428141042331734, 1.0071582696511716, 0.0182525867063003, -0.14879998365691066, -0.002
2800938849420445, 0.003892016571707278]
ŝ = 0.3078837919499401
```

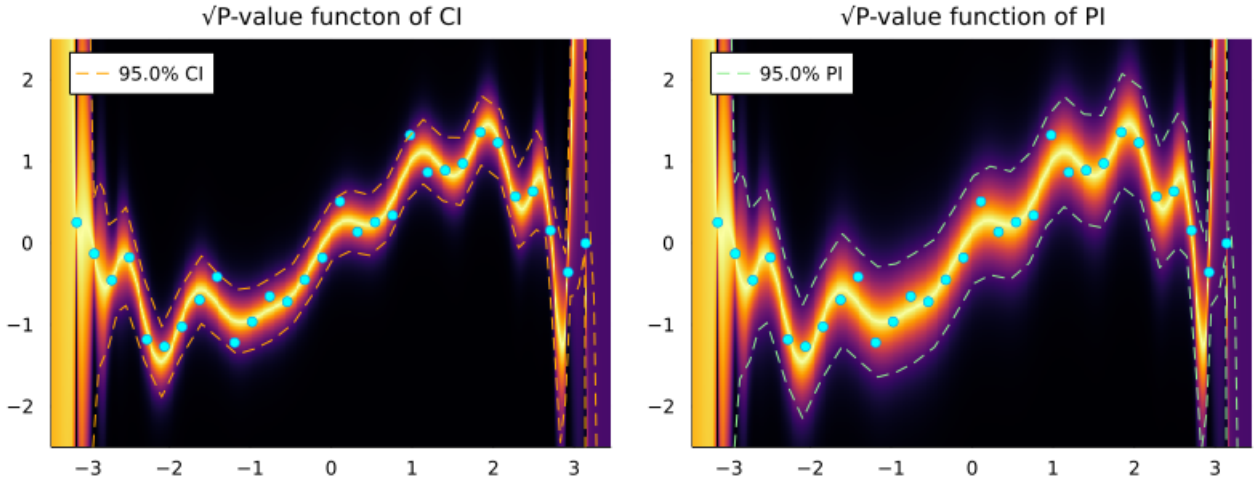
Out[30]:



In [31]:
```
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 = [0.14247028140681808, 1.2914469265448518, -2.862708649041329, -3.1780625037772205, 7.5180553
80487802, 7.790792095545583, -8.165629013870868, -8.17395305360277, 4.666047726772347, 4.3453609
37853182, -1.5269201557827485, -1.3012484041405847, 0.2944652989355421, 0.22929508226073686, -0.
03296579977722806, -0.023587239919145065, 0.001976582640073381, 0.0013103573390256058, -4.899517
106916385e-5, -3.036313618793969e-5]
ŝ = 0.24567191751997788
```

Out[31]:



4 ロジスティック回帰

In [ ]:
```
1
```