正規分布モデルの共役事前分布によるベイズ統計

- 黒木玄

- 2022-09-03〜2022-09-06

目次

In [1]:

```julia
ENV["COLUMNS"] = 120

using Distributions
using LinearAlgebra
using Random
using StatsPlots
default(fmt=:png, size=(500, 350),
    titlefontsize=10, tickfontsize=6, guidefontsize=9,
    plot_titlefontsize=10)
using SymPy
using Turing
```

In [2]:

```julia
# 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]:

```
# One sample t-test

function pvalue_ttest(x̄, s², n, μ)
    t = (x̄ - μ)/√(s²/n)
    2ccdf(TDist(n-1), abs(t))
end

function pvalue_ttest(x, μ)
    x̄, s², n = mean(x), var(x), length(x)
    pvalue_ttest(x̄, s², n, μ)
end

function confint_ttest(x̄, s², n; α = 0.05)
    c = quantile(TDist(n-1), 1-α/2)
    [x̄ - c*√(s²/n), x̄ + c*√(s²/n)]
end

function confint_ttest(x; α = 0.05)
    x̄, s², n = mean(x), var(x), length(x)
    confint_ttest(x̄, s², n; α)
end
```

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

In [4]:

```
# Bayesian analogue of one sample t-test

posterior_μ_ttest(n, x̄, s²) = x̄ + √(s²/n)*TDist(n-1)
posterior_μ_ttest(x) = posterior_μ_ttest(length(x), mean(x), var(x))

preddist_ttest(n, x̄, s²) = x̄ + √(s²*(1 + 1/n))*TDist(n-1)
preddist_ttest(x) = preddist_ttest(length(x), mean(x), var(x))
```

Out[4]: preddist_ttest (generic function with 2 methods)

In [5]:
```julia
# Jeffreys事前分布などのimproper事前分布を定義するために以下が使われる.

"""
    PowerPos(p::Real)

The *positive power distribution* with real-valued parameter `p` is the improper distribution
of real numbers that has the improper probability density function

```math
f(x) = \\begin{cases}
0 & \\text{if } x \\leq 0, \\\\
x^p & \\text{otherwise}.
\\end{cases}
```
"""
struct PowerPos{T<:Real} <: ContinuousUnivariateDistribution
    p::T
end
PowerPos(p::Integer) = PowerPos(float(p))

Base.minimum(d::PowerPos{T}) where T = zero(T)
Base.maximum(d::PowerPos{T}) where T = T(Inf)

Base.rand(rng::Random.AbstractRNG, d::PowerPos) = rand(rng) + 0.5
function Distributions.logpdf(d::PowerPos, x::Real)
    T = float(eltype(x))
    return x ≤ 0 ? T(-Inf) : d.p*log(x)
end

Distributions.pdf(d::PowerPos, x::Real) = exp(logpdf(d, x))

For vec support
function Distributions.loglikelihood(d::PowerPos, x::AbstractVector{<:Real})
    T = float(eltype(x))
    return any(xi ≤ 0 for xi in x) ? T(-Inf) : d.p*log(prod(x))
end

@doc PowerPos
```

Out[5]:
```
PowerPos(p::Real)
```

The *positive power distribution* with real-valued parameter p is the improper distribution of real numbers that has the improper probability density function

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ x^p & \text{otherwise}. \end{cases}$$

In [6]:
```julia
# 以下は使わないが,
# Flat() や PowerPos(p) と正規分布や逆ガンマ分布の関係は次のようになっている.

MyNormal(μ, σ) = σ == Inf ? Flat() : Normal(μ, σ)
MyInverseGamma(κ, θ) = θ == 0 ? PowerPos(-κ-1) : InverseGamma(κ, θ)
```

Out[6]:
```
MyInverseGamma (generic function with 1 method)
```

1 正規分布モデルの共役事前分布とその応用

1.1 逆ガンマ正規分布

平均 $\mu \in \mathbb{R}$, 分散 $v = \sigma^2 \in \mathbb{R}_{>0}$ の正規分布の確率密度函数を次のように表す:

$$p_{\text{Normal}}(y|\mu, v) = \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{1}{2v}(y-\mu)^2\right) \quad (y \in \mathbb{R}).$$

分散パラメータ σ^2 を v に書き直している理由は, σ^2 を1つの変数として扱いたいからである.

パラメータ $\kappa, \theta > 0$ の逆ガンマ分布の確率密度函数を次のように書くことにする:

$$p_{\text{InverseGamma}}(v|\kappa, \theta) = \frac{\theta^\kappa}{\Gamma(\kappa)} v^{-\kappa-1} \exp\left(-\frac{\theta}{v}\right) \quad (v > 0).$$

v がこの逆ガンマ分布に従う確率変数だとすると,

$$\frac{1}{v} \sim \mathrm{Gamma}\left(\kappa, \frac{1}{\theta}\right) = \frac{1}{2\theta}\mathrm{Gamma}\left(\frac{2\kappa}{2}, 2\right) = \frac{1}{2\theta}\mathrm{Chisq}(2\kappa),$$
$$E[v] = \frac{\theta}{\kappa - 1}, \quad \mathrm{var}(v) = \frac{E[v]^2}{\kappa - 2}.$$

A と B が μ, v に関する定数因子の違いを除いて等しいことを $A \propto B$ と書くことにする.

逆ガンマ正規分布の密度函数を次のように定義する:

$$\begin{aligned} p_{\mathrm{InverseGammaNormal}}(\mu, v|\mu_*, v_*, \kappa, \theta) &= p_{\mathrm{Normal}}(\mu|\mu_*, v_* v) p_{\mathrm{InverseGamma}}(v|\kappa, \theta) \\ &\propto v^{-(\kappa+1/2)-1} \exp\left(-\frac{1}{v}\left(\theta + \frac{1}{2v_*}(\mu - \mu_*)^2\right)\right). \end{aligned}$$

この逆ガンマ正規分布の密度函数に従う確率変数を μ, v と書くと,

$$E[v] = \frac{\theta}{\kappa - 1}, \quad \mathrm{var}(v) = \frac{E[v]^2}{\kappa - 2}, \quad \mathrm{cov}(\mu, v) = 0, \quad E[\mu] = \mu_*, \quad \mathrm{var}(\mu) = v_* E[v].$$

この逆ガンマ正規分布が正規分布の共役事前分布になっていることを次の節で確認する.

1.2 共役事前分布のBayes更新

データの数値 $y_1, \dots, y_n$ が与えられたとき, 正規分布モデルの尤度函数は

$$\prod_{i=1}^n p_{\mathrm{Normal}}(y_i|\mu, v) \propto v^{-n/2} \exp\left(-\frac{1}{2v}\sum_{i=1}^n (y_i - \mu)^2\right)$$

の形になる. このとき,

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

とおくと,

$$\sum_{i=1}^n (y_i - \mu)^2 = n(\mu - \bar{y})^2 + n\hat{\sigma}^2$$

なので, 尤度を最大化する μ, v は $\mu = \bar{y}, v = \hat{\sigma}^2$ になることがわかる.

さらに, 次が成立することもわかる:

$$\begin{aligned} &\prod_{i=1}^n p_{\mathrm{Normal}}(y_i|\mu, v) \times p_{\mathrm{InverseGammaNormal}}(\mu, v|\mu_*, v_*, \kappa, \theta) \\ &\propto v^{-n/2} \exp\left(-\frac{n}{2v}\left((\mu - \bar{y})^2 + \hat{\sigma}^2\right)\right) \times v^{-(\kappa+1/2)-1} \exp\left(-\frac{1}{v}\left(\theta + \frac{1}{2v_*}(\mu - \mu_*)^2\right)\right) \\ &= v^{-(\kappa+n/2+1/2)-1} \exp\left(-\frac{1}{v}\left(\theta + \frac{n}{2}\left(\hat{\sigma}^2 + \frac{(\bar{y} - \mu_*)^2}{1 + nv_*}\right) + \frac{1 + nv_*}{2v_*}\left(\mu - \frac{\mu_* + nv_*\bar{y}}{1 + nv_*}\right)^2\right)\right). \end{aligned}$$

ゆえに共役事前分布から得られる事後分布のパラメータは次のようになる:

$$\begin{aligned} \tilde{\kappa} &= \kappa + \frac{n}{2} = \frac{n}{2}\left(1 + \frac{2\kappa}{n}\right), \\ \tilde{\theta} &= \theta + \frac{n}{2}\left(\hat{\sigma}^2 + \frac{(\bar{y} - \mu_*)^2}{1 + nv_*}\right) = \frac{n\hat{\sigma}^2}{2}\left(1 + \frac{2\theta}{n\hat{\sigma}^2} + \frac{(\bar{y} - \mu_*)^2}{(1 + nv_*)\hat{\sigma}^2}\right), \\ \tilde{\mu}_* &= \frac{\mu_* + nv_*\bar{y}}{1 + nv_*} = \bar{y}\frac{1 + \mu_*/(nv_*\bar{y})}{1 + 1/(nv_*)}, \\ \tilde{v}_* &= \frac{v_*}{1 + nv_*} = \frac{1}{n}\frac{1}{1 + 1/(nv_*)}. \end{aligned}$$

In [7]:
```
function bayesian_update(μstar, vstar, κ, θ, n, ȳ, σ̂²)
    μstar_new = (μstar/vstar + n*ȳ)/(1/vstar + n)
    vstar_new = 1/(1/vstar + n)
    κ_new = κ + n/2
    θ_new = θ + (n/2)*(σ̂² + ((ȳ - μstar)^2/vstar)/(1/vstar + n))
    μstar_new, vstar_new, κ_new, θ_new
end

function bayesian_update(μstar, vstar, κ, θ, y)
    n, ȳ, σ̂² = length(y), mean(y), var(y; corrected=false)
    bayesian_update(μstar, vstar, κ, θ, n, ȳ, σ̂²)
end
```

Out[7]:
```
bayesian_update (generic function with 2 methods)
```

In [8]:
```
@vars n ȳ v̂ μ v μ0 v0 κ θ
```

Out[8]:
```
(n, ȳ, v̂, μ, v, μ0, v0, κ, θ)
```

In [9]:
```
negloglik = n/2*log(v) + n/(2v)*((μ - ȳ)^2 + v̂)
```

Out[9]:

$$\frac{n\log(v)}{2} + \frac{n\left(\hat{v} + \left(-\bar{y} + \mu\right)^2\right)}{2v}$$

In [10]:
```
neglogpri = (κ + 1//2 + 1)*log(v) + 1/v*(θ + 1/(2v0)*(μ-μ0)^2)
```

Out[10]:

$$\left(\kappa + \frac{3}{2}\right)\log(v) + \frac{\theta + \frac{(\mu-\mu_0)^2}{2v_0}}{v}$$

In [11]:
```
neglogpost = (κ + n/2 + 1//2 + 1)*log(v) +  1/v*(
    θ + n/2*(v̂ + 1/(1+n*v0)*(ȳ - μ0)^2) +
    (1 + n*v0)/(2v0)*(μ - (μ0 + n*v0*ȳ)/(1 + n*v0))^2)
```

Out[11]:

$$\left(\frac{n}{2} + \kappa + \frac{3}{2}\right)\log(v) + \frac{\frac{n\left(\hat{v} + \frac{(\bar{y}-\mu_0)^2}{nv_0+1}\right)}{2} + \theta + \frac{\left(\mu - \frac{nv_0\bar{y}+\mu_0}{nv_0+1}\right)^2(nv_0+1)}{2v_0}}{v}$$

In [12]:
```
simplify(negloglik + neglogpri - neglogpost)
```

Out[12]: 0

In [13]:
```
bayesian_update(μ0, v0, κ, θ, n, ȳ, v̂) ▷ collect
```

Out[13]:

$$\begin{bmatrix} \dfrac{n\bar{y} + \frac{\mu_0}{v_0}}{n + \frac{1}{v_0}} \\ \dfrac{1}{n + \frac{1}{v_0}} \\ \dfrac{n}{2} + \kappa \\ \dfrac{n\left(\hat{v} + \frac{(\bar{y}-\mu_0)^2}{v_0\left(n + \frac{1}{v_0}\right)}\right)}{2} + \theta \end{bmatrix}$$

1.3 μの周辺事前・事後分布および事前・事後予測分布

確率密度函数

$$p(\mu|\mu_*, v_*, \kappa, \theta) = \int_{\mathbb{R}_{>0}} p_{\mathrm{InverseGammaNormal}}(\mu, v|\mu_*, v_*, \kappa, \theta)\, dv$$

で定義されるμ の周辺事前分布は次になる:

$$\mu \sim \mu_* + \sqrt{\frac{\theta}{\kappa} v_*}\ \mathrm{TDist}(2\kappa).$$

確率密度函数

$$p_*(y_{\text{new}}|\mu_*, v_*, \kappa, \theta) = \iint_{\mathbb{R}\times\mathbb{R}_{>0}} p_{\text{Normal}}(y_{\text{new}}|\mu, v) p_{\text{InverseGammaNormal}}(\mu, v|\mu_*, v_*, \kappa, \theta)\, d\mu\, dv$$

で定義される y_{new} の事前予測分布は次になる:

$$y_{\text{new}} \sim \mu_* + \sqrt{\frac{\theta}{\kappa}(1+v_*)}\ \text{TDist}(2\kappa).$$

パラメータをBayes更新後のパラメータ

$$\begin{aligned}
\tilde{\kappa} &= \kappa + \frac{n}{2} = \frac{n}{2}\left(1 + \frac{2\kappa}{n}\right), \\
\tilde{\theta} &= \theta + \frac{n}{2}\left(\hat{\sigma}^2 + \frac{(\bar{y}-\mu_*)^2}{1+nv_*}\right) = \frac{n\hat{\sigma}^2}{2}\left(1 + \frac{2\theta}{n\hat{\sigma}^2} + \frac{(\bar{y}-\mu_*)^2}{(1+nv_*)\hat{\sigma}^2}\right), \\
\tilde{\mu}_* &= \frac{\mu_* + nv_*\bar{y}}{1+nv_*} = \bar{y}\frac{1+\mu_*/(nv_*\bar{y})}{1+1/(nv_*)}, \\
\tilde{v}_* &= \frac{v_*}{1+nv_*} = \frac{1}{n}\frac{1}{1+1/(nv_*)}.
\end{aligned}$$

に置き換えればこれは μ の周辺事後分布および事後予測分布になる.

その事後分布を使った区間推定の幅は

- n が大きいほど狭くなる.
- κ が大きいほど狭くなる.
- θ が大きいほど広くなる.
- $|\bar{y}-\mu_*|/\hat{\sigma}$ が大きいほど広くなる.
- $|\bar{y}-\mu_*|/\hat{\sigma}$ が大きくても, v_* がさらに大きければ狭くなる.

In [14]:
```
posterior_μ(μstar, vstar, κ, θ) = μstar + √(θ/κ*vstar)*TDist(2κ)
preddist(μstar, vstar, κ, θ) = μstar + √(θ/κ*(1 + vstar))*TDist(2κ)
```

Out[14]:
```
preddist (generic function with 1 method)
```

1.4 Jeffreys事前分布の場合

パラメータ空間が $\{(\mu, v) = (\mu, \sigma^2) \in \mathbb{R}\times\mathbb{R}_{>0}\}$ の 2 次元の正規分布モデルのJeffreys事前分布 $p_{\text{Jeffreys}}(\mu, v)$ は

$$p_{\text{Jeffreys}}(\mu, v) \propto v^{-3/2}$$

になることが知られている. ただし, 右辺の $(\mu, v) \in \mathbb{R}\times\mathbb{R}_{>0}$ に関する積分は ∞ になるので, この場合のJeffreys事前分布はimproperである.

逆ガンマ正規分布の密度函数

$$p_{\text{InverseGammaNormal}}(\mu, v|\mu_*, v_*, \kappa, \theta) \propto v^{-(\kappa+1/2)-1}\exp\left(-\frac{1}{v}\left(\theta + \frac{1}{2v_*}(\mu-\mu_*)^2\right)\right).$$

と比較すると, Jeffreys事前分布に対応する共役事前分布のパラメータ値は形式的に次になることがわかる:

$$\kappa \to 0, \quad \theta \to 0, \quad v_* \to \infty.$$

そのとき, Bayes更新後のパラメータの公式は次のようにシンプルになる:

$$\tilde{\kappa} = \frac{n}{2}, \quad \tilde{\theta} = \frac{n\hat{\sigma}^2}{2}, \quad \tilde{\mu}_* = \bar{y}, \quad \tilde{v}_* = \frac{1}{n}.$$

さらに, 前節の公式から, $n\to\infty$ のとき, 一般のパラメータ値に関するBayes更新の結果は, $n\to\infty$ のとき漸近的にこのJeffreys事前分布の場合に一致する.

さらに, Jeffreys事前分布の場合には

$$\frac{\tilde{\theta}}{\tilde{\kappa}} = \hat{\sigma}^2, \quad \tilde{v}_* = \frac{1}{n}, \quad 2\tilde{\kappa} = n.$$

ゆえに, μ に関する周辺事後分布は

$$\mu \sim \bar{y} + \frac{\hat{\sigma}}{\sqrt{n}}\,\mathrm{TDist}(n)$$

になり, 事後予測分布は次になる:

$$y_{\mathrm{new}} \sim \bar{y} + \hat{\sigma}\sqrt{1 + \frac{1}{n}}\,\mathrm{TDist}(n).$$

In [15]:
```julia
prior_jeffreys() = 0.0, Inf, 0.0, 0.0

posterior_μ_jeffreys(n, ȳ, σ̂²) = ȳ + √(σ̂²/n)*TDist(n)

function posterior_μ_jeffreys(y)
    n, ȳ, σ̂² = length(y), mean(y), var(y; corrected=false)
    posterior_μ_jeffreys(n, ȳ, σ̂²)
end

preddist_jeffreys(n, ȳ, σ̂²) = ȳ + √(σ̂²*(1+1/n))*TDist(n)

function preddist_jeffreys(y)
    n, ȳ, σ̂² = length(y), mean(y), var(y; corrected=false)
    preddist_jeffreys(n, ȳ, σ̂²)
end
```

Out[15]:
```
preddist_jeffreys (generic function with 2 methods)
```

In [16]:
```julia
μ_true, σ_true, n = 10, 3, 5
@show dist_true = Normal(μ_true, σ_true) n
y = rand(Normal(μ_true, σ_true), n)
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10.0, σ=3.0)
n = 5
```

Out[16]:
```
5-element Vector{Float64}:
 13.421461823230711
 10.091318547559782
  9.114689343604514
  8.142099605865505
 12.850119689447698
```

In [17]:
```julia
n, ȳ, σ̂² = length(y), mean(y), var(y; corrected=false)
```

Out[17]:
```
(5, 10.723937801941641, 4.290612291316625)
```

In [18]:
```julia
post_μ = posterior_μ(bayesian_update(prior_jeffreys()..., y)...)
```

Out[18]:
```
LocationScale{Float64, Continuous, TDist{Float64}}(
μ: 10.723937801941641
σ: 0.926348993772501
ρ: TDist{Float64}(ν=5.0)
)
```

In [19]:
```julia
posterior_μ_jeffreys(y) ≈ post_μ
```

Out[19]:
```
true
```

1.5 Jeffreys事前分布の場合の結果の数値的確認

In [20]:

```julia
# プロット用函数

function plot_posterior_μ(chn, y, postμ_theoretical;
        xlim = quantile.(postμ_theoretical, (0.0001, 0.9999)), kwargs...)
    postμ_ttest = posterior_μ_ttest(y)
    plot(legend=:outertop)
    if !isnothing(chn)
        stephist!(vec(chn[:μ]); norm=true,
            label="MCMC posterior of μ", c=1)
    end
    plot!(postμ_theoretical, xlim...;
        label="theoretical posterior of μ", c=2, ls=:dash)
    plot!(postμ_ttest, xlim...;
        label="ȳ+√(s²/n)TDist(n-1)", c=3, ls=:dashdot)
    plot!(; xlim, kwargs...)
end

function plot_preddist(chn, y, pred_theoretical;
        xlim = quantile.(pred_theoretical, (0.0001, 0.9999)), kwargs...)
    pdf_pred(y_new) = mean(pdf(Normal(μ, √σ²), y_new)
        for (μ, σ²) in zip(vec(chn[:μ]), vec(chn[:σ²])))
    pred_ttest = preddist_ttest(y)

    plot(legend=:outertop)
    if !isnothing(chn)
        plot!(pdf_pred, xlim...; label="MCMC prediction", c=1)
    end
    plot!(pred_theoretical, xlim...;
        label="theoretical prediction", c=2, ls=:dash)
    plot!(pred_ttest, xlim...;
        label="ȳ+√(s²(1+1/n))TDist(n-1)", c=3, ls=:dashdot)
    plot!(; kwargs...)
end
```

Out[20]:

```
plot_preddist (generic function with 1 method)
```

In [21]:

```julia
@model function normaldistmodel_jeffreys(y)
    σ² ~ PowerPos(-3/2)
    μ ~ Flat()
    y ~ MvNormal(fill(μ, length(y)), σ²*I)
end
```

Out[21]:

```
normaldistmodel_jeffreys (generic function with 2 methods)
```

In [22]:

```julia
μ_true, σ_true, n = 1e4, 1e2, 5
@show dist_true = Normal(μ_true, σ_true) n
y = rand(Normal(μ_true, σ_true), n)
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
n = 5
```

Out[22]:

```
5-element Vector{Float64}:
 10046.991358602998
 10090.945260495413
 10031.966223927362
 10049.736827492683
  9902.696010134123
```

In [23]:
```julia
L = 10^5
n_threads = min(Threads.nthreads(), 10)
chn = sample(normaldistmodel_jeffreys(y), NUTS(), MCMCThreads(), L, n_threads);
```

```
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
```

In [24]:
```julia
chn
```

Out[24]:
```
Chains MCMC chain (100000×14×10 Array{Float64, 3}):

Iterations        = 1001:1:101000
Number of chains  = 10
Samples per chain = 100000
Wall duration     = 26.53 seconds
Compute duration  = 240.38 seconds
parameters        = σ², μ
internals         = lp, n_steps, is_accept, acceptance_rate, log_density, hamiltonian_energy, ha
miltonian_energy_error, max_hamiltonian_energy_error, tree_depth, numerical_error, step_size, no
m_step_size

Summary Statistics
  parameters         mean         std   naive_se      mcse           ess      rhat   ess_per_sec
      Symbol      Float64     Float64    Float64   Float64       Float64   Float64       Float64

          σ²    6842.6405   9896.2407     9.8962   18.8615   268961.4025    1.0000     1118.9009
           μ   10024.4777     37.0518     0.0371    0.0625   338589.4340    1.0000     1408.5591

Quantiles
  parameters        2.5%       25.0%       50.0%       75.0%       97.5%
      Symbol     Float64     Float64     Float64     Float64     Float64

          σ²   1589.0321   3084.0481   4695.9500   7648.9445  24776.7344
           μ   9950.8760  10003.7177  10024.5403  10045.2740  10097.9316
```

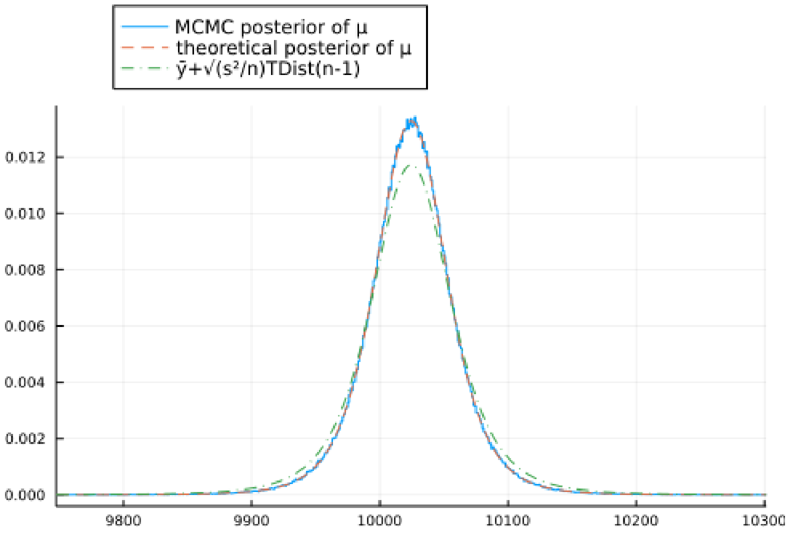
In [25]:
```julia
@show confint_ttest(y);
```

```
confint_ttest(y) = [9935.686679038185, 10113.24759322285]
```

In [26]:

```
postμ_theoretical = posterior_μ_jeffreys(y)
plot_posterior_μ(chn, y, postμ_theoretical)
```

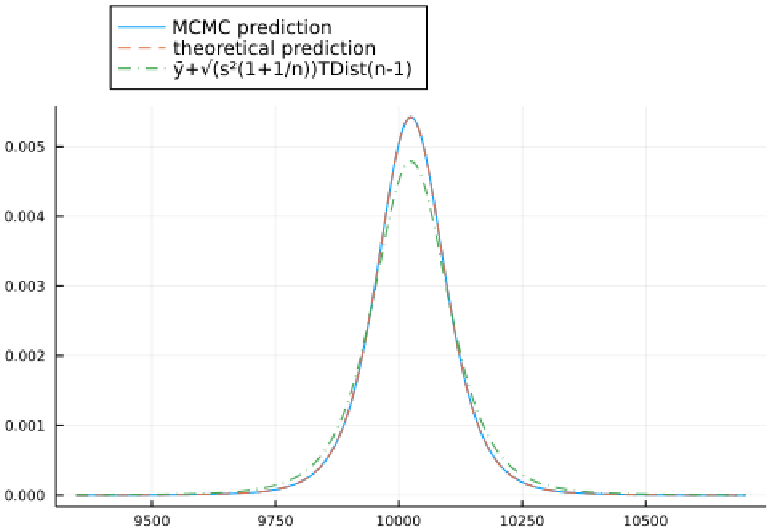
Out[26]:



In [27]:

```
pred_theoretical = preddist_jeffreys(y)
plot_preddist(chn, y, pred_theoretical)
```

Out[27]:



1.6 平均と対数分散について一様な事前分布の場合

平均 μ と分数の対数 $\log v = \log \sigma^2$ に関する一様な事前分布は

$$p_{\text{flat}}(\mu, v) \propto v^{-1}$$

になる. ただし, 右辺の $(\mu, v) \in \mathbb{R} \times \mathbb{R}_{>0}$ に関する積分は ∞ になるので, この事前分布はimproperである.

逆ガンマ正規分布の密度函数

$$p_{\text{InverseGammaNormal}}(\mu, v \mid \mu_*, v_*, \kappa, \theta) \propto v^{-(\kappa+1/2)-1} \exp\left(-\frac{1}{v}\left(\theta + \frac{1}{2v_*}(\mu - \mu_*)^2\right)\right).$$

と比較すると, 平均と対数分散について一様な事前分布に対応する共役事前分布のパラメータ値は形式的に次になることがわかる:

$$\kappa \to -\frac{1}{2}, \quad \theta \to 0, \quad v_* \to \infty.$$

このとき, Bayes更新後のパラメータの公式は次のようになる:

$$\tilde{\kappa} = \frac{n-1}{2}, \quad \tilde{\theta} = \frac{n\hat{\sigma}^2}{2}, \quad \tilde{\mu}_* = \bar{y}, \quad \tilde{v}_* = \frac{1}{n}.$$

この場合には

$$\frac{\tilde{\theta}}{\tilde{\kappa}} = \frac{n\hat{\sigma}^2}{n-1} = s^2, \quad \tilde{v}_* = \frac{1}{n}, \quad 2\tilde{\kappa} = n - 1.$$

ここで, s^2 はデータの数値 $y_1, \dots, y_n$ の不偏分散

$$s^2 = \frac{1}{n-1}\sum_{i=1}^n (y_i - \bar{y})^2 = \frac{n\hat{\sigma}^2}{n-1} > \hat{\sigma}^2$$

であり, s はその平方根である.

ゆえに, μ に関する周辺事後分布は

$$\mu \sim \bar{y} + \frac{s}{\sqrt{n}}\,\text{TDist}(n-1)$$

になり, y_{new} に関する事後予測分布は次になる:

$$y_{\text{new}} \sim \bar{y} + s\sqrt{1 + \frac{1}{n}}\,\text{TDist}(n-1).$$

したがって, 前節の結果と比較すると, Jeffreys事前分布の事後分布と予測分布による区間推定よりもこの場合の区間推定は少し広くなる.

In [28]:

```
prior_flat() = 0.0, Inf, -1/2, 0.0

posterior_μ_flat(n, ȳ, s²) = ȳ + √(s²/n)*TDist(n-1)

function posterior_μ_flat(y)
    n, ȳ, s² = length(y), mean(y), var(y)
    posterior_μ_flat(n, ȳ, s²)
end

preddist_flat(n, ȳ, s²) = ȳ + √(s²*(1+1/n))*TDist(n-1)

function preddist_flat(y)
    n, ȳ, s² = length(y), mean(y), var(y)
    preddist_flat(n, ȳ, s²)
end
```

Out[28]:

```
preddist_flat (generic function with 2 methods)
```

In [29]:

```
y = rand(Normal(10, 3), 5)
@show dist_true = Normal(μ_true, σ_true) n
n, ȳ, s² = length(y), mean(y), var(y)
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
n = 5
```

Out[29]:

```
(5, 9.547963580248759, 4.072331700326717)
```

In [30]:
```julia
post_μ = posterior_μ(bayesian_update(prior_flat()..., y)...)
```

Out[30]:
```
LocationScale{Float64, Continuous, TDist{Float64}}(
μ: 9.547963580248759
σ: 0.9024778889620196
ρ: TDist{Float64}(ν=4.0)
)
```

In [31]:
```julia
posterior_μ_flat(y) ≈ post_μ
```

Out[31]:
```
true
```

1.7 平均と対数分散について一様な事前分布の場合の結果の数値的確認

In [32]:
```julia
@model function normaldistmodel_flat(y)
    σ² ~ PowerPos(-1)
    μ ~ Flat()
    y ~ MvNormal(fill(μ, length(y)), σ²*I)
end
```

Out[32]:
```
normaldistmodel_flat (generic function with 2 methods)
```

In [33]:
```julia
μ_true, σ_true, n = 1e4, 1e2, 5
@show dist_true = Normal(μ_true, σ_true) n
y = rand(Normal(μ_true, σ_true), n)
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
n = 5
```

Out[33]:
```
5-element Vector{Float64}:
 10042.778128366943
 10062.9725375297
  9923.845841751234
  9965.230031011248
 10006.793852352781
```

In [34]:
```julia
L = 10^5
n_threads = min(Threads.nthreads(), 10)
chn = sample(normaldistmodel_flat(y), NUTS(), MCMCThreads(), L, n_threads);
```

```
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
```

In [35]:
```
chn
```

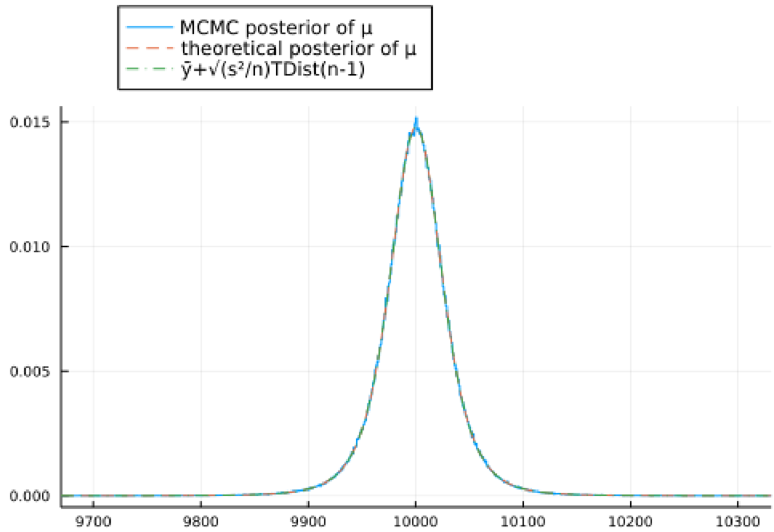
Out[35]:
```
Chains MCMC chain (100000×14×10 Array{Float64, 3}):

Iterations        = 1001:1:101000
Number of chains  = 10
Samples per chain = 100000
Wall duration     = 19.96 seconds
Compute duration  = 165.38 seconds
parameters        = σ², μ
internals         = lp, n_steps, is_accept, acceptance_rate, log_density, hamiltonian_energy, ha
miltonian_energy_error, max_hamiltonian_energy_error, tree_depth, numerical_error, step_size, no
m_step_size

Summary Statistics
  parameters        mean          std    naive_se      mcse           ess      rhat   ess_per_se
c
      Symbol     Float64      Float64     Float64   Float64       Float64   Float64      Float6
4

          σ²   6432.2097   16654.5449     16.6545   41.5589   155370.8686    1.0000      939.506
4
           μ  10000.2627      36.0629      0.0361    0.0714   251118.1140    1.0000     1518.476
9

Quantiles
  parameters       2.5%      25.0%      50.0%      75.0%      97.5%
      Symbol    Float64    Float64    Float64    Float64    Float64

          σ²  1153.4809  2388.5418  3834.0137  6685.0097  26551.8170
           μ  9929.8062  9981.5035 10000.3199 10019.0844  10070.3459
```

In [36]:
```
@show confint_ttest(y);
```

```
confint_ttest(y) = [9929.949145045925, 10070.69901135884]
```

In [37]:
```
postμ_theoretical = posterior_μ_flat(y)
plot_posterior_μ(chn, y, postμ_theoretical)
```

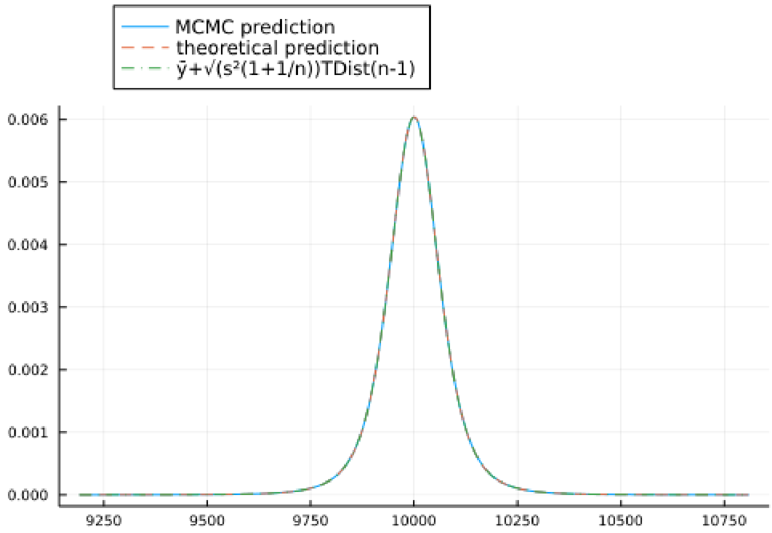
Out[37]:



In [38]:

```
pred_theoretical = preddist_flat(y)
plot_preddist(chn, y, pred_theoretical)
```

Out[38]:



1.8 通常の信頼区間と予測区間との比較

通常の t 分布を使う平均の信頼区間と次の値の予測区間の構成では以下を使う:

$$\frac{\bar{y}-\mu}{s/\sqrt{n}} \sim \mathrm{TDist}(n-1), \qquad \frac{y_{\mathrm{new}}-\bar{y}}{s\sqrt{1+1/n}} \sim \mathrm{TDist}(n-1).$$

ここで, s^2 はデータの数値の不偏分散であり, s はその平方根である.

したがって, 前節の結果と比較すると, 通常の信頼区間と予測区間は, 平均と対数分散に関する一様事前分布に関する事後分布と予測分布を用いた区間推定に一致する.

1.9 データの数値から事前分布を決めた場合

$a, b > 0$ であると仮定する.

データの数値から共役事前分布のパラメータを次の条件によって決めたと仮定する:

$$E[\mu] = \mu_* = \bar{y}, \quad E[v] = \frac{\theta}{\kappa-1} = \hat{\sigma}^2, \quad \mathrm{var}(\mu) = v_* E[v] = a\hat{\sigma}^2, \quad \mathrm{var}(v) = \frac{E[v]^2}{\kappa-2} = b\hat{\sigma}^4.$$

これは次と同値である:

$$\mu_* = \bar{y}, \quad v_* = a, \quad \kappa = 2 + \frac{1}{b}, \quad \theta = \hat{\sigma}^2\left(1+\frac{1}{b}\right).$$

このパラメータ値に対応する共役事前分布を以下では **適応事前分布** (adaptive prior)と呼ぶことにする(注意: ここだけの用語).

これのBayes更新の結果は以下のようになる:

$$
\begin{aligned}
\tilde{\kappa} &= 2 + \frac{1}{b} + \frac{n}{2} = \frac{n}{2}\left(1 + \frac{2(2+1/b)}{n}\right) && \to 2 + \frac{n}{2}, \\
\tilde{\theta} &= \hat{\sigma}^2\left(1 + \frac{1}{b} + \frac{n}{2}\right) + \frac{n}{2}\frac{(\bar{y}-\bar{y})^2}{1+na} = \frac{n\hat{\sigma}^2}{2}\left(1 + \frac{2(1+1/b))}{n}\right) && \to \hat{\sigma}^2\left(1 + \frac{n}{2}\right), \\
\tilde{\mu}_* &= \frac{\bar{y} + n v_* \bar{y}}{1 + n v_*} = \bar{y} && \to \bar{y}, \\
\tilde{v}_* &= \frac{a}{1+na} = \frac{1}{n}\frac{1}{1+1/(na)} && \to \frac{1}{n}.
\end{aligned}
$$

以上における $\to$ は $a \to \infty$, $b \to \infty$ での極限を意味する.

適応事前分布の構成のポイントは, $\mu_* = \bar{y}$ となっているおかげで, $\tilde{\mu}_*$ も $\tilde{\mu}_* = \bar{y}$ となってバイアスが消え, さらに, $\tilde{\theta}$ の中の $\dfrac{n}{2}\dfrac{(\bar{y}-\mu_*)^2}{1+na}$ の項が消えて, 区間推定の幅が無用に広くならずに済むことである.

ただし, 適応事前分布の場合には

$$
\frac{\tilde{\theta}}{\tilde{\kappa}} = \hat{\sigma}^2\frac{1 + 2(1+1/b)/n}{1 + 2(2+1/b)/n} < \hat{\sigma}^2, \quad v_* = \frac{1}{n}\frac{1}{1+1/(na)} < \frac{1}{n}
$$

なので, 区間推定の幅はJeffreys事前分布の場合よりも少し狭くなる.

しかし, n が大きければそれらの違いは小さくなる.

In [39]:
```
function prior_adaptive(n, ȳ, σ̂²; a = 2.5, b = 2.5)
    μstar = ȳ
    vstar = a
    κ = 2 + 1/b
    θ = σ̂²*(1 + 1/b)
    μstar, vstar, κ, θ
end

function prior_adaptive(y; a = 2.5, b = 2.5)
    n, ȳ, σ̂² = length(y), mean(y), var(y; corrected=false)
    prior_adaptive(n, ȳ, σ̂²; a, b)
end

function posterior_adaptive(n, ȳ, σ̂²; a = 2.5, b = 2.5)
    μstar = ȳ
    vstar = 1/(1/a + n)
    κ = 2 + 1/b + n/2
    θ = σ̂²*(1 + 1/b + n/2)
    μstar, vstar, κ, θ
end

function posterior_adaptive(y; a = 2.5, b = 2.5)
    n, ȳ, σ̂² = length(y), mean(y), var(y; corrected=false)
    posterior_adaptive(n, ȳ, σ̂²; a, b)
end
```

Out[39]:
```
posterior_adaptive (generic function with 2 methods)
```

In [40]:
```
μ_true, σ_true, n = 1e4, 1e2, 5
@show dist_true = Normal(μ_true, σ_true) n
y = rand(Normal(μ_true, σ_true), n)
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
n = 5
```

Out[40]:
```
5-element Vector{Float64}:
  9933.795478608836
 10197.691591107683
  9873.296478779508
  9968.595731251091
  9775.778787905336
```

In [41]:
```
n, ȳ, σ̂² = length(y), mean(y), var(y; corrected=false)
```

Out[41]:
```
(5, 9949.831613530489, 19639.166236215202)
```

In [42]:
```
μstar, vstar, κ, θ = prior_adaptive(y)
a, b = 2.5, 2.5
@show ȳ, σ̂², a*σ̂², b*σ̂²^2
(ȳ, σ̂², a*σ̂², b*σ̂²^2) .≈ (μstar, θ/(κ - 1), (θ/(κ - 1))*vstar, (θ/(κ - 1))^2/(κ - 2))
```

```
(ȳ, σ̂², a * σ̂², b * σ̂² ^ 2) = (9949.831613530489, 19639.166236215202, 49097.915590538, 9.6424212
6134238e8)
```

Out[42]:
```
(true, true, true, true)
```

In [43]:
```
posterior_adaptive(n, ȳ, σ̂²)
```

Out[43]:
```
(9949.831613530489, 0.18518518518518517, 4.9, 76592.74832123928)
```

In [44]:
```
bayesian_update(prior_adaptive(y)..., y)
```

Out[44]:
```
(9949.831613530489, 0.18518518518518517, 4.9, 76592.74832123928)
```

In [45]:
```
posterior_adaptive(y)
```

Out[45]:
```
(9949.831613530489, 0.18518518518518517, 4.9, 76592.74832123928)
```

In [46]:
```
posterior_adaptive(y) .≈ bayesian_update(prior_adaptive(y)..., y)
```

Out[46]:
```
(true, true, true, true)
```

1.10 n = 5 では適応事前分布の場合と無情報事前分布の場合の結果が結構違う.

In [47]:
```
@model function normaldistmodel_adaptive(y; a = 2.5, b = 2.5)
    μstar, vstar, κ, θ = prior_adaptive(y; a, b)
    σ² ~ InverseGamma(κ, θ)
    μ ~ Normal(μstar, √(vstar * σ²))
    y ~ MvNormal(fill(μ, length(y)), σ²*I)
end
```

Out[47]:
```
normaldistmodel_adaptive (generic function with 2 methods)
```

In [48]:
```
μ_true, σ_true, n = 1e4, 1e2, 5
@show dist_true = Normal(μ_true, σ_true) n
y = rand(Normal(μ_true, σ_true), n)
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
n = 5
```

Out[48]:
```
5-element Vector{Float64}:
 10030.5799588623
  9998.014042880812
 10059.31532070359
 10005.17272597195
  9931.520071491572
```

In [49]:
```julia
L = 10^5
n_threads = min(Threads.nthreads(), 10)
chn = sample(normaldistmodel_adaptive(y), NUTS(), MCMCThreads(), L, n_threads);
```

```
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
```

In [50]:
```julia
chn
```

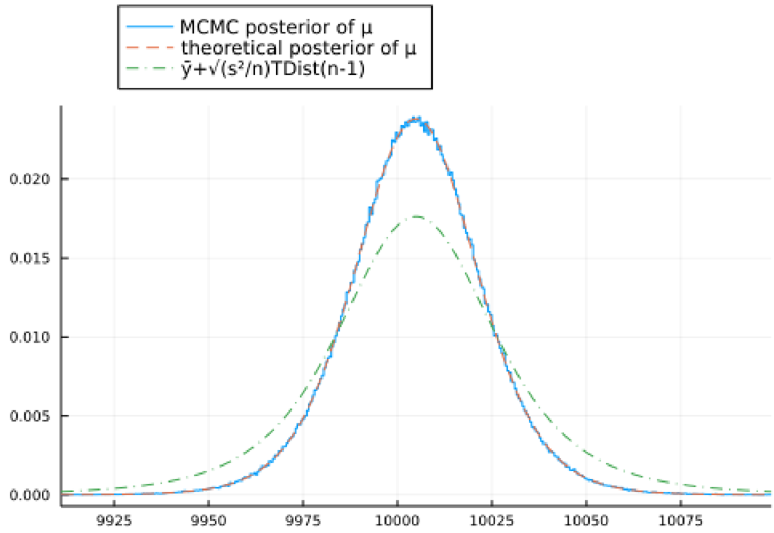
Out[50]:
```
Chains MCMC chain (100000×14×10 Array{Float64, 3}):

Iterations        = 1001:1:101000
Number of chains  = 10
Samples per chain = 100000
Wall duration     = 18.5 seconds
Compute duration  = 167.67 seconds
parameters        = σ², μ
internals         = lp, n_steps, is_accept, acceptance_rate, log_density, hamiltonian_energy, ha
miltonian_energy_error, max_hamiltonian_energy_error, tree_depth, numerical_error, step_size, no
m_step_size

Summary Statistics
  parameters        mean        std   naive_se      mcse           ess     rhat   ess_per_sec
      Symbol     Float64    Float64    Float64   Float64       Float64  Float64       Float64

          σ²   1811.4635  1073.8114     1.0738    1.5523   502189.8472   1.0000     2995.0549
           μ  10004.9134    18.3083     0.0183    0.0238   614003.2042   1.0000     3661.9086

Quantiles
  parameters        2.5%      25.0%      50.0%      75.0%      97.5%
      Symbol     Float64    Float64    Float64    Float64    Float64

          σ²    698.3470  1145.3723  1546.2427  2150.0831  4503.5897
           μ   9968.3987  9993.4939 10004.9111 10016.3625 10041.3650
```

In [51]:
```julia
@show confint_ttest(y);
```

```
confint_ttest(y) = [9945.851301450186, 10063.989546513903]
```

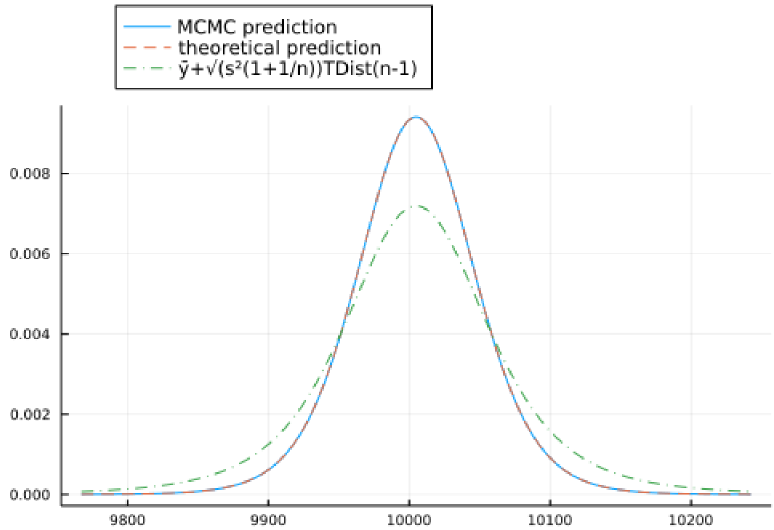
In [52]:
```
postμ_theoretical = posterior_μ(posterior_adaptive(y)...)
plot_posterior_μ(chn, y, postμ_theoretical)
```

Out[52]:



In [53]:
```
pred_theoretical = preddist(posterior_adaptive(y)...)
plot_preddist(chn, y, pred_theoretical)
```

Out[53]:



以上のように $n = 5$ の場合には, 適応事前分布の場合の結果は無情報事前分布の場合の結果(緑のdashdotライン)とかなり違う.

1.11 n = 20 ではデフォルト事前分布の場合と無情報事前分布の場合の結果が近付く.

In [54]:
```
μ_true, σ_true, n = 1e4, 1e2, 20
@show dist_true = Normal(μ_true, σ_true)
y = rand(dist_true, n)
@show length(y) mean(y) var(y);
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
length(y) = 20
mean(y) = 10021.312271029448
var(y) = 13942.23507271364
```

In [55]:
```
L = 10^5
n_threads = min(Threads.nthreads(), 10)
chn = sample(normaldistmodel_adaptive(y), NUTS(), MCMCThreads(), L, n_threads);
```

```
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
```

In [56]:
```
chn
```

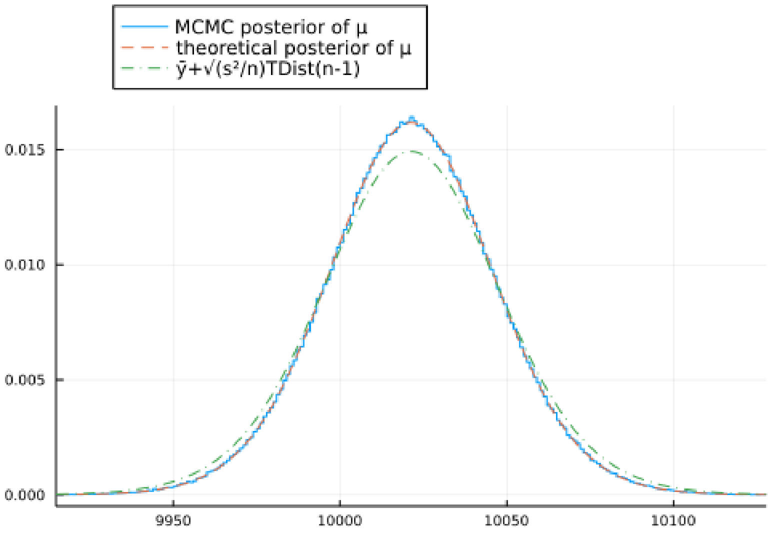
Out[56]:
```
Chains MCMC chain (100000×14×10 Array{Float64, 3}):

Iterations        = 1001:1:101000
Number of chains  = 10
Samples per chain = 100000
Wall duration     = 15.84 seconds
Compute duration  = 151.17 seconds
parameters        = σ², μ
internals         = lp, n_steps, is_accept, acceptance_rate, log_density, hamiltonian_energy, ha
miltonian_energy_error, max_hamiltonian_energy_error, tree_depth, numerical_error, step_size, no
m_step_size

Summary Statistics
  parameters         mean         std   naive_se      mcse           ess      rhat   ess_per_sec
      Symbol      Float64     Float64    Float64   Float64       Float64   Float64       Float64

          σ²   13252.0016   4113.1679     4.1132    4.6948   725981.9288    1.0000     4802.4842
           μ   10021.3589     25.5266     0.0255    0.0282   833465.2790    1.0000     5513.5034

Quantiles
  parameters        2.5%       25.0%       50.0%       75.0%       97.5%
      Symbol     Float64     Float64     Float64     Float64     Float64

          σ²   7475.8314  10373.7813  12516.5525  15294.5654  23285.6434
           μ   9970.8576  10004.6664  10021.3395  10038.0758  10071.8367
```

In [57]:
```
@show confint_ttest(y);
```

```
confint_ttest(y) = [9966.05042071424, 10076.574121344656]
```

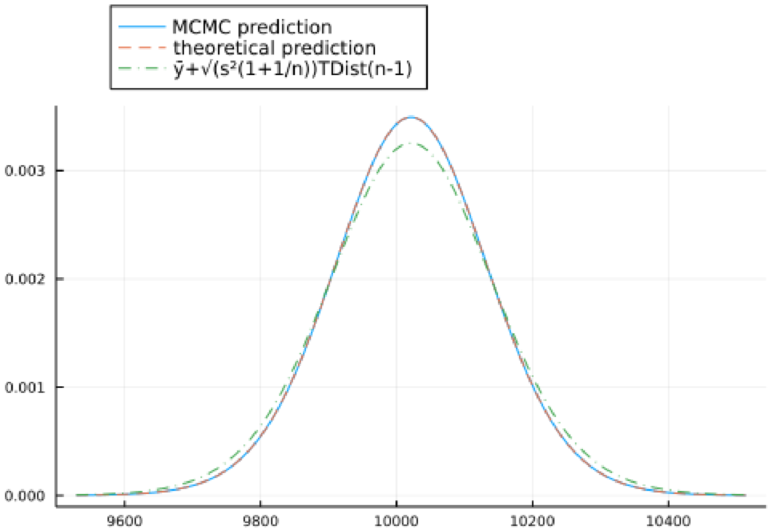
In [58]:
```
postμ_theoretical = posterior_μ(posterior_adaptive(y)...)
plot_posterior_μ(chn, y, postμ_theoretical)
```

Out[58]:



In [59]:
```
pred_theoretical = preddist(posterior_adaptive(y)...)
plot_preddist(chn, y, pred_theoretical)
```

Out[59]:



1.12 n = 20 で事前分布とデータの数値の相性が悪い場合

In [60]:
```
@model function normaldistmodel(y, μstar, vstar, κ, θ)
    σ² ~ InverseGamma(κ, θ)
    μ ~ Normal(μstar, √(vstar * σ²))
    y ~ MvNormal(fill(μ, length(y)), σ²*I)
end
```

Out[60]: normaldistmodel (generic function with 2 methods)

In [61]:
```
# 固定された事前分布の設定

a, b = 5.0^2, 5.0^2
μstar, vstar, κ, θ = 0.0, a, 2 + 1/b, 1 + 1/b
@show μstar vstar κ θ
println()

Eμ, Ev = μstar, θ/(κ - 1)
var_μ, var_v = vstar*Ev, Ev^2/(κ - 2)
@show Eμ Ev var_μ var_v;
```

```
μstar = 0.0
vstar = 25.0
κ = 2.04
θ = 1.04

Eμ = 0.0
Ev = 1.0
var_μ = 25.0
var_v = 24.99999999999998
```

以下では以上のようにして定めた事前分布を使う.

この事前分布における μ の平均と分散はそれぞれ 0 と 5^2 であり, $v = \sigma^2$ の平均と分散はそれぞれ 1 と 5^2 である.

In [62]:
```
μ_true, σ_true, n = 1e4, 1e2, 20
@show dist_true = Normal(μ_true, σ_true)
y = rand(dist_true, n)
@show length(y) mean(y) var(y);
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
length(y) = 20
mean(y) = 9975.2893410502
var(y) = 9047.15859738387
```

平均と分散がそれぞれ $10000, 100^2$ の正規分布でサイズ 20 のサンプルを生成している.

平均 10000 と分散 100^2 は上で定めた事前分布と極めて相性が悪い.

In [63]:
```
L = 10^5
n_threads = min(Threads.nthreads(), 10)
chn = sample(normaldistmodel(y, μstar, vstar, κ, θ), NUTS(), MCMCThreads(), L, n_threads);
```

```
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
│   isfinite.((θ, r, ℓπ, ℓκ)) = (true, false, false, false)
└ @ AdvancedHMC D:\.julia\packages\AdvancedHMC\51xgc\src\hamiltonian.jl:47
┌ Warning: The current proposal will be rejected due to numerical error(s).
```

In [64]:
```
chn
```

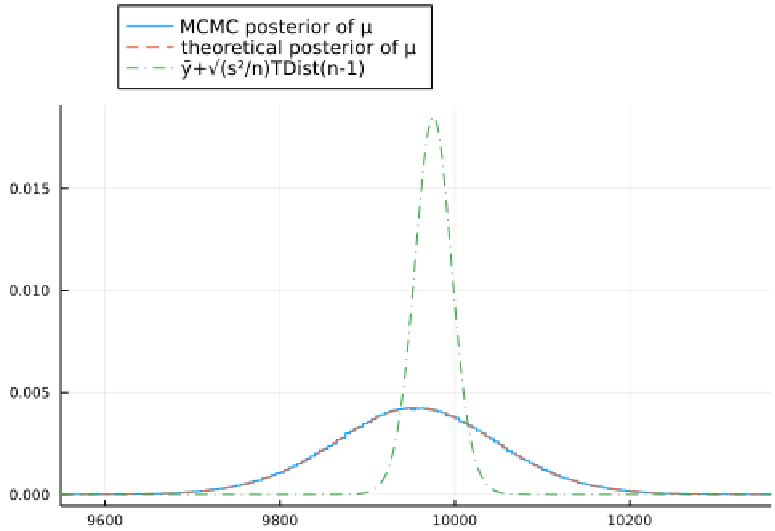
Out[64]:
```
Chains MCMC chain (100000×14×10 Array{Float64, 3}):

Iterations        = 1001:1:101000
Number of chains  = 10
Samples per chain = 100000
Wall duration     = 19.88 seconds
Compute duration  = 193.25 seconds
parameters        = σ², μ
internals         = lp, n_steps, is_accept, acceptance_rate, log_density, hamiltonian_energy, ha
miltonian_energy_error, max_hamiltonian_energy_error, tree_depth, numerical_error, step_size, no
m_step_size

Summary Statistics
  parameters         mean          std    naive_se       mcse            ess       rhat    ess_per_s
ec
      Symbol      Float64      Float64     Float64    Float64        Float64    Float64        Float
64

          σ²  187678.3265   59231.3056     59.2313    66.8673    754361.6817     1.0000      3903.61
39
           μ    9955.3048      96.8643      0.0969     0.1054    865190.5498     1.0000      4477.12
28

Quantiles
  parameters          2.5%         25.0%         50.0%         75.0%         97.5%
      Symbol       Float64       Float64       Float64       Float64       Float64

          σ²   104916.3421   146309.4780   177025.1066   216837.5925   332730.2136
           μ     9763.6855     9891.8617     9955.3523    10018.9425    10146.6951
```

In [65]:
```
@show confint_ttest(y);
```

```
confint_ttest(y) = [9930.773424097964, 10019.805258002436]
```

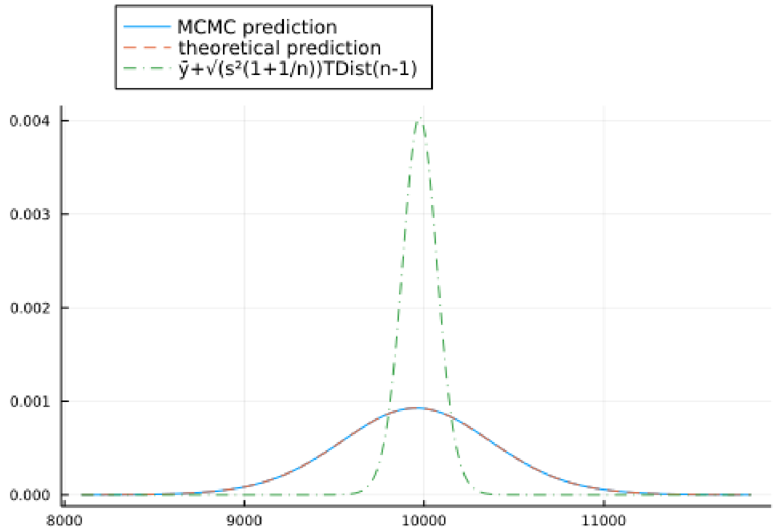
In [66]:
```
postμ_theoretical = posterior_μ(bayesian_update(μstar, vstar, κ, θ, y)...)
plot_posterior_μ(chn, y, postμ_theoretical)
```

Out[66]:



In [67]:
```
pred_theoretical = preddist(bayesian_update(μstar, vstar, κ, θ, y)...)
plot_preddist(chn, y, pred_theoretical)
```

Out[67]:



1.13 n = 200 で事前分布とデータの数値の相性が悪い場合

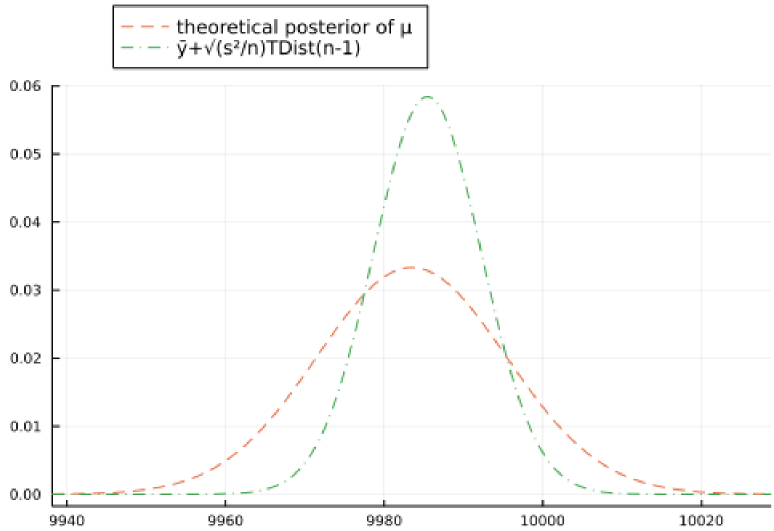
前節の続き

In [68]:
```
μ_true, σ_true, n = 1e4, 1e2, 200
@show dist_true = Normal(μ_true, σ_true)
y = rand(dist_true, n)
@show length(y) mean(y) var(y);
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
length(y) = 200
mean(y) = 9985.454981200888
var(y) = 9312.384006700653
```

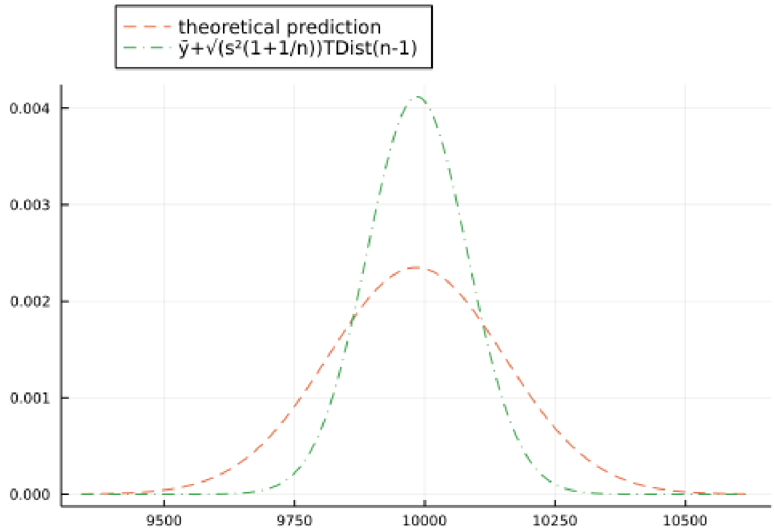
In [69]:
```
postμ_theoretical = posterior_μ(bayesian_update(μstar, vstar, κ, θ, y)...)
plot_posterior_μ(nothing, y, postμ_theoretical)
```

Out[69]:



In [70]:
```
pred_theoretical = preddist(bayesian_update(μstar, vstar, κ, θ, y)...)
plot_preddist(nothing, y, pred_theoretical)
```

Out[70]:



1.14 n = 2000 で事前分布とデータの数値の相性が悪い場合

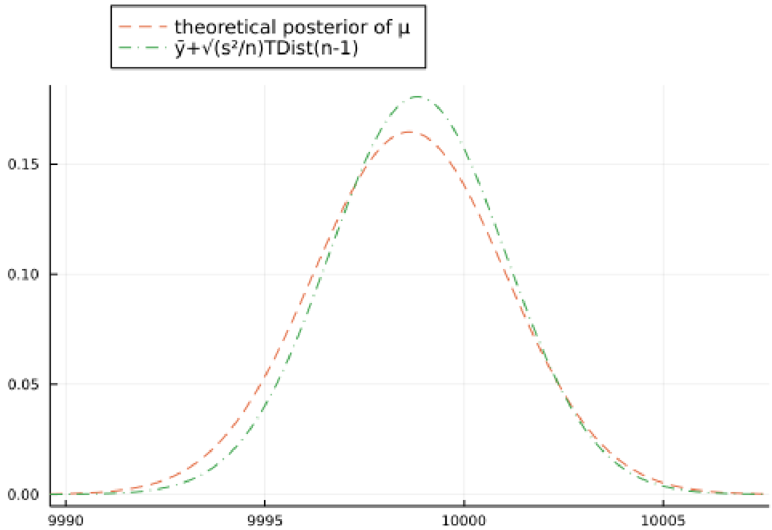
前節の続き

In [71]:
```
μ_true, σ_true, n = 1e4, 1e2, 2000
@show dist_true = Normal(μ_true, σ_true)
y = rand(dist_true, n)
@show length(y) mean(y) var(y);
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
length(y) = 2000
mean(y) = 9998.829716287655
var(y) = 9762.499844804266
```

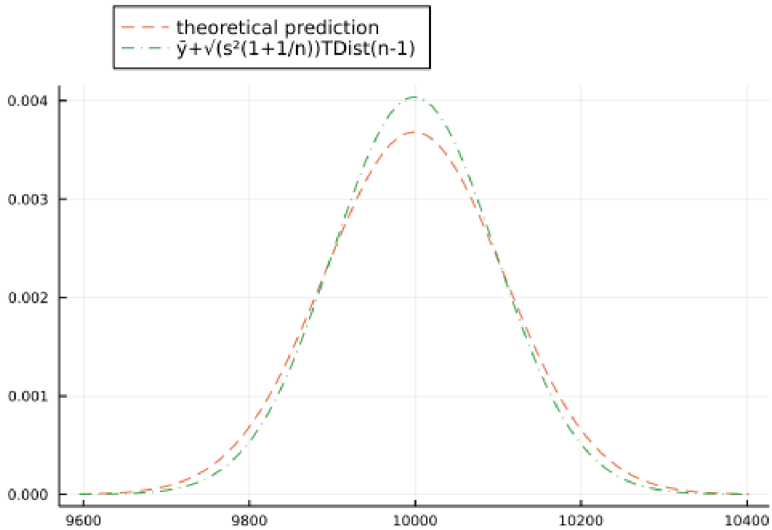
In [72]:
```
postμ_theoretical = posterior_μ(bayesian_update(μstar, vstar, κ, θ, y)...)
plot_posterior_μ(nothing, y, postμ_theoretical)
```

Out[72]:



In [73]:
```
pred_theoretical = preddist(bayesian_update(μstar, vstar, κ, θ, y)...)
plot_preddist(nothing, y, pred_theoretical)
```

Out[73]:



1.15 n = 20000 で事前分布とデータの数値の相性が悪い場合

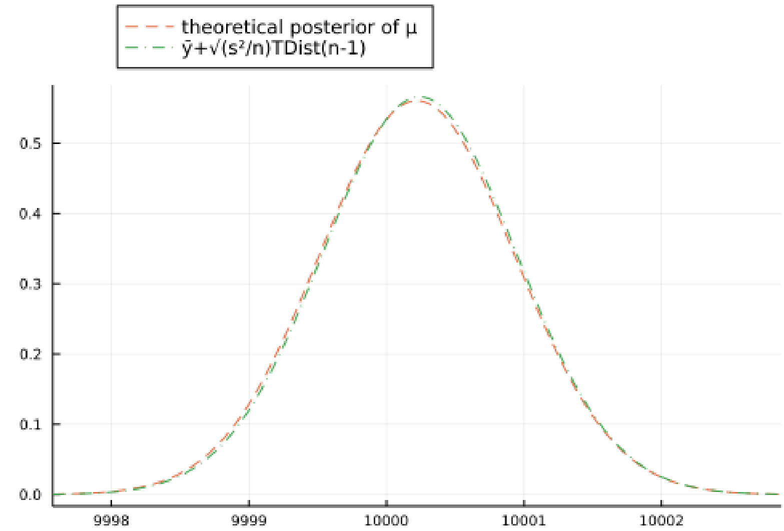
前節の続き

In [74]:
```
μ_true, σ_true, n = 1e4, 1e2, 20000
@show dist_true = Normal(μ_true, σ_true)
y = rand(dist_true, n)
@show length(y) mean(y) var(y);
```

```
dist_true = Normal(μ_true, σ_true) = Normal{Float64}(μ=10000.0, σ=100.0)
length(y) = 20000
mean(y) = 10000.24090421595
var(y) = 9932.528226665096
```

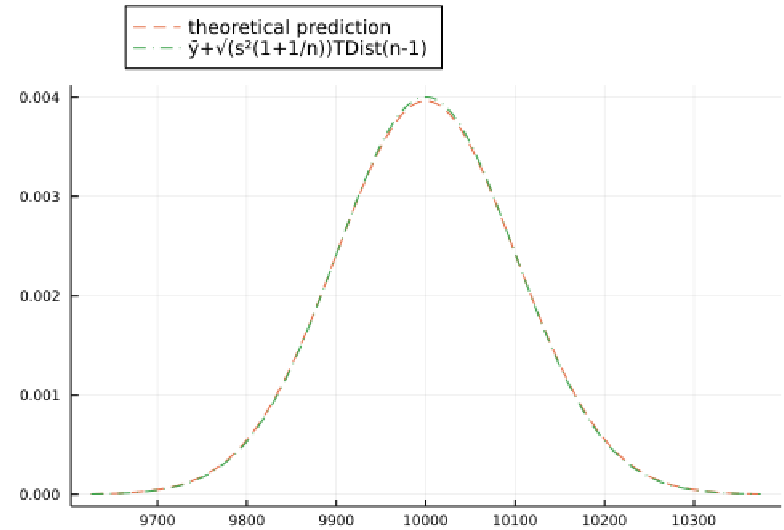
In [75]:
```
postμ_theoretical = posterior_μ(bayesian_update(μstar, vstar, κ, θ, y)...)
plot_posterior_μ(nothing, y, postμ_theoretical)
```

Out[75]:



In [76]:
```
pred_theoretical = preddist(bayesian_update(μstar, vstar, κ, θ, y)...)
plot_preddist(nothing, y, pred_theoretical)
```

Out[76]:



In [ ]:
```

```