Brunner-Munzel検定について

- 黒木玄

- 2022-08-05, 2022-09-19, 2023-09-13

文献

- E. Brunner and U. Munzel. The nonparametric Behrens-Fisher problem: Asymptotic theory and a small-sample approximation. Biometrical Journal, 42:17–25, 2000. [pdf (https://www.researchgate.net/profile/Edgar-Brunner/publication/264799502_Nonparametric_Hypotheses_and_Rank_Statistics_for_Unbalanced_Factorial_Designs/links/57
Hypotheses-and-Rank-Statistics-for-Unbalanced-Factorial-Designs.pdf)]
- Karin Neubert and Edgar Brunner, A studentized permutation test for the non-parametric Behrens-Fisher problem, Computational Statistics and Data Analysis, Vol. 51, pp. 5192-5204 (2007). https://doi.org/10.1016/j.csda.2006.05.024 (https://doi.org/10.1016/j.csda.2006.05.024)
- Claus P. Nowak, Markus Pauly, Edgar Brunner. The nonparametric Behrens-Fisher problem in small samples. https://arxiv.org/abs/2208.01231 (https://arxiv.org/abs/2208.01231)

目次

1 準備

1.1 パッケージの読み込みなど

In [1]:

```
using Base.Threads
using BenchmarkTools
using Distributions
using PrettyPrinting
using QuadGK
using Random
using RCall
using Roots
using StatsBase
using StatsFuns
using StatsPlots
default(fmt=:png, size=(400, 250),
    titlefontsize=10, guidefontsize=8, tickfontsize=6)

x ≲ y = x < y || x ≈ y
x ≳ y = x > y || x ≈ y
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
```

Out[1]:

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


Loading web-font STIX-Web/Size1/Regular

1.2 組み合わせの生成子

In [2]:

```
"""
    nextcombination!(n, t, c = typeof(t)[min(t-1, i) for i in 1:t])

`[1,2,…,n]` からの重複無しの `t` 個の組み合わせ `c` をすべて生成したい.

`nextcombination!(n, t, c)` は配列で表現された組み合わせ `c` をその次の組み合わせに書き換

初期条件を `c = typeof(t)[min(t-1, i) for i in 1:t]` にすると,
`binomial(n, t)` 回の `nextcombination!(n, t, c)` ですべての組み合わせが生成される.
"""
function nextcombination!(n, t, c = typeof(t)[min(t-1, i) for i in 1:t])
    t == 0 && return c
    @inbounds for i in t:-1:1
        c[i] += 1
        c[i] > (n - (t - i)) && continue
        for j in i+1:t
            c[j] = c[j-1] + 1
        end
        break
    end
    c
end

"""
    mycombinations!(n::Integer, t, c)

事前に割り当てられた組み合わせを格納する配列 `c` を使って,
`[1,2,…,n]` からの重複無しの `t` 個の組み合わせのすべてを生成する生成子を返す.
"""
function mycombinations!(n::Integer, t, c)
    for i in 1:t c[i] = min(t - 1, i) end
    (nextcombination!(n, t, c) for _ in 1:binomial(n, t))
end

"""
    mycombinations!(a, t, c)

事前に割り当てられた組み合わせを格納する配列 `c` を使って,
配列 `a` からのインデックスに重複がない `t` 個の組み合わせのすべてを生成する生成子を返す.
"""
function mycombinations!(a, t, c)
    t < 0 && (t = length(a) + 1)
    (view(a, indices) for indices in mycombinations!(length(a), t, c))
end

"""
    mycombinations(x, t)

`x` が整数ならば `[1,2,…,x]` からの, `x` が配列ならば `x` からの,
インデックスに重複がない `t` 個の組み合わせのすべてを生成する生成子を返す.
"""
mycombinations(x, t) = mycombinations!(x, t, Vector{typeof(t)}(undef, t))
```

Out[2]: mycombinations

Loading web-font STIX-Web/Size1/Regular

1.3 Welchのt検定

In [3]:

```
function tvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ=0)
    (x̄ - ȳ - Δμ) / √(sx²/m + sy²/n)
end

function tvalue_welch(x, y; Δμ=0)
    m, x̄, sx² = length(x), mean(x), var(x)
    n, ȳ, sy² = length(y), mean(y), var(y)
    tvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ)
end

function degree_of_freedom_welch(m, sx², n, sy²)
    (sx²/m + sy²/n)^2 / ((sx²/m)^2/(m-1) + (sy²/n)^2/(n-1))
end

function degree_of_freedom_welch(x, y)
    m, sx² = length(x), var(x)
    n, sy² = length(y), var(y)
    degree_of_freedom_welch(m, sx², n, sy²)
end

function pvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ=0)
    t = tvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ)
    ν = degree_of_freedom_welch(m, sx², n, sy²)
    2ccdf(TDist(ν), abs(t))
end

function pvalue_welch(x, y; Δμ=0)
    m, x̄, sx² = length(x), mean(x), var(x)
    n, ȳ, sy² = length(y), mean(y), var(y)
    pvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ)
end

function confint_welch(m, x̄, sx², n, ȳ, sy²; α=0.05)
    ν = degree_of_freedom_welch(m, sx², n, sy²)
    c = quantile(TDist(ν), 1-α/2)
    SEhat = √(sx²/m + sy²/n)
    [x̄-ȳ-c*SEhat, x̄-ȳ+c*SEhat]
end

function confint_welch(x, y; α=0.05)
    m, x̄, sx² = length(x), mean(x), var(x)
    n, ȳ, sy² = length(y), mean(y), var(y)
    confint_welch(m, x̄, sx², n, ȳ, sy²; α)
end
```

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

1.4 単峰型の函数が正の値になる場所を見つける函数

In [4]:

```
function findpositive(f, a, b; maxsplit = 30)
    @assert f(a) < 0
    @assert f(b) < 0
    c = (a + b)/2
    f(c) > 0 && return c
    w = b - a
    for k in 2:maxsplit
        for d in range(w/2^(k+1), w/2-w/2^(k+1), step=w/2^k)
            x = c + d
            f(x) > 0 && return x
            x = c - d
            f(x) > 0 && return x
        end
    end
    error("k > maxplit = $maxsplit")
end

f(x) = abs(x) < 1e-4 ? 1.0 : -1.0

@time findpositive(f, -100abs(randn()), 20abs(randn()))
```

```
  0.000373 seconds
```

Out[4]: -5.2804658903227164e-5

Loading web-font STIX-Web/Size1/Regular

1.5 2つの分布が「互角」になるシフトの仕方を求める函数

In [5]:

```
"""
    prob_x_le_y(distx::UnivariateDistribution, disty::UnivariateDistribution;
        a = 0.0)

この函数は，連続分布 `distx`, `disty` と実数 `a` について，
`distx` と `disty` に従って生成される乱数をそれぞれ X, Y と書くとき，
X ≤ Y + a が成立する確率を返す．
"""
function prob_x_le_y(distx::UnivariateDistribution, disty::UnivariateDistribution,
        a = 0.0)
    H(y) = cdf(distx, y) * pdf(disty, y-a)
    quadgk(H, extrema(disty + a)...)[1]
end

"""
    tieshift(distx::UnivariateDistribution, disty::UnivariateDistribution;
        p = 0.5)

この函数は，連続分布 `distx`, `disty` と実数 `p` について，
`distx` と `disty` に従って生成される乱数をそれぞれ X, Y と書くとき，
X ≤ Y + a が成立する確率が `p` に等しくなるような実数 a を返す．
"""
function tieshift(distx::UnivariateDistribution, disty::UnivariateDistribution;
        p=0.5)
    find_zero(a -> prob_x_le_y(distx, disty, a) - p, 0.0)
end

@show tieshift(Normal(0, 1), Normal(2, 2))
@show tieshift(Normal(0, 1), Laplace(2, 2))
@show tieshift(Normal(0, 1), Uniform(0, 1));
```

```
tieshift(Normal(0, 1), Normal(2, 2)) = -1.999999999999923
tieshift(Normal(0, 1), Laplace(2, 2)) = -1.99999999999945
tieshift(Normal(0, 1), Uniform(0, 1)) = -0.4999999999999999
```

Loading web-font STIX-Web/Size1/Regular

2 Brunner-Munzel検定

2.1 Brunner-Munzel検定の実装

Loading web-font STIX-Web/Size1/Regular

In [6]:

```
"""
    h_brunner_munzel(x, y)

この函数は, x < y のとき 1.0 を, x = y のとき 0.5 を, それら以外のとき 0.0 返す.
"""
h_brunner_munzel(x, y) = (x < y) + (x == y)/2

@doc raw"""
    phat_brunner_munzel(X, Y)

まず以下のようにおく:

```math
\begin{aligned}
&
H(x, y) = \begin{cases} 1 & (x < y) \\ 1/2 & (x = y) \\ 0 & (x > y), \end{cases}
\\ &
m = \mathrm{length}(X), \quad
n = \mathrm{length}(Y), \quad
x_i = X[i], \quad
y_j = Y[j]
\end{aligned}
```

この函数は次の $\hat{p}$ を返す:

```math
\hat{p} = \frac{1}{mn}\sum_{i=1}^m \sum_{j=1}^n H(x_i, y_j).
```
"""
phat_brunner_munzel(X, Y) = mean(h_brunner_munzel(x, y) for x in X, y in Y)

@doc raw"""
    statistics_brunner_munzel(X, Y,
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64);
        p = 1/2
)

この函数はデータ `X`, `Y` について, Brunner-Munzel検定関係の統計量達を計算する. 詳細は以下

函数 $H(x, y)$ と $\hat{p}$, H^x_i, H^y_j, $\bar{H}^x$, $\bar{H}^y$ を次のように定め

```math
\begin{aligned}
&
m = \mathrm{length}(X), \quad
n = \mathrm{length}(Y), \quad
x_i = X[i], \quad
y_j = Y[j],
\\ &
\hat{p} = \frac{1}{mn}\sum_{i=1}^m \sum_{j=1}^n H(x_i, y_j),
\\ &
H(x, y) = \begin{cases} 1 & (x < y) \\ 1/2 & (x = y) \\ 0 & (x > y), \end{cases}
\\ &
H^x_i = \sum_{j=1}^n H(y_j, x_i), \quad
H^y_j = \sum_{i=1}^m H(x_i, y_j),
\\ &
\bar{H}^x = \frac{1}{m} \sum_{i=1}^m H^x_i = n - n\hat{p},
\\ &
\bar{H}^y = \frac{1}{n} \sum_{j=1}^n H^y_j = m\hat{p}.
\end{aligned}
```

この函数は以下達の named tuple で返す:

```math
\begin{aligned}
&
\mathrm{phat} =
\hat{p} = \frac{\bar{H}^x - \bar{H}^y + n}{m + n},
\\ &
\mathrm{sx2} =
\hat{\sigma}_x^2 = \frac{1}{n^2}\frac{1}{m-1}\sum_{i=1}^m (H^x_i - \bar{H}^x)^2,
\\ &
\mathrm{sy2} =
\hat{\sigma}_y^2 = \frac{1}{m^2}\frac{1}{n-1}\sum_{j=1}^n (H^y_j - \bar{H}^y)^2,
\\ &
\mathrm{sehat} =
\widehat{\mathrm{se}} = \sqrt{\frac{\hat{\sigma}_x^2}{m} + \frac{\hat{\sigma}_y^2}{n}},
```

Loading web-font STIX-Web/Size1/Regular
```

```
\\ &
\mathrm{tvalue} = t = \frac{\hat{p} - p}{\widehat{\mathrm{se}}},
\\ &
\mathrm{df} =
\nu =
\frac
{\left(\hat{\sigma}_x^2/m + \hat{\sigma}_y^2/n\right)^2}
{
\dfrac{\left(\hat{\sigma}_x^2/m\right)^2}{m-1} +
\dfrac{\left(\hat{\sigma}_y^2/n\right)^2}{n-1}
},
\\ &
\mathrm{pvalue} =
2\mathrm{ccdf}(\mathrm{TDist}(\nu), |t|).
\end{aligned}
```
"""
function statistics_brunner_munzel(X, Y,
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64);
        p = 1/2
)
    m, n = length(X), length(Y)
    for (i, x) in pairs(X)
        Hx[i] = sum(h_brunner_munzel(y, x) for y in Y)
    end
    for (j, y) in pairs(Y)
        Hy[j] = sum(h_brunner_munzel(x, y) for x in X)
    end
    phat = (mean(Hy) - mean(Hx) + n)/(m + n)
    sx2, sy2 = var(Hx)/n^2, var(Hy)/m^2
    sehat = √(sx2/m + sy2/n)
    tvalue = (phat - p)/sehat
    df = safediv((sx2/m + sy2/n)^2, (sx2/m)^2/(m-1) + (sy2/n)^2/(n-1))
    pvalue = (df != 0 && isfinite(df)) ? 2ccdf(TDist(df), abs(tvalue)) : zero(df)
    (; phat, sx2, sy2, sehat, tvalue, df, pvalue)
end

@doc raw"""
    pvalue_brunner_munzel(X, Y,
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64);
        p = 1/2
)

この函数はBrunner-Munzel検定のP値 `pvalue` を返す.
"""
function pvalue_brunner_munzel(X, Y,
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64);
        p = 1/2
)
    statistics_brunner_munzel(X, Y, Hx, Hy; p).pvalue
end

"""
    tieshift(X::AbstractVector, Y::AbstractVector; p = 1/2)

この函数は `phat_brunner_munzel(X, Y .+ a)` の値が `p` に等しくなる `a` を返す.
"""
function tieshift(X::AbstractVector, Y::AbstractVector; p = 1/2)
    shiftmin = minimum(X) - maximum(Y) - 0.1
    shiftmax = maximum(X) - minimum(Y) + 0.1
    find_zero(a -> phat_brunner_munzel(X, Y .+ a) - p, (shiftmin, shiftmax))
end

@doc raw"""
    brunner_munzel(X, Y,
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64),
        Ytmp = similar(Y, Float64);
        p = 1/2,
        α = 0.05,
        maxsplit = 30
)

この函数はBrunner-Munzel検定を実行する. 詳細は以下の通り.

この函数は `phat`, `sehat`, `tvalue`, `df`, `p`, `pvalue`, `α` および\
以下達の named tuple を返す.
```

Loading web-font STIX-Web/Size1/Regular
```

```
```math
\begin{aligned}
&
\mathrm{confint_p} = (\text{$p$ の信頼度 $1-\alpha$ の信頼区間}),
\\ &
\mathrm{confint_shift} = (\text{2つの集団が互角になるようなシフトの信頼度 $1-\alpha$ の
\\ &
\mathrm{pvalue_shift} = ($\mathrm{confint\_shift}$ の計算で使われたP値函数),
\\ &
\mathrm{shifthat} = (\text{2つの集団が互角になるようなシフトの点推定値}).
\end{aligned}
```

さらに, $\mathrm{shiftmin}$, $\mathrm{shiftmax}$ はデータから推定されるシフトの下限と上限

"""
function brunner_munzel(X, Y,
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64),
        Ytmp = similar(Y, Float64);
        p = 1/2,
        α = 0.05,
        maxsplit = 30
)
    (; phat, sehat, tvalue, df, pvalue) = statistics_brunner_munzel(X, Y, Hx, Hy; p)

    c = df == 0 ? Inf : quantile(TDist(df), 1 - α/2)
    confint_p = [max(0, phat - c*sehat), min(1, phat + c*sehat)]

    function pvalue_shift(a)
        @. Ytmp = Y + a
        pvalue_brunner_munzel(X, Ytmp, Hx, Hy; p)
    end
    shiftmin = minimum(X) - maximum(Y) - 0.1
    shiftmax = maximum(X) - minimum(Y) + 0.1
    shifthat = tieshift(X, Y; p)
    confint_shift = [
        find_zero(a -> pvalue_shift(a) - α, (shiftmin, shifthat))
        find_zero(a -> pvalue_shift(a) - α, (shifthat, shiftmax))
    ]

    (; phat, sehat, tvalue, df, p, pvalue, α, confint_p,
        confint_shift, pvalue_shift, shifthat, shiftmin, shiftmax)
end

function show_plot_brunner_munzel(X, Y,
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64),
        Ytmp = similar(Y, Float64);
        p = 1/2,
        α = 0.05,
        showXY = false,
        kwargs...
)
    showXY && (@show X Y)
    (; phat, sehat, tvalue, df, p, pvalue, α, confint_p,
        confint_shift, pvalue_shift, shifthat, shiftmin, shiftmax) =
        brunner_munzel(X, Y, Hx, Hy, Ytmp; p, α)
    pprint((; phat, sehat, tvalue, df, p, pvalue, α, confint_p,
            confint_shift, shifthat))
    println()
    @show median(X) median(Y)
    plot(pvalue_shift, shiftmin, shiftmax; label="")
    vline!([tieshift(X, Y)]; label="", ls=:dash)
    title!("P-value function of shift")
    plot!(ytick=0:0.05:1)
    plot!(; kwargs...)
end
```

Out[6]: show_plot_brunner_munzel (generic function with 4 methods)

In [7]:
```
@doc h_brunner_munzel
```

Out[7]:
```
\begin{verbatim}
h_brunner_munzel(x, y)
\end{verbatim}
```
この函数は, x < y のとき 1.0 を, x = y のとき 0.5 を, それら以外のとき 0.0 返す.

Loading web-font STIX-Web/Size1/Regular

In [8]:
```
1 @doc phat_brunner_munzel
```

Out[8]:

\begin{verbatim}
phat_brunner_munzel(X, Y)
\end{verbatim}

まず以下のようにおく:

$$H(x,y) = \begin{cases} 1 & (x < y) \\ 1/2 & (x = y) \\ 0 & (x > y), \end{cases}$$

$$m = \mathrm{length}(X), \quad n = \mathrm{length}(Y), \quad x_i = X[i], \quad y_j = Y[j]$$

この函数は次の $\hat{p}$ を返す:

$$\hat{p} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} H(x_i, y_j).$$

In [9]:
```
1 @doc statistics_brunner_munzel
```

Out[9]: \begin{verbatim} statistics_brunner_munzel(X, Y, Hx = similar(X, Float64), Hy = similar(Y, Float64); p = 1/2) \end{verbatim} この函数はデータ \texttt{X}, \texttt{Y} について, Brunner-Munzel検定関係の統計量達を計算する. 詳細は以下の通り. 函数 $H(x, y)$ と $\hat{p}$, H^x_i, H^y_j, $\bar{H}^x$, $\bar{H}^y$ を次のように定める: $$\begin{aligned} & m = \mathrm{length}(X), \quad n = \mathrm{length}(Y), \quad x_i = X[i], \quad y_j = Y[j], \\ & \hat{p} = \frac{1}{mn}\sum_{i=1}^m \sum_{j=1}^n H(x_i, y_j), \\ & H(x, y) = \begin{cases} 1 & (x < y) \\ 1/2 & (x = y) \\ 0 & (x > y), \end{cases} \\ & H^x_i = \sum_{j=1}^n H(y_j, x_i), \quad H^y_j = \sum_{i=1}^m H(x_i, y_j), \\ & \bar{H}^x = \frac{1}{m} \sum_{i=1}^m H^x_i = n - n\hat{p}, \\ & \bar{H}^y = \frac{1}{n} \sum_{j=1}^n H^y_j = m\hat{p}. \end{aligned}$$ この函数は以下達の named tuple で返す: $$\begin{aligned} & \mathrm{phat} = \hat{p} = \frac{\bar{H}^x - \bar{H}^y + n}{m + n}, \\ & \mathrm{sx2} = \hat{\sigma}_x^2 = \frac{1}{n^2}\frac{1}{m-1}\sum_{i=1}^m (H^x_i - \bar{H}^x)^2, \\ & \mathrm{sy2} = \hat{\sigma}_y^2 = \frac{1}{m^2}\frac{1}{n-1}\sum_{j=1}^n (H^y_j - \bar{H}^y)^2, \\ & \mathrm{sehat} = \widehat{\mathrm{se}} = \sqrt{\frac{\hat{\sigma}_x^2}{m} + \frac{\hat{\sigma}_y^2}{n}}, \\ & \mathrm{tvalue} = t = \frac{\hat{p} - p}{\widehat{\mathrm{se}}}, \\ & \mathrm{df} = \nu = \frac {\left(\hat{\sigma}_x^2/m + \hat{\sigma}_y^2/n\right)^2} { \dfrac{\left(\hat{\sigma}_x^2/m\right)^2}{m-1} + \dfrac{\left(\hat{\sigma}_y^2/n\right)^2}{n-1} }, \\ & \mathrm{pvalue} = 2\mathrm{ccdf}(\mathrm{TDist}(\nu), |t|). \end{aligned}$$

In [10]:
```
1 @doc brunner_munzel
```

Out[10]:
```
brunner_munzel(X, Y,
    Hx = similar(X, Float64),
    Hy = similar(Y, Float64),
    Ytmp = similar(Y, Float64);
    p = 1/2,
    α = 0.05,
    maxsplit = 30
)

この函数はBrunner-Munzel検定を実行する. 詳細は以下の通り.

この函数は phat, sehat, tvalue, df, p, pvalue, α および
以下達の named tuple を返す.

\begin{aligned}
&
\mathrm{confint\_p} = (\text{$p$ の信頼度 $1-\alpha$ の信頼区間}),
\\ &
\mathrm{confint\_shift} = (\text{2つの集団が互角になるようなシフトの信頼度 $1-\alpha$ の信頼区間}),
\\ &
\mathrm{pvalue\_shift} = ($\mathrm{confint\_shift}$ の計算で使われたP値函数),
\\ &
\mathrm{shifthat} = (\text{2つの集団が互角になるようなシフトの点推定値}).
\end{aligned}

さらに, \mathrm{shiftmin}, \mathrm{shiftmax}
はデータから推定されるシフトの下限と上限.
```

Loading web-font STIX-Web/Size1/Regular

In [11]:

```
X = randn(10)
Y = randn(10)
@show shiftmin = minimum(X) - maximum(Y) - 1
@show shiftmax = maximum(X) - minimum(Y) + 1
pvalue_brunner_munzel(X, Y)
```

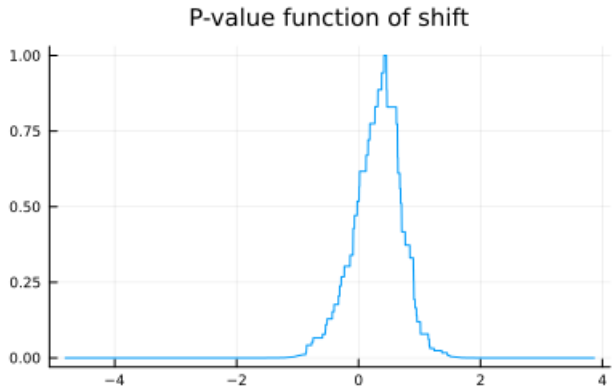
```
shiftmin = (minimum(X) - maximum(Y)) - 1 = -4.810896555167297
shiftmax = (maximum(X) - minimum(Y)) + 1 = 3.8662791521107494
```

Out[11]: 0.5181701995353223

In [12]:

```
plot(a -> pvalue_brunner_munzel(X, Y .+ a), shiftmin, shiftmax;
    label="", title="P-value function of shift")
```

Out[12]:



2.2 よく使われているっぽいテストデータで正しく実装されているかを確認

https://okumuralab.org/~okumura/stat/brunner-munzel.html (https://okumuralab.org/~okumura/stat/brunner-munzel.html)

```
x = c(1,2,1,1,1,1,1,1,1,1,2,4,1,1)
y = c(3,3,4,3,1,2,3,1,1,5,4)
brunnermunzel.test(x, y)

data:  x and y
Brunner-Munzel Test Statistic = 3.1375, df = 17.683, p-value = 0.005786
95 percent confidence interval:
 0.5952169 0.9827052
sample estimates:
P(X<Y)+.5*P(X=Y)
        0.788961
```

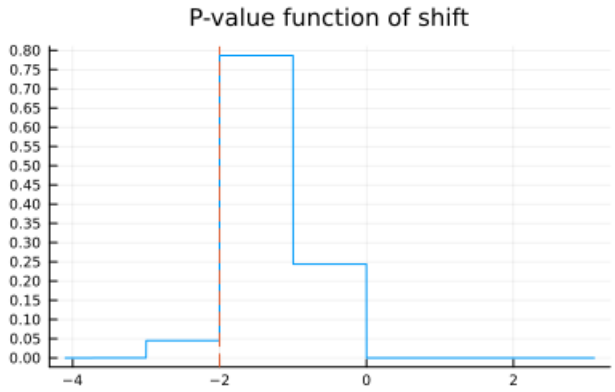
Loading web-font STIX-Web/Size1/Regular

In [13]:

```
X = [1,2,1,1,1,1,1,1,1,1,2,4,1,1]
Y = [3,3,4,3,1,2,3,1,1,5,4]
show_plot_brunner_munzel(X, Y)
```

```
(phat = 0.788961038961039,
 sehat = 0.09210009046816862,
 tvalue = 3.1374674823029505,
 df = 17.682841979481545,
 p = 0.5,
 pvalue = 0.005786208666151463,
 α = 0.05,
 confint_p = [0.5952168642537363, 0.9827052136683416],
 confint_shift = [-2.0000000000000004, -5.551115123125783e-17],
 shifthat = -1.9999999999999998)
median(X) = 1.0
median(Y) = 3.0
```

Out[13]:



In [14]:

```
X = [1,2,1,1,1,1,1,1,1,1,2,4,1,1]
Y = [3,3,4,3,1,2,3,1,1,5,4]
@rput X Y
R"""
library(lawstat)
brunner.munzel.test(X, Y)
"""
```

```
Warning: RCall.jl: Warning: package 'lawstat' was built under R version 4.4.1
@ RCall D:\.julia\packages\RCall\dDAVd\src\io.jl:172
```

Out[14]:

```
RObject{VecSxp}

        Brunner-Munzel Test

data:  X and Y
Brunner-Munzel Test Statistic = 3.1375, df = 17.683, p-value = 0.005786
95 percent confidence interval:
 0.5952169 0.9827052
sample estimates:
P(X<Y)+.5*P(X=Y)
        0.788961
```

このように Brunner-Munzel 検定は R では [lawstat (https://cran.r-project.org/package=lawstat)](https://cran.r-project.org/package=lawstat) パッケージの [brunner.munzel.test (https://rdrr.io/cran/lawstat/man/brunner.munzel.test.html)](https://rdrr.io/cran/lawstat/man/brunner.munzel.test.html) で使える.

Loading web-font STIX-Web/Size1/Regular

2.3 組み合わせの生成子

In [15]:
```
"""
    complementcomb!(complcomb::AbstractVector, comb::AbstractVector)

'comb' が {1,2,…,N} から重複無しに m 個を選ぶ組み合わせを表す配列であり,
'comb' の中で数は小さな順に並んでいるとし, 'complcomb' は長さ N - m の配列であると仮定する

このとき, この函数は配列 'complcomb' に配列 'comb' の補集合を格納し, 'complcomb' を返す.

この函数はメモリ割り当てゼロで実行される.
"""
function complementcomb!(complcomb::AbstractVector, comb::AbstractVector)
    N = length(comb) + length(complcomb)
    k = 0
    a = 0
    @inbounds for b in comb
        for i in a+1:b-1
            k += 1
            complcomb[k] = i
        end
        a = b
    end
    @inbounds for i in a+1:N
        k +=1
        complcomb[k] = i
    end
    complcomb
end

"""
    complementcomb(N, comb::AbstractVector)

'comb' が {1,2,…,N} から重複無しに m 個を選ぶ組み合わせを表す配列であり,
'comb' の中で数は小さな順に並んでいると仮定する.

この函数は 'comb' の補集合の配列を返す.

この函数は返り値の配列の分だけのメモリ割り当てを行う.
"""
complementcomb(N, comb::AbstractVector) =
    complementcomb!(similar(comb, N - length(comb)), comb)
```

Out[15]:
```
complementcomb
```

In [16]:
```
@doc complementcomb!
```

Out[16]:
\begin{verbatim} complementcomb!(complcomb::AbstractVector, comb::AbstractVector) \end{verbatim} \texttt{comb} が \{1,2,…,N\} から重複無しに m 個を選ぶ組み合わせを表す配列であり, \texttt{comb} の中で数は小さな順に並んでいるとし, \texttt{complcomb} は長さ N - m の配列であると仮定する. このとき, この函数は配列 \texttt{complcomb} に配列 \texttt{comb} の補集合を格納し, \texttt{complcomb} を返す. この函数はメモリ割り当てゼロで実行される.

In [17]:
```
@doc complementcomb
```

Out[17]:
\begin{verbatim} complementcomb(N, comb::AbstractVector) \end{verbatim} \texttt{comb} が \{1,2,…,N\} から重複無しに m 個を選ぶ組み合わせを表す配列であり, \texttt{comb} の中で数は小さな順に並んでいると仮定する. この函数は \texttt{comb} の補集合の配列を返す. この函数は返り値の配列の分だけのメモリ割り当てを行う.

In [18]:
```
N = 10
comb = [2, 4, 5, 8]
ccomb = similar(comb, N - length(comb))
@btime complementcomb!($ccomb, $comb);
```

```
  14.629 ns (0 allocations: 0 bytes)
```

Loading web-font STIX-Web/Size1/Regular

In [19]:

```
N, m = 5, 3
ccomb = Vector{Int}(undef, N-m)
[(copy(comb), copy(complementcomb!(ccomb, comb))) for comb in mycombinations(1:N, m)]
```

Out[19]:

```
10-element Vector{Tuple{Vector{Int64}, Vector{Int64}}}:
 ([1, 2, 3], [4, 5])
 ([1, 2, 4], [3, 5])
 ([1, 2, 5], [3, 4])
 ([1, 3, 4], [2, 5])
 ([1, 3, 5], [2, 4])
 ([1, 4, 5], [2, 3])
 ([2, 3, 4], [1, 5])
 ([2, 3, 5], [1, 4])
 ([2, 4, 5], [1, 3])
 ([3, 4, 5], [1, 2])
```

In [20]:

```
N, m = 5, 3
ccomb = Vector{Int}(undef, N-m)
[(copy(comb), complementcomb(N, comb)) for comb in mycombinations(1:N, m)]
```

Out[20]:

```
10-element Vector{Tuple{Vector{Int64}, Vector{Int64}}}:
 ([1, 2, 3], [4, 5])
 ([1, 2, 4], [3, 5])
 ([1, 2, 5], [3, 4])
 ([1, 3, 4], [2, 5])
 ([1, 3, 5], [2, 4])
 ([1, 4, 5], [2, 3])
 ([2, 3, 4], [1, 5])
 ([2, 3, 5], [1, 4])
 ([2, 4, 5], [1, 3])
 ([3, 4, 5], [1, 2])
```

Loading web-font STIX-Web/Size1/Regular

2.4 Brunner-Munzel検定のpermutation版の実装

In [21]:
```
"""
    permutation_tvalues_brunner_munzel(X, Y,
        XandY = Vector{Float64}(undef, length(X)+length(Y)),
        Tval = Vector{Float64}(undef, binomial(length(X)+length(Y), length(X))),
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64)
    )

Brunner-Munzel検定のt値を `[X; Y]` から\
インデックスの重複無しに `length(X)` 個取る組み合わせと\
その補集合への分割のすべてについて計算して, `Tval` に格納して返す.
"""
function permutation_tvalues_brunner_munzel(X, Y,
        XandY = Vector{Float64}(undef, length(X)+length(Y)),
        Tval = Vector{Float64}(undef, binomial(length(X)+length(Y), length(X))),
        Hx = similar(X, Float64),
        Hy = similar(Y, Float64),
        ccomb = Vector{Int}(undef, length(Y))
    )
    m, n = length(X), length(Y)
    N = m + n
    @views XandY[1:m] .= X
    @views XandY[m+1:N] .= Y
    for (k, comb) in enumerate(mycombinations(1:N, m))
        complementcomb!(ccomb, comb)
        Tval[k] = statistics_brunner_munzel(
            view(XandY, comb), view(XandY, ccomb), Hx, Hy).tvalue
    end
    Tval
end

"""
    pvalue_brunner_munzel_perm(X, Y,
        Tval = permutation_tvalues_brunner_munzel(X, Y),
        tval = statistics_brunner_munzel(X, Y).tvalue;
        le = ≲
    )

Brunner-Munzel検定のpermutation版のP値を返す.
"""
function pvalue_brunner_munzel_perm(X, Y,
        Tval = permutation_tvalues_brunner_munzel(X, Y),
        tval = statistics_brunner_munzel(X, Y).tvalue;
        le = ≲
    )
    pvalue_perm = mean(T -> le(abs(tval), abs(T)), Tval)
end
```

Out[21]: pvalue_brunner_munzel_perm

In [22]:
```
@doc permutation_tvalues_brunner_munzel
```

Out[22]: \begin{verbatim} permutation_tvalues_brunner_munzel(X, Y, XandY = Vector{Float64}(undef, length(X)+length(Y)), Tval = Vector{Float64}(undef, binomial(length(X)+length(Y), length(X))), Hx = similar(X, Float64), Hy = similar(Y, Float64)) \end{verbatim} Brunner-Munzel検定のt値を \texttt{[X; Y]} からインデックスの重複無しに \texttt{length(X)} 個取る組み合わせとその補集合への分割のすべてについて計算して, \texttt{Tval} に格納して返す.

In [23]:
```
@doc pvalue_brunner_munzel_perm
```

Out[23]: \begin{verbatim} pvalue_brunner_munzel_perm(X, Y, Tval = permutation_tvalues_brunner_munzel(X, Y), tval = statistics_brunner_munzel(X, Y).tvalue; le = ≲) \end{verbatim} Brunner-Munzel検定のpermutation版のP値を返す.

https://okumuralab.org/~okumura/stat/brunner-munzel.html (https://okumuralab.org/~okumura/stat/brunner-munzel.html)

Loading web-font STIX-Web/Size1/Regular

```
bm = brunner.munzel.test(x, y)$statistic
n1 = length(x)
n2 = length(y)
N = n1 + n2
xandy = c(x, y)
foo = function(X) {
```

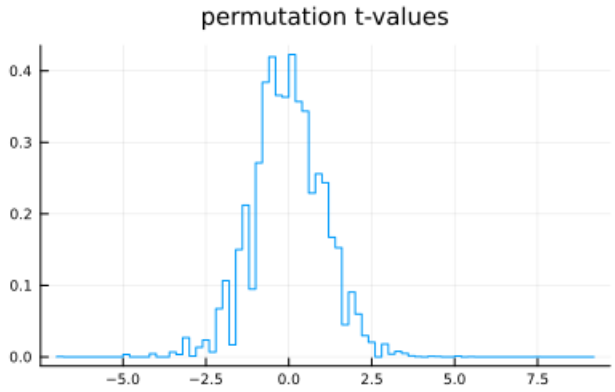
In [24]:

```
X = [1,2,1,1,1,1,1,1,1,1,2,4,1,1]
Y = [3,3,4,3,1,2,3,1,1,5,4]
@show X Y
@show m, n = length(X), length(Y)

Tval = @time permutation_tvalues_brunner_munzel(X, Y)
@show pvalue_brunner_munzel_perm(X, Y, Tval)
stephist(Tval; norm=true, bin=101, label="", title="permutation t-values")
```

```
X = [1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 4, 1, 1]
Y = [3, 3, 4, 3, 1, 2, 3, 1, 1, 5, 4]
(m, n) = (length(X), length(Y)) = (14, 11)
  4.369441 seconds (2.91 M allocations: 798.552 MiB, 2.66% gc time, 6.21% compilation time)
pvalue_brunner_munzel_perm(X, Y, Tval) = 0.008037645264055279
```

Out[24]:



2.5 permutation版が正しく実装されているかの確認

- https://github.com/toshi-ara/brunnermunzel/issues/14 (https://github.com/toshi-ara/brunnermunzel/issues/14)
- https://github.com/toshi-ara/brunnermunzel/files/4395032/mwe.R.zip (https://github.com/toshi-ara/brunnermunzel/files/4395032/mwe.R.zip)

追記 2022-08-06: https://twitter.com/TA25140989/status/1555825941451923457 (https://twitter.com/TA25140989/status/1555825941451923457) を参照せよ.

https://github.com/toshi-ara/brunnermunzel/tree/development (https://github.com/toshi-ara/brunnermunzel/tree/development) の修正版の brunnermunzel パッケージをインストールし直した. 以下のセルの実行結果が変わるはずなので, その記録を残しておく.

以下の記録を見なくても,

- https://github.com/genkuroki/public/blob/e4faafc52721b63876b3b705f9450eade3c902f5/0034/Brunner-Munzel.ipynb (https://github.com/genkuroki/public/blob/e4faafc52721b63876b3b705f9450eade3c902f5/0034/Brunner-Munzel.ipynb)

で閲覧できるが, わざわざ見に行くのも面倒なのでこのファイルにも記録を残しておく.

以前の実行結果:

```
@show pval_J - pval_J_le;
```

Loading web-font STIX-Web/Size1/Regular

```
pval_J - pval_J_le = [0.0, 0.0, 0.007936507936507936, 0.023809523809523808, 0.02380952380
9523808, 0.0793650793650793, 0.0, 0.0, 0.007936507936507936, 0.0357142857142857, 0.0, 0.0
15873015873015928, 0.015873015873015872, 0.0, 0.015873015873015872, 0.0, 0.0, 0.0, 0.0079
36507936507936, 0.0, 0.023809523809523836, 0.015873015873015928, 0.011904761904761918, 0.
0, 0.015873015873015928, 0.0, 0.003968253968253968, 0.0, 0.0, 0.0, 0.0, 0.007936507936507
936, 0.0, 0.015873015873015928, 0.0, 0.015873015873015928, 0.0, 0.0, 0.0, 0.0079365079365
07908, 0.0, 0.007936507936507908, 0.0, 0.007936507936507908, 0.0, 0.0, 0.0079365079365079
36, 0.007936507936507936, 0.011904761904761918, 0.007936507936507936, 0.0396825396825396
5, 0.0, 0.007936507936507936, 0.0, 0.05555555555555547, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.015873015873015872, 0.015873015873015872, 0.0, 0.0158730158
73015928, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.015873015873015928, 0.007936507936507964, 0.007
936507936507964, 0.0, 0.003968253968253954, 0.0, 0.0, 0.0, 0.015873015873015872, 0.0, 0.
0, 0.0357142857142857, 0.015873015873015872, 0.0, 0.023809523809523725, 0.0, 0.0238095238
09523808, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

idx = @show findall(pval_J .!= pval_J_le)
length(idx)

findall(pval_J .!= pval_J_le) = [3, 4, 5, 6, 9, 10, 12, 13, 15, 19, 21, 22, 23, 25, 27, 3
2, 34, 36, 40, 42, 44, 47, 48, 49, 50, 51, 53, 55, 68, 69, 71, 78, 79, 80, 82, 86, 89, 9
0, 92, 94]

40

@show pval_R - pval_J_le;

pval_R - pval_J_le = [0.0, 0.0, 0.007936507936507936, 0.023809523809523808, 0.02380952380
9523808, 0.0793650793650793, 0.0, 0.0, 0.007936507936507936, 0.0357142857142857, 0.0, 0.0
15873015873015928, 0.015873015873015872, 0.0, 0.015873015873015872, 0.0, 0.0, 0.0, 0.0079
36507936507936, 0.0, 0.023809523809523836, 0.007936507936507908, 0.011904761904761918, 0.
0, 0.015873015873015928, 0.0, 0.003968253968253968, 0.0, 0.0, 0.0, 0.0, 0.007936507936507
936, 0.0, 0.00793650793650802, 0.0, 0.015873015873015928, 0.0, 0.0, 0.0, 0.00793650793650
7908, 0.0, 0.007936507936507908, 0.0, 0.007936507936507908, 0.0, 0.0, 0.0079365079365079
36, 0.007936507936507936, 0.011904761904761918, 0.007936507936507936, 0.03968253968253965,
0.0, 0.007936507936507936, 0.0, 0.05555555555555547, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.015873015873015872, 0.015873015873015872, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.007936507936507908, 0.007936507936507964, 0.007936507936507964, 0.
0, 0.003968253968253954, 0.0, 0.0, 0.0, 0.015873015873015872, 0.0, 0.0, 0.035714285714285
7, 0.015873015873015872, 0.0, 0.023809523809523725, 0.0, 0.023809523809523808, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0]

idx = @show findall(pval_R .!= pval_J_le)
length(idx)

findall(pval_R .!= pval_J_le) = [3, 4, 5, 6, 9, 10, 12, 13, 15, 19, 21, 22, 23, 25, 27, 3
2, 34, 36, 40, 42, 44, 47, 48, 49, 50, 51, 53, 55, 68, 69, 78, 79, 80, 82, 86, 89, 90, 9
2, 94]

39

@show pval_J - pval_R;

pval_J - pval_R = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.00793650793650802, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.007936507936507908, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.015873015873015928, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.00793650793650802, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

idx = @show findall(.!(pval_J .≈ pval_R))
length(idx)

findall(.!(pval_J .≈ pval_R)) = [22, 34, 71, 78]

4

all(pval_J .≥ pval_R .≥ pval_J_le)

true
```

Loading web-font STIX-Web/Size1/Regular

以前のコメントは以下の通り:

なるほど！

https://github.com/toshi-ara/brunnermunzel/issues/14

に書いてあるように，22, 34, 71 and 78 の4つで，値が一致していない．

〇〇以下または〇〇以上の判定を $x \approx y$ のときも true にする必要があるのだが，その部分で違いが生じているものと思われる．

現時点では https://CRAN.R-project.org/package=brunnermunzel にアクセスすると，

>Package ‘brunnermunzel’ was removed from the CRAN repository.
>
>Formerly available versions can be obtained from the [archive](https://cran.r-project.or

In [25]:

```
R"""
library(brunnermunzel)
set.seed(1290)
reps = 100
xx = c()
yy = c()
pval_R = numeric(reps)
for (i in seq_len(reps)){
  x = rnorm(5)
  y = rnorm(5)

  xx = c(xx, x)
  yy = c(yy, y)

  res_bm_perm <- brunnermunzel.permutation.test(x,y)
  pval_R[i] <- res_bm_perm$p.value
}
"""

@rget xx yy pval_R
XX = reshape(xx, 5, 100)
YY = reshape(yy, 5, 100)

pval_J = zeros(100)
pval_J_le = zeros(100)
for i in 1:100
    pval_J[i]    = pvalue_brunner_munzel_perm(XX[:,i], YY[:,i]; le = ≲)
    pval_J_le[i] = pvalue_brunner_munzel_perm(XX[:,i], YY[:,i]; le = ≤)
end
```

```
┌ Warning: RCall.jl: Warning: package 'brunnermunzel' was built under R version 4.4.1
└ @ RCall D:\.julia\packages\RCall\dDAVd\src\io.jl:172
```

In [26]:

```
@show pval_J - pval_J_le;
```

```
pval_J - pval_J_le = [0.0, 0.0, 0.007936507936507936, 0.023809523809523808, 0.02380952380952
3808, 0.07936507936507793, 0.0, 0.0, 0.007936507936507936, 0.03571428571428571, 0.0, 0.0158730
15873015928, 0.015873015873015872, 0.0, 0.015873015873015872, 0.0, 0.0, 0.0, 0.0079365079365
07936, 0.0, 0.023809523809523836, 0.015873015873015928, 0.011904761904761918, 0.0, 0.0158730
15873015928, 0.0, 0.003968253968253968, 0.0, 0.0, 0.0, 0.0, 0.007936507936507936, 0.0, 0.015
873015873015928, 0.0, 0.015873015873015928, 0.0, 0.0, 0.0, 0.007936507936507908, 0.0, 0.0079
36507936507908, 0.0, 0.007936507936507908, 0.0, 0.0, 0.0, 0.007936507936507936, 0.007936507936507
936, 0.011904761904761918, 0.007936507936507936, 0.03968253968253965, 0.0, 0.007936507936507
936, 0.0, 0.05555555555555547, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.015873015873015872, 0.015873015873015872, 0.0, 0.015873015873015928, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.015873015873015928, 0.007936507936507964, 0.007936507936507964, 0.0, 0.003968253
968253954, 0.0, 0.0, 0.0, 0.015873015873015872, 0.0, 0.0, 0.03571428571428571, 0.015873015873
015872, 0.0, 0.023809523809523725, 0.0, 0.023809523809523808, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

In [27]:

```
idx = @show findall(pval_J .!= pval_J_le)
length(idx)
```

```
findall(pval_J .!= pval_J_le) = [3, 4, 5, 6, 9, 10, 12, 13, 15, 19, 21, 22, 23, 25, 27, 32,
34, 36, 40, 42, 44, 47, 48, 49, 50, 51, 53, 55, 68, 69, 71, 78, 79, 80, 82, 86, 89, 90, 92,
94]
```

Out[27]: 40

Loading web-font STIX-Web/Size1/Regular

In [28]:
```
@show pval_R - pval_J_le;
```

```
pval_R - pval_J_le = [0.0, 0.0, 0.007936507936507936, 0.023809523809523808, 0.02380952380952
3808, 0.07936507936507936, 0.0, 0.0, 0.007936507936507936, 0.03571428571428571, 0.0, 0.0158730
15873015928, 0.015873015873015872, 0.0, 0.015873015873015872, 0.0, 0.0, 0.0, 0.0079365079365
07936, 0.0, 0.023809523809523836, 0.015873015873015928, 0.011904761904761918, 0.0, 0.0158730
15873015928, 0.0, 0.003968253968253968, 0.0, 0.0, 0.0, 0.0, 0.007936507936507936, 0.0, 0.015
873015873015928, 0.0, 0.015873015873015928, 0.0, 0.0, 0.0, 0.007936507936507908, 0.0, 0.0079
36507936507908, 0.0, 0.007936507936507908, 0.0, 0.0, 0.007936507936507936, 0.007936507936507
936, 0.011904761904761918, 0.007936507936507936, 0.03968253968253965, 0.0, 0.007936507936507
936, 0.0, 0.05555555555555547, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.015873015873015872, 0.015873015873015872, 0.0, 0.015873015873015928, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.015873015873015928, 0.007936507936507964, 0.007936507936507964, 0.0, 0.003968253
968253954, 0.0, 0.0, 0.0, 0.015873015873015872, 0.0, 0.0, 0.03571428571428571, 0.015873015873
015872, 0.0, 0.023809523809523725, 0.0, 0.023809523809523808, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

In [29]:
```
idx = @show findall(pval_R .!= pval_J_le)
length(idx)
```

```
findall(pval_R .!= pval_J_le) = [3, 4, 5, 6, 9, 10, 12, 13, 15, 19, 21, 22, 23, 25, 27, 32,
34, 36, 40, 42, 44, 47, 48, 49, 50, 51, 53, 55, 68, 69, 71, 78, 79, 80, 82, 86, 89, 90, 92,
94]
```

Out[29]: 40

In [30]:
```
@show pval_J - pval_R;
```

```
pval_J - pval_R = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

In [31]:
```
idx = @show findall(.!(pval_J .≈ pval_R))
length(idx)
```

```
findall(.!(pval_J .≈ pval_R)) = Int64[]
```

Out[31]: 0

In [32]:
```
all(pval_J .≥ pval_R .≥ pval_J_le)
```

Out[32]: true

2022-08-06: やった! 値が完全に一致した! permutation版Brunner-Munzel検定について,

- https://github.com/toshi-ara/brunnermunzel/tree/development (https://github.com/toshi-ara/brunnermunzel/tree/development)

の実装と私による実装の計算結果は以上の場合において完全に一致している.

3 計算例

In [33]:
```
m, n = 10, 10
X, Y = rand(Normal(0, 1), m), rand(Normal(0, 2), n)
@show pval_brmu = pvalue_brunner_munzel(X, Y)
@show pval_perm = pvalue_brunner_munzel_perm(X, Y);
```

```
pval_brmu = pvalue_brunner_munzel(X, Y) = 1.0
pval_perm = pvalue_brunner_munzel_perm(X, Y) = 1.0
```

Loading web-font STIX-Web/Size1/Regular

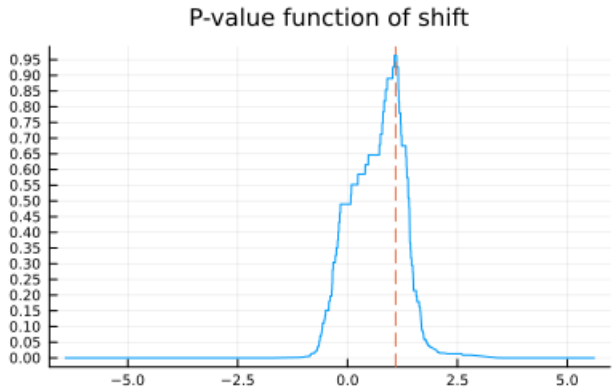
In [34]:

```
Random.seed!(4)

m, n = 10, 20
X, Y = rand(Normal(0, 1), m), rand(Normal(0, 2), n)
show_plot_brunner_munzel(X, Y)
```

```
(phat = 0.425,
 sehat = 0.10709480961098924,
 tvalue = -0.7003140513758763,
 df = 27.239689302515885,
 p = 0.5,
 pvalue = 0.4896699803996247,
 α = 0.05,
 confint_p = [0.20535003891236225, 0.6446499610876377],
 confint_shift = [-0.642106099966724, 1.7940553106227974],
 shifthat = 1.09533347422796)
median(X) = 0.4366097489932352
median(Y) = -0.8104755143666504
```

Out[34]:

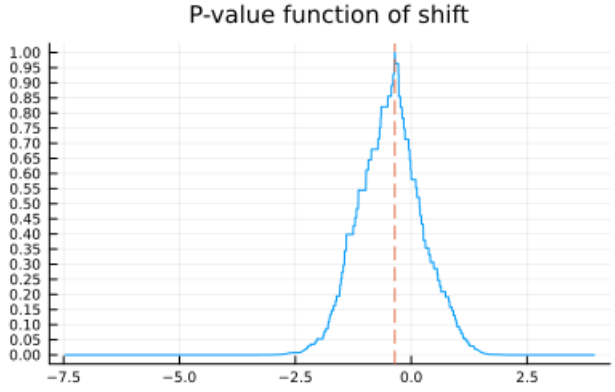


In [35]:

```
m, n = 10, 20
X, Y = rand(Normal(0, 1), m), rand(Normal(0, 2), n)
show_plot_brunner_munzel(X, Y)
```

```
(phat = 0.5549999999999999,
 sehat = 0.10651441985678231,
 tvalue = 0.5163620106456207,
 df = 27.01715719647146,
 p = 0.5,
 pvalue = 0.6098024812257896,
 α = 0.05,
 confint_p = [0.33645695680473703, 0.7735430431952628],
 confint_shift = [-2.032882441018063, 1.1693659401747338],
 shifthat = -0.3550706804781761)
median(X) = -0.17035935201789232
median(Y) = -0.16549665940751157
```

Out[35]:



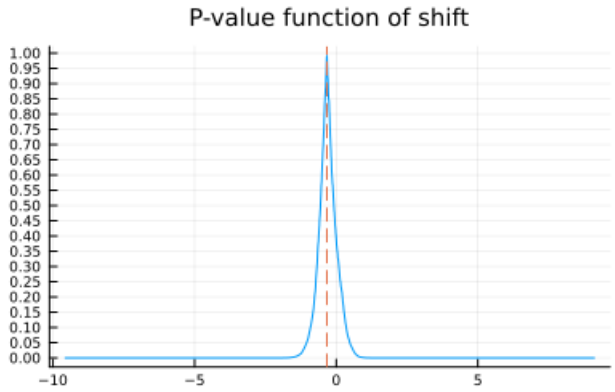
Loading web-font STIX-Web/Size1/Regular

In [36]:

```
m, n = 100, 50
X, Y = rand(Normal(0, 1), m), rand(Normal(0, 2), n)
show_plot_brunner_munzel(X, Y)
```

```
(phat = 0.5518,
 sehat = 0.059995235294905844,
 tvalue = 0.8634018975903285,
 df = 55.799136185276105,
 p = 0.5,
 pvalue = 0.39161265935808287,
 α = 0.05,
 confint_p = [0.431605553791199, 0.6719944462088009],
 confint_shift = [-1.0466768697894908, 0.480114930555916],
 shifthat = -0.32905087954292944)
median(X) = -0.044389033553791826
median(Y) = 0.35316051685566774
```

Out[36]:



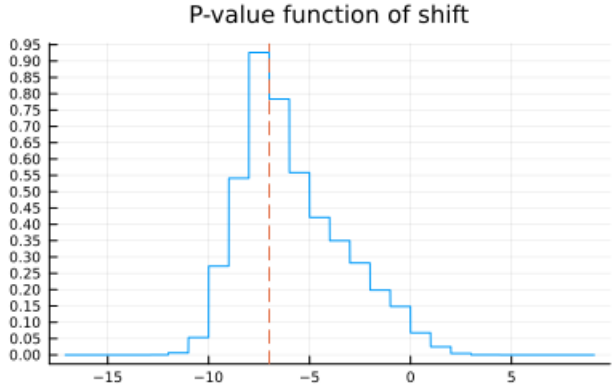
In [37]:

```
Random.seed!(4)

m, n = 10, 20
X, Y = rand(1:m, m), rand(1:n, n)
show_plot_brunner_munzel(X, Y)
```

```
(phat = 0.6725,
 sehat = 0.1005725567999867,
 tvalue = 1.715179622439735,
 df = 20.71835365424782,
 p = 0.5,
 pvalue = 0.1012319072053917,
 α = 0.05,
 confint_p = [0.4631746597119115, 0.8818253402880885],
 confint_shift = [-10.999999999999998, 0.9999999999999997],
 shifthat = -7.000000000000001)
median(X) = 6.5
median(Y) = 14.5
```

Out[37]:



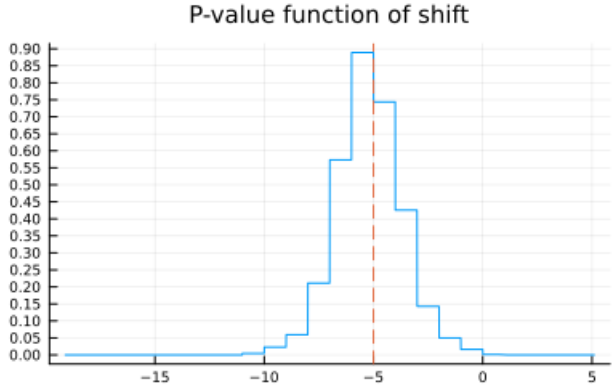
Loading web-font STIX-Web/Size1/Regular

In [38]:

```
m, n = 10, 20
X, Y = rand(1:m, m), rand(1:n, n)
show_plot_brunner_munzel(X, Y)
```

```
(phat = 0.765,
 sehat = 0.08626255311649104,
 tvalue = 3.0720166564295583,
 df = 27.60349766038036,
 p = 0.5,
 pvalue = 0.0047415464916824275,
 α = 0.05,
 confint_p = [0.5881847509519754, 0.9418152490480246],
 confint_shift = [-8.999999999999998, -1.9999999999999993],
 shifthat = -5.0)
median(X) = 5.5
median(Y) = 12.0
```

Out[38]:

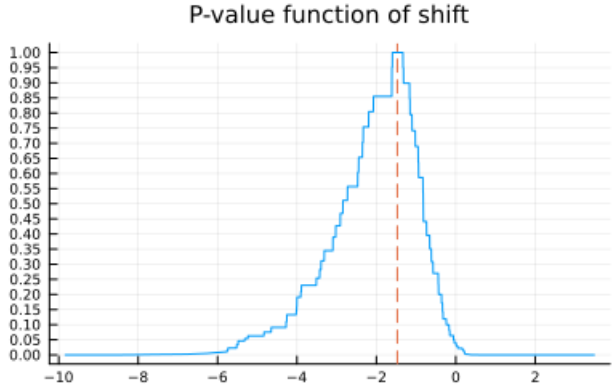


In [39]:

```
distx, disty = LogNormal(), LogNormal(1)
m, n = 10, 10
X, Y = rand(distx, m), rand(disty, n)
show_plot_brunner_munzel(X, Y)
```

```
(phat = 0.75,
 sehat = 0.11205157542647741,
 tvalue = 2.231115439907736,
 df = 18.0,
 p = 0.5,
 pvalue = 0.03863103202434312,
 α = 0.05,
 confint_p = [0.5145883755427825, 0.9854116244572175],
 confint_shift = [-5.318179432610124, -0.05841508479073754],
 shifthat = -1.4733410643825495)
median(X) = 1.308993041607416
median(Y) = 2.9790059563220423
```

Out[39]:



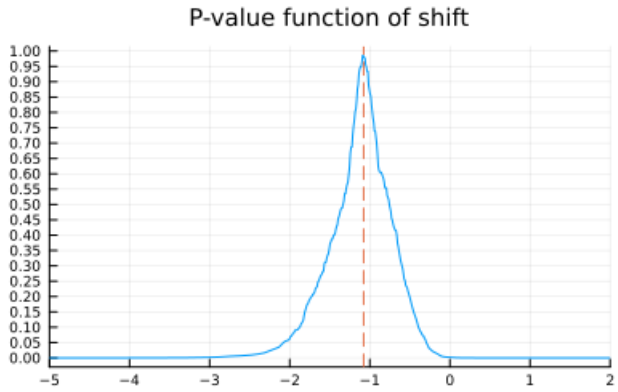
Loading web-font STIX-Web/Size1/Regular

In [40]:

```
distx, disty = LogNormal(), LogNormal(1)
m, n = 40, 40
X, Y = rand(distx, m), rand(disty, n)
show_plot_brunner_munzel(X, Y; xlim=(-5, 2))
```

```
(phat = 0.6912499999999999,
 sehat = 0.05960002903974512,
 tvalue = 3.208891053936604,
 df = 75.84080882915966,
 p = 0.5,
 pvalue = 0.001953574321225764,
 α = 0.05,
 confint_p = [0.5725422250913458, 0.8099577749086541],
 confint_shift = [-2.07997163873439, -0.29464656357396524],
 shifthat = -1.0768936945378669)
median(X) = 1.044779302249417
median(Y) = 2.2527593566191237
```

Out[40]:

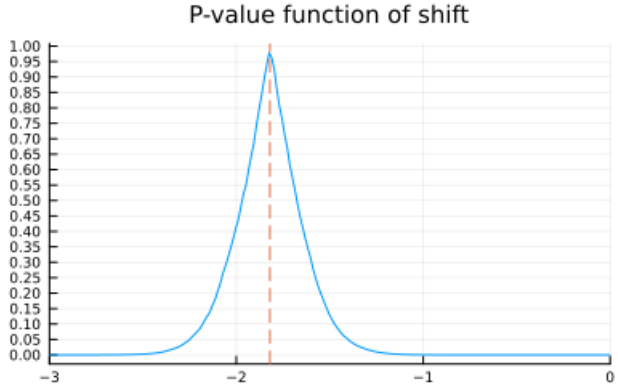


In [41]:

```
distx, disty = LogNormal(), LogNormal(1)
m, n = 160, 160
X, Y = rand(distx, m), rand(disty, n)
show_plot_brunner_munzel(X, Y; xlim=(-3, 0))
```

```
(phat = 0.79234375,
 sehat = 0.025077139980214335,
 tvalue = 11.657778767062629,
 df = 315.6587283148897,
 p = 0.5,
 pvalue = 2.3287816342564484e-26,
 α = 0.05,
 confint_p = [0.7430042840809774, 0.8416832159190226],
 confint_shift = [-2.250313266911184, -1.404197630202607],
 shifthat = -1.8203983580938916)
median(X) = 0.8417596982690712
median(Y) = 2.974128627007357
```

Out[41]:



Loading web-font STIX-Web/Size1/Regular

4 Brunner-Munzel検定とWelchのt検定の比較

4.1 第一種の過誤の確率

Loading web-font STIX-Web/Size1/Regular

In [42]:

```
function sim_brunner_mumzel(;
        distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10,
        L = 10^6)
    pval_bm = Vector{Float64}(undef, L)
    tmpX = [Vector{Float64}(undef, m) for _ in 1:nthreads()]
    tmpY = [Vector{Float64}(undef, n) for _ in 1:nthreads()]
    tmpHx = [Vector{Float64}(undef, m) for _ in 1:nthreads()]
    tmpHy = [Vector{Float64}(undef, n) for _ in 1:nthreads()]
    @threads for i in 1:L
        X = rand!(distx, tmpX[threadid()])
        Y = rand!(disty, tmpY[threadid()])
        pval_bm[i] = pvalue_brunner_munzel(X, Y, tmpHx[threadid()], tmpHy[threadid()])
    end
    ecdf(pval_bm)
end

function sim_welch(;
        distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10,
        L = 10^6)
    pval_w = Vector{Float64}(undef, L)
    tmpX = [Vector{Float64}(undef, m) for _ in 1:nthreads()]
    tmpY = [Vector{Float64}(undef, n) for _ in 1:nthreads()]
    @threads for i in 1:L
        X = rand!(distx, tmpX[threadid()])
        Y = rand!(disty, tmpY[threadid()])
        pval_w[i] = pvalue_welch(X, Y)
    end
    ecdf(pval_w)
end

function printcompact(io, xs...)
    print(IOContext(io, :compact ⇒ true), xs...)
end

function distname(dist)
    replace(sprint(printcompact, dist), r"\{[^\}]*\}"⇒"")
end

function plot_ecdf(ecdf_pval, distx, disty, m, n, a;
        testname = "", kwargs...)
    plot(p -> ecdf_pval(p), 0, 0.1; label="ecdf of P-values")
    plot!([0, 0.1], [0, 0.1]; label="", ls=:dot, c=:black)
    plot!(legend=:topleft)
    plot!(xtick=0:0.01:0.1, ytick=0:0.01:1)
    plot!(xguide="nominal significance level α",
        yguide="probability of P-value < α")
    s = (a < 0 ? "" : "+") * string(round(a; digits=4))
    title!("$(testname)X: $(distname(distx)), m=$m\n\
        Y: $(distname(disty))$s, n=$n")
    plot!(size=(400, 450))
    plot!(; kwargs...)
end

function plot_pvals(;
        distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10,
        L = 10^6, a = nothing, Δμ = nothing, kwargs...)
    @show (mean(distx), std(distx))
    @show (mean(disty), std(disty))

    if isnothing(a)
        @show a = tieshift(distx, disty)
        @show prob_x_le_y(distx, disty + a)
    else
        @show a
        @show median(distx) - median(disty)
    end
    if isnothing(Δμ)
        @show Δμ = mean(distx) - mean(disty)
        @show mean(distx), mean(disty + Δμ)
    else
        @show Δμ
        @show mean(distx), mean(disty + Δμ)
    end

    ecdf_bm = @time sim_brunner_mumzel(;
        distx = distx,
        disty = disty + a,
        m, n, L, kwargs...)
    ecdf_w = @time sim_welch(;
        distx = distx,
```

Loading web-font STIX-Web/S21/Regular

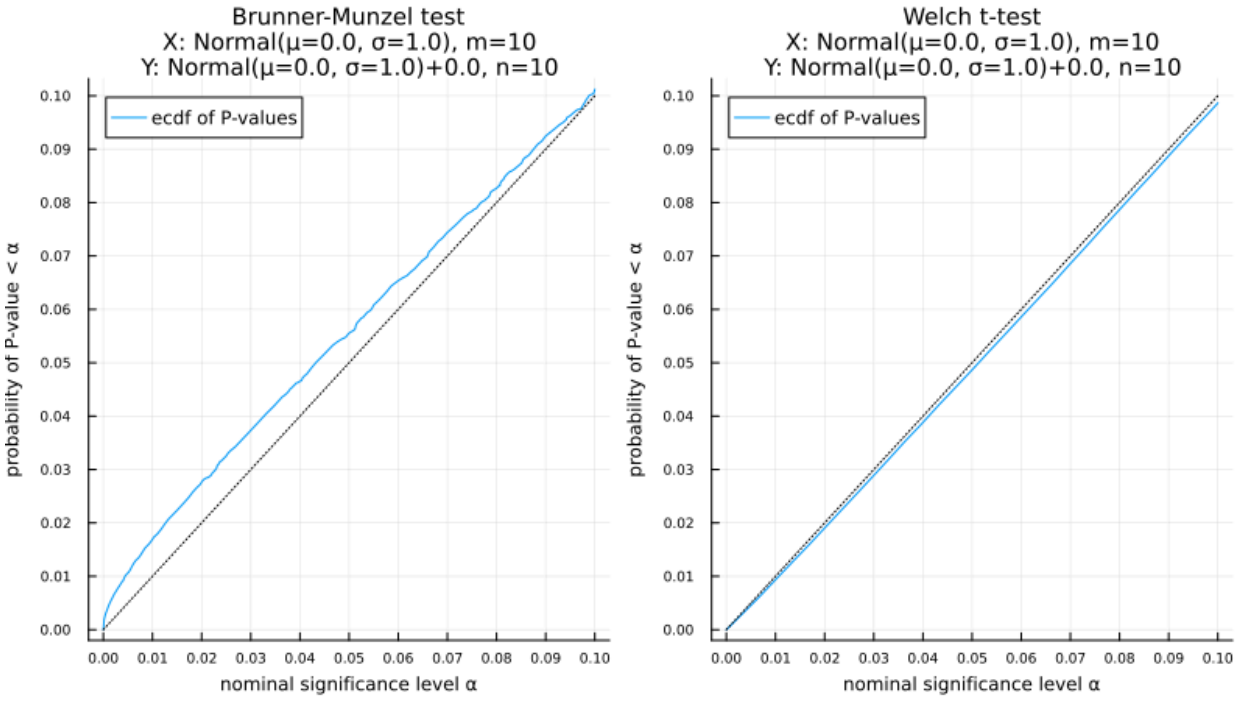
```
        disty = disty + Δμ,
        m, n, L, kwargs...)
    ymax = max(ecdf_bm(0.1), ecdf_w(0.1))
    P1 = plot_ecdf(ecdf_bm, distx, disty, m, n, a;
        testname="Brunner-Munzel test\n",
        ylim=(-0.002, 1.02*ymax), kwargs...)
    P2 = plot_ecdf(ecdf_w, distx, disty, m, n, Δμ;
        testname="Welch t-test\n",
        ylim=(-0.002, 1.02*ymax), kwargs...)
    plot(P1, P2; size=(800, 450), topmargin=3.5Plots.mm)
end
```

Out[42]: plot_pvals (generic function with 1 method)

In [43]:
```
plot_pvals(; distx = Normal(0, 1), disty = Normal(0, 1), m = 10, n = 10)
```

```
(mean(distx), std(distx)) = (0.0, 1.0)
(mean(disty), std(disty)) = (0.0, 1.0)
a = tieshift(distx, disty) = 0.0
prob_x_le_y(distx, disty + a) = 0.5
Δμ = mean(distx) - mean(disty) = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.323238 seconds (640.44 k allocations: 196.441 MiB, 3.10% gc time, 212.22% compilation ti
me: 4% of which was recompilation)
  0.311052 seconds (629.88 k allocations: 208.505 MiB, 10.85% gc time, 170.78% compilation t
ime)
```

Out[43]:

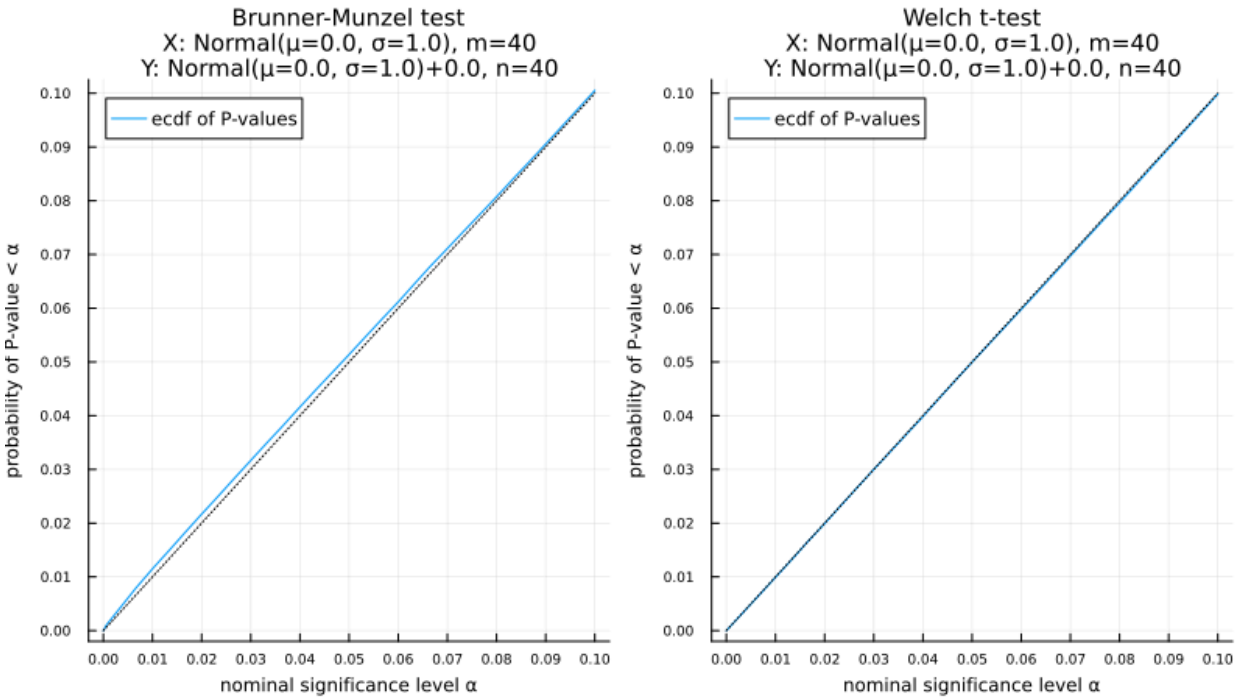


Loading web-font STIX-Web/Size1/Regular

In [44]:
```
1 plot_pvals(; distx = Normal(0, 1), disty = Normal(0, 1), m = 40, n = 40)
```

```
(mean(distx), std(distx)) = (0.0, 1.0)
(mean(disty), std(disty)) = (0.0, 1.0)
a = tieshift(distx, disty) = 0.0
prob_x_le_y(distx, disty + a) = 0.5
Δμ = mean(distx) - mean(disty) = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.697588 seconds (636.69 k allocations: 215.096 MiB)
  0.290773 seconds (645.87 k allocations: 217.731 MiB, 13.63% gc time)
```

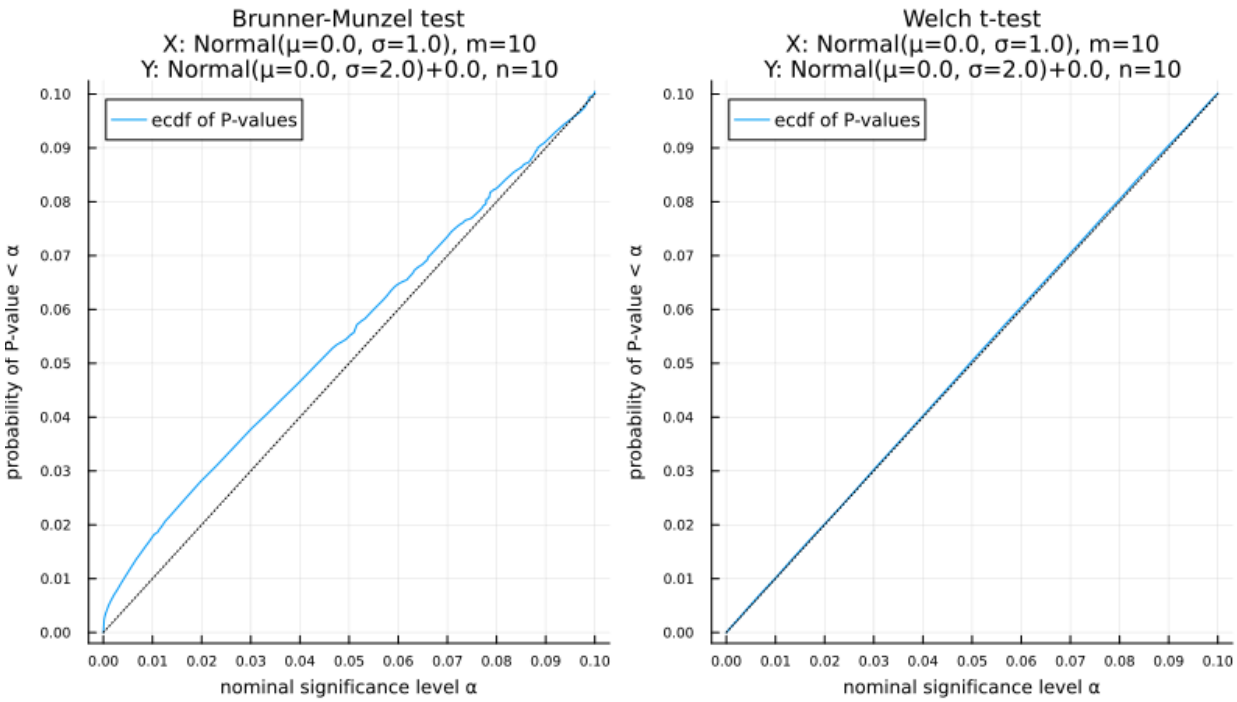
Out[44]:



In [45]:
```
1 plot_pvals(; distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10)
```

```
(mean(distx), std(distx)) = (0.0, 1.0)
(mean(disty), std(disty)) = (0.0, 2.0)
a = tieshift(distx, disty) = 7.685641860444171e-14
prob_x_le_y(distx, disty + a) = 0.5
Δμ = mean(distx) - mean(disty) = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.324353 seconds (524.30 k allocations: 182.475 MiB, 9.79% gc time)
  0.294450 seconds (585.29 k allocations: 200.184 MiB, 20.81% gc time)
```

Out[45]:

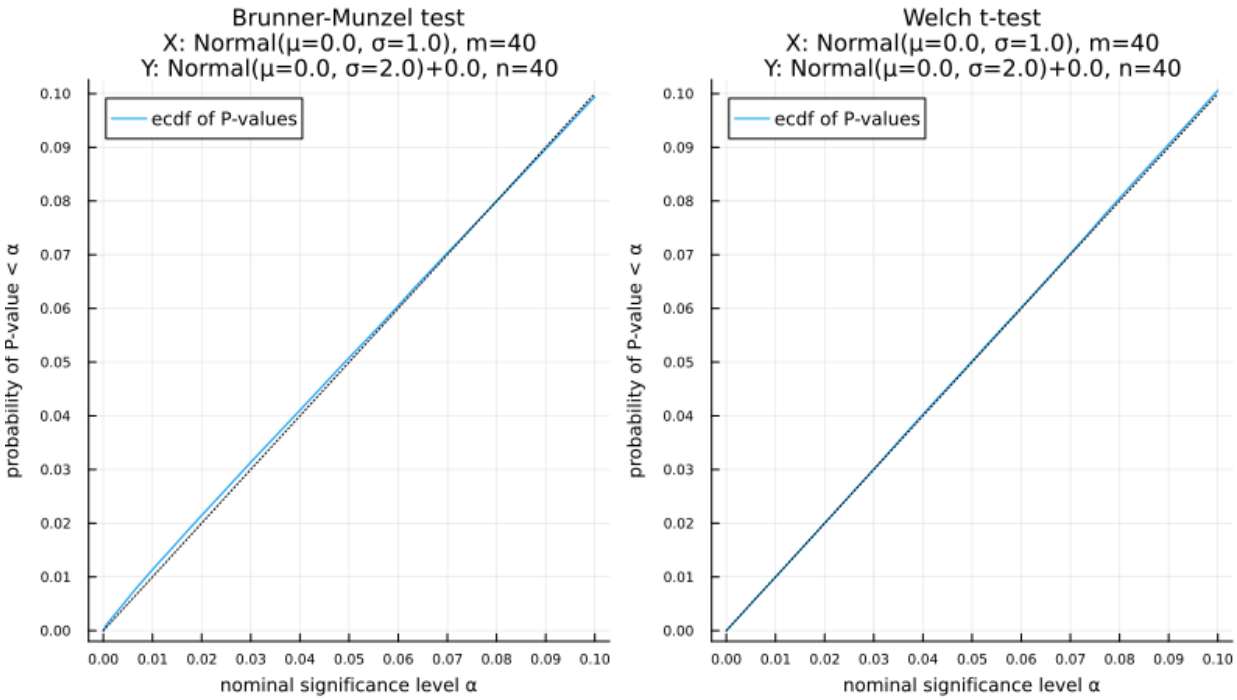


Loading web-font STIX-Web/Size1/Regular

In [46]:
```
1 plot_pvals(; distx = Normal(0, 1), disty = Normal(0, 2), m = 40, n = 40)
```

```
(mean(distx), std(distx)) = (0.0, 1.0)
(mean(disty), std(disty)) = (0.0, 2.0)
a = tieshift(distx, disty) = 7.685641860444171e-14
prob_x_le_y(distx, disty + a) = 0.5
Δμ = mean(distx) - mean(disty) = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.782463 seconds (633.98 k allocations: 214.285 MiB, 4.49% gc time)
  0.254228 seconds (647.02 k allocations: 218.062 MiB)
```

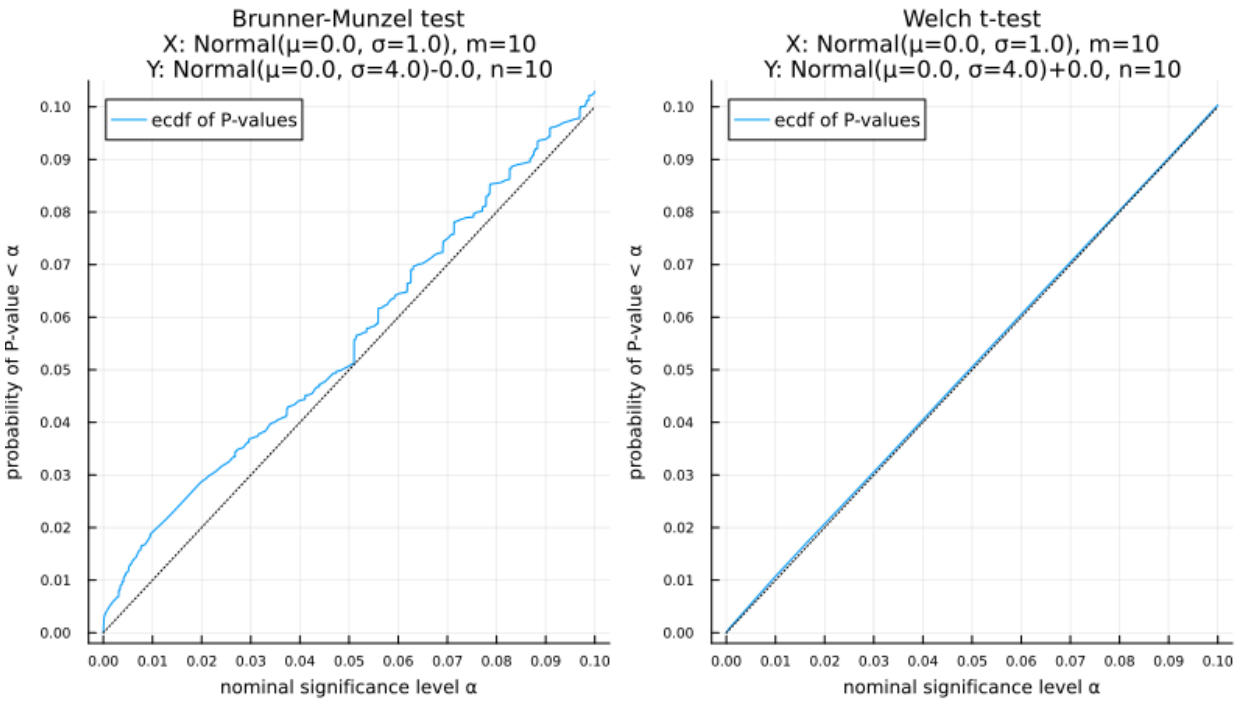
Out[46]:



In [47]:
```
1 plot_pvals(; distx = Normal(0, 1), disty = Normal(0, 4), m = 10, n = 10)
```

```
(mean(distx), std(distx)) = (0.0, 1.0)
(mean(disty), std(disty)) = (0.0, 4.0)
a = tieshift(distx, disty) = -2.715980146908965e-13
prob_x_le_y(distx, disty + a) = 0.5
Δμ = mean(distx) - mean(disty) = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.323551 seconds (486.84 k allocations: 171.615 MiB, 11.33% gc time)
  0.263402 seconds (534.49 k allocations: 185.457 MiB, 15.19% gc time)
```

Out[47]:

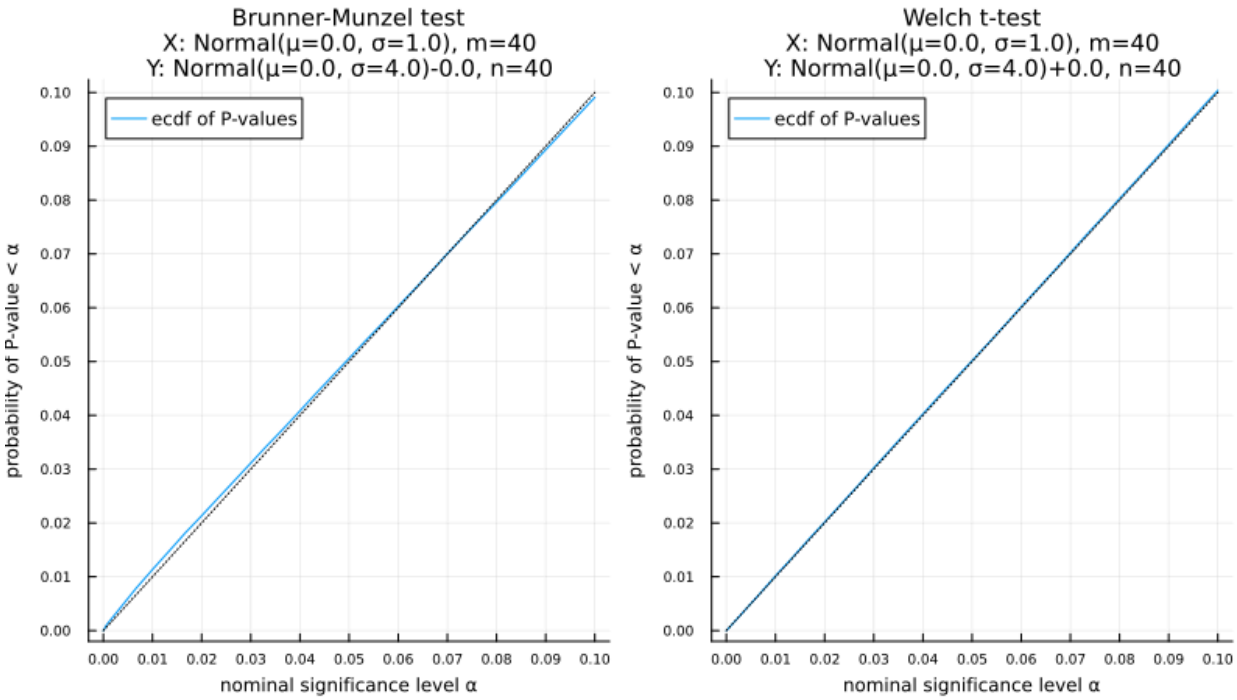


Loading web-font STIX-Web/Size1/Regular

In [48]:
```
1 plot_pvals(; distx = Normal(0, 1), disty = Normal(0, 4), m = 40, n = 40)
```

```
(mean(distx), std(distx)) = (0.0, 1.0)
(mean(disty), std(disty)) = (0.0, 4.0)
a = tieshift(distx, disty) = -2.715980146908965e-13
prob_x_le_y(distx, disty + a) = 0.5
Δμ = mean(distx) - mean(disty) = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.783298 seconds (633.66 k allocations: 214.193 MiB, 4.72% gc time)
  0.244073 seconds (645.09 k allocations: 217.503 MiB)
```

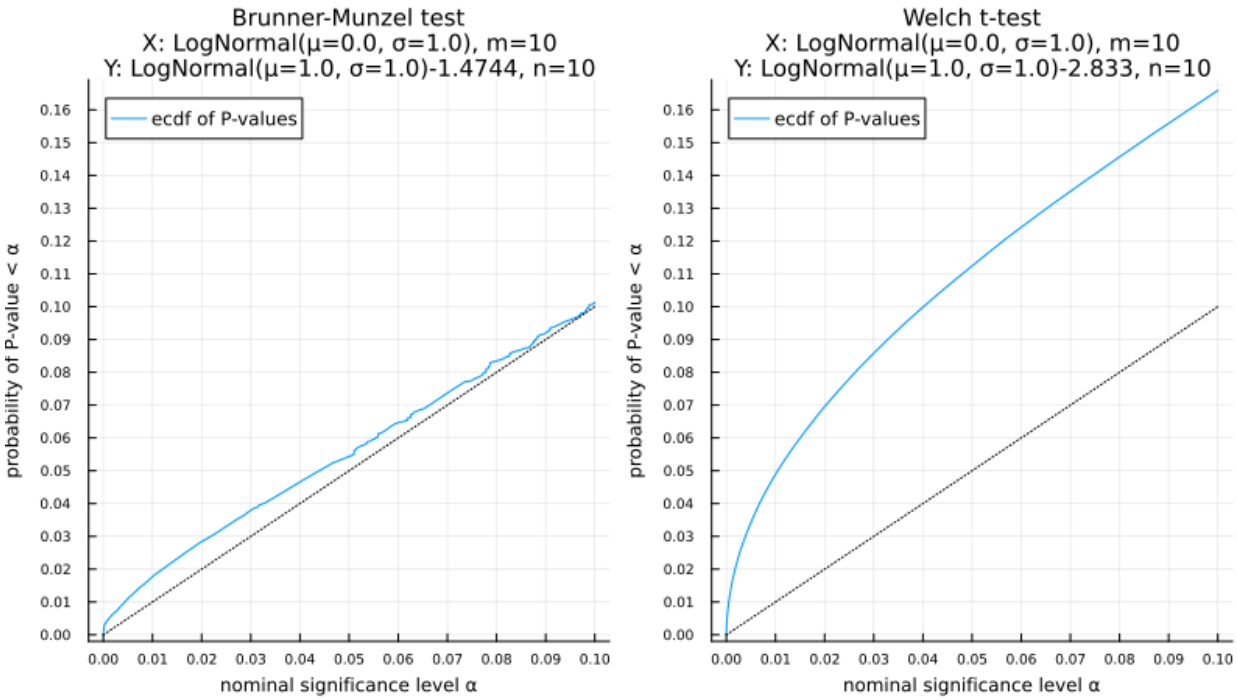
Out[48]:



In [49]:
```
1 plot_pvals(; distx = LogNormal(), disty = LogNormal(1), m = 10, n = 10)
```

```
(mean(distx), std(distx)) = (1.6487212707001282, 2.1611974158950877)
(mean(disty), std(disty)) = (4.4816890703380645, 5.874743663340262)
a = tieshift(distx, disty) = -1.4744426128871542
prob_x_le_y(distx, disty + a) = 0.5
Δμ = mean(distx) - mean(disty) = -2.8329677996379363
(mean(distx), mean(disty + Δμ)) = (1.6487212707001282, 1.6487212707001282)
  0.333222 seconds (528.03 k allocations: 179.550 MiB, 8.37% gc time, 120.31% compilation ti
me)
  0.259252 seconds (623.33 k allocations: 208.906 MiB, 124.62% compilation time)
```

Out[49]:

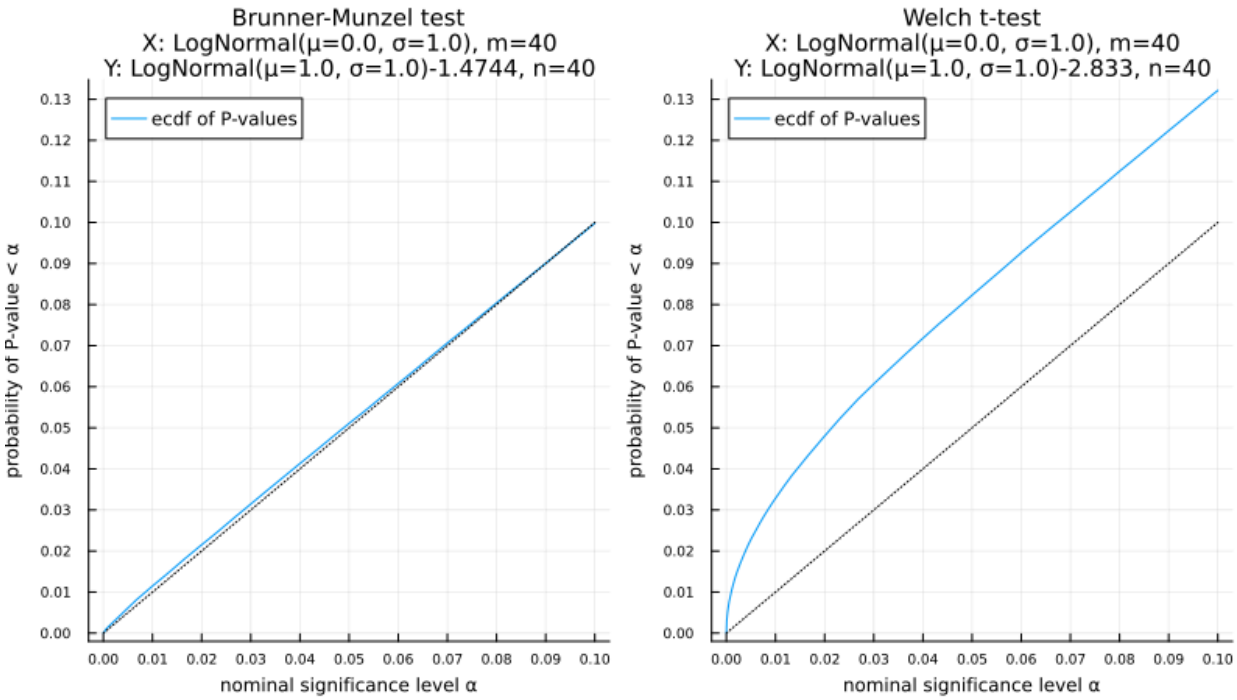


Loading web-font STIX-Web/Size1/Regular

In [50]:
```
1 plot_pvals(; distx = LogNormal(), disty = LogNormal(1), m = 40, n = 40)
```

```
(mean(distx), std(distx)) = (1.6487212707001282, 2.1611974158950877)
(mean(disty), std(disty)) = (4.4816890703380645, 5.874743663340262)
a = tieshift(distx, disty) = -1.4744426128871542
prob_x_le_y(distx, disty + a) = 0.5
Δμ = mean(distx) - mean(disty) = -2.8329677996379363
(mean(distx), mean(disty + Δμ)) = (1.6487212707001282, 1.6487212707001282)
  0.883382 seconds (636.07 k allocations: 214.890 MiB, 4.82% gc time)
  0.419585 seconds (709.10 k allocations: 236.063 MiB, 15.21% gc time)
```

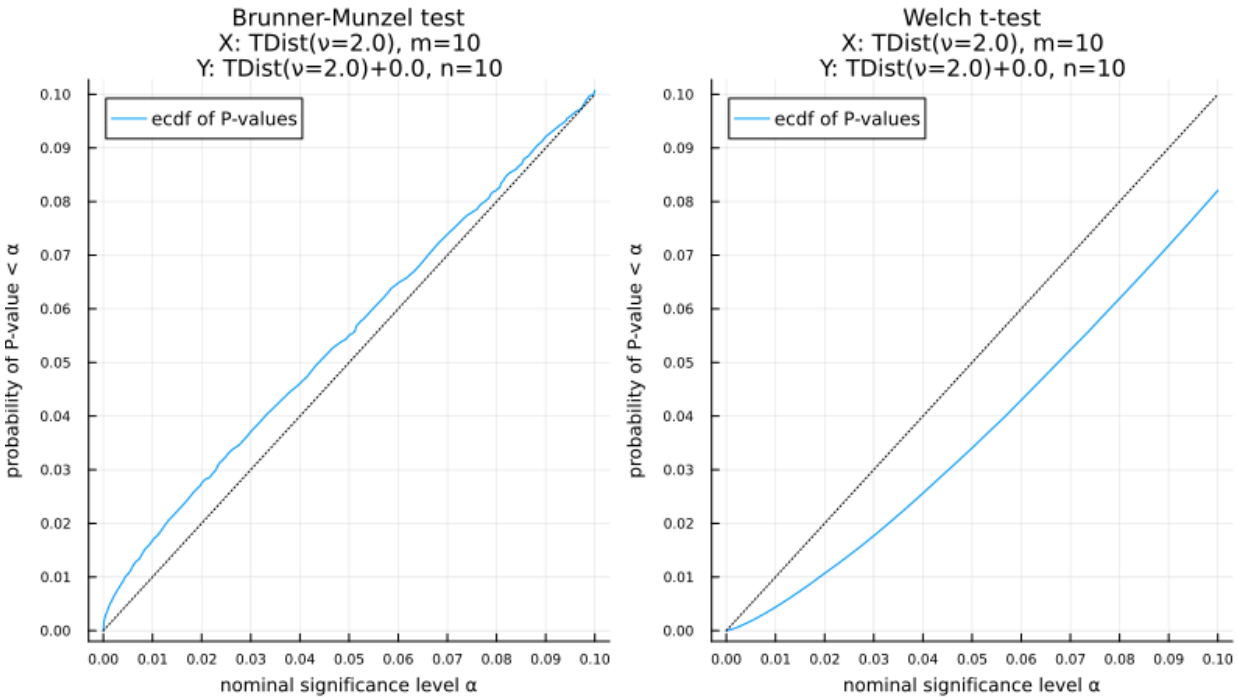
Out[50]:



In [51]:
```
1 plot_pvals(; distx = TDist(2), disty = TDist(2), m = 10, n = 10, Δμ = 0.0)
```

```
(mean(distx), std(distx)) = (0.0, Inf)
(mean(disty), std(disty)) = (0.0, Inf)
a = tieshift(distx, disty) = 0.0
prob_x_le_y(distx, disty + a) = 0.5000000000000001
Δμ = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.434859 seconds (688.11 k allocations: 199.328 MiB, 5.49% gc time, 369.47% compilation ti
me)
  0.343614 seconds (674.43 k allocations: 223.719 MiB, 17.65% gc time, 98.98% compilation ti
me)
```

Out[51]:

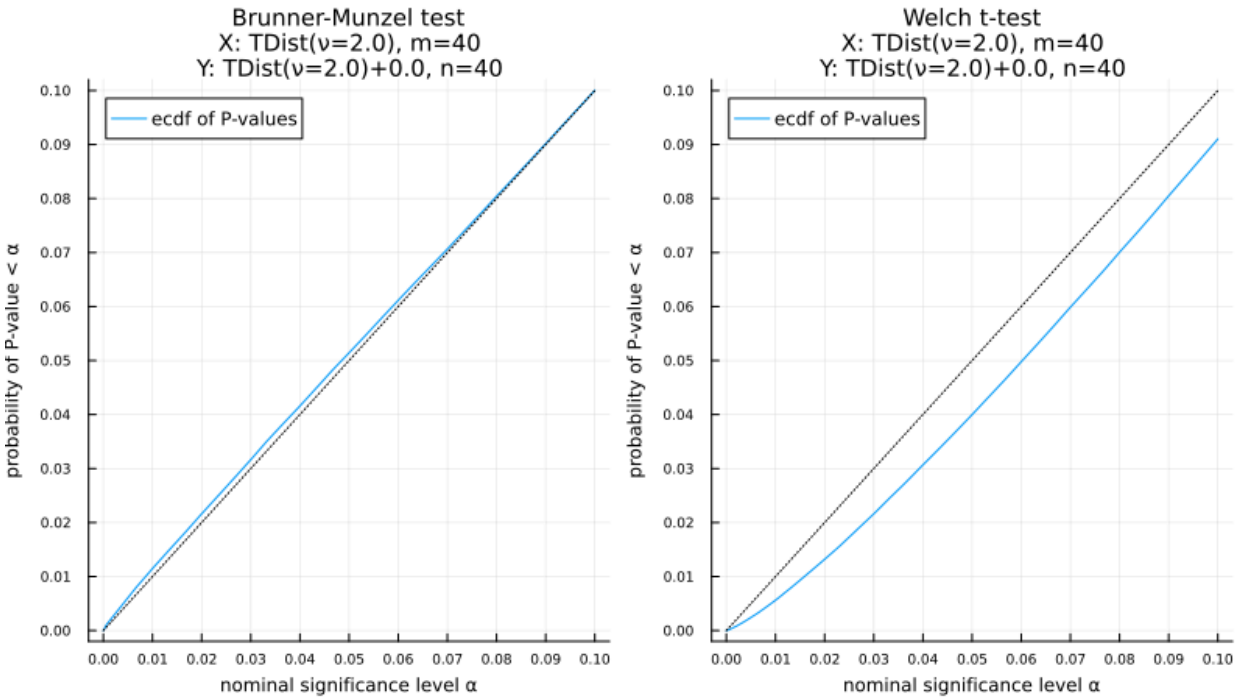


Loading web-font STIX-Web/Size1/Regular

In [52]:
```
1 plot_pvals(; distx = TDist(2), disty = TDist(2), m = 40, n = 40, Δμ = 0.0)
```

```
(mean(distx), std(distx)) = (0.0, Inf)
(mean(disty), std(disty)) = (0.0, Inf)
a = tieshift(distx, disty) = 0.0
prob_x_le_y(distx, disty + a) = 0.5000000000000001
Δμ = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.896450 seconds (636.62 k allocations: 215.052 MiB, 3.57% gc time)
  0.404594 seconds (690.55 k allocations: 230.683 MiB)
```

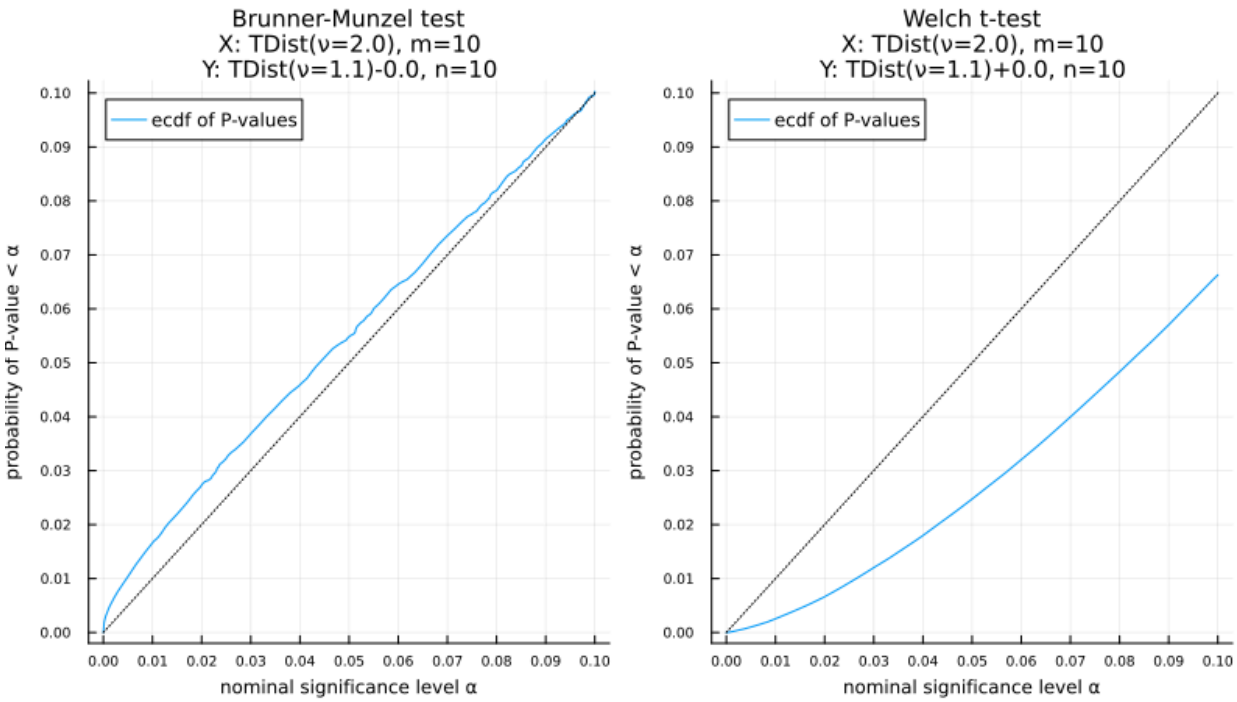
Out[52]:



In [53]:
```
1 plot_pvals(; distx = TDist(2), disty = TDist(1.1), m = 10, n = 10, Δμ = 0.0)
```

```
(mean(distx), std(distx)) = (0.0, Inf)
(mean(disty), std(disty)) = (0.0, Inf)
a = tieshift(distx, disty) = -3.4064502213362996e-9
prob_x_le_y(distx, disty + a) = 0.5
Δμ = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  0.425950 seconds (547.89 k allocations: 189.315 MiB, 14.85% gc time)
  0.381495 seconds (714.35 k allocations: 237.577 MiB, 11.94% gc time)
```

Out[53]:



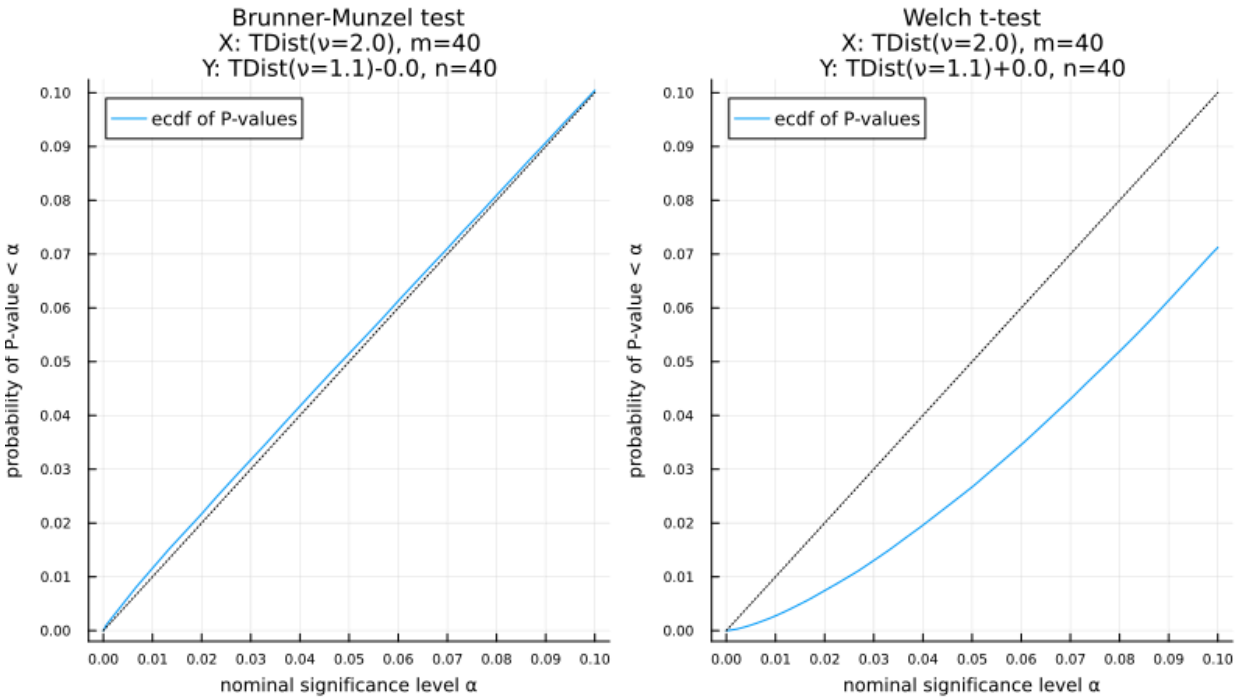
Loading web-font STIX-Web/Size1/Regular

In [54]:

```
1 plot_pvals(; distx = TDist(2), disty = TDist(1.1), m = 40, n = 40, Δμ = 0.0)
```

```
(mean(distx), std(distx)) = (0.0, Inf)
(mean(disty), std(disty)) = (0.0, Inf)
a = tieshift(distx, disty) = -3.4064502213362996e-9
prob_x_le_y(distx, disty + a) = 0.5
Δμ = 0.0
(mean(distx), mean(disty + Δμ)) = (0.0, 0.0)
  1.052793 seconds (636.12 k allocations: 214.907 MiB)
  0.625337 seconds (757.21 k allocations: 250.009 MiB, 4.88% gc time)
```

Out[54]:



Loading web-font STIX-Web/Size1/Regular

4.2 Brunner-Munzel検定は中央値に関する検定ではないことの証拠

In [55]:

```
distx, disty = Uniform(-1, 1), Exponential()
m, n, = 100, 100

@show distx, std(distx)
@show disty, std(disty)

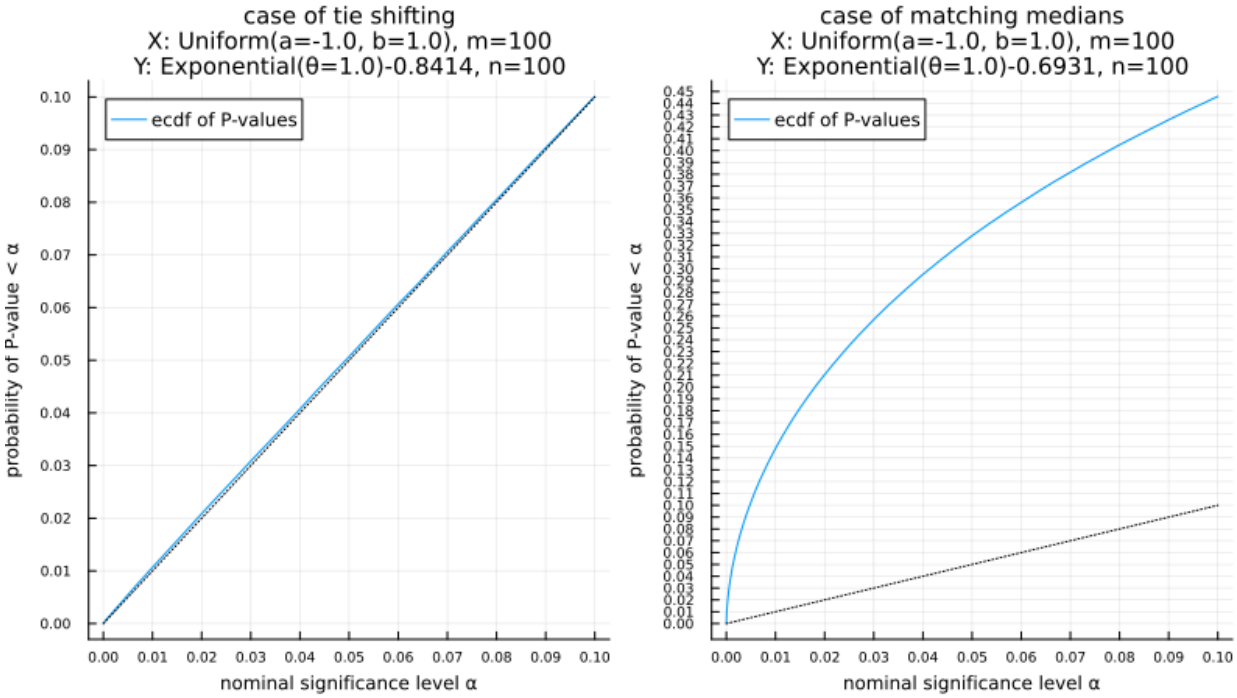
@show a = tieshift(distx, disty)
ecdf_pval1 = @time sim_brunner_mumzel(;
    distx = distx, disty = disty + a, m, n)
P1 = plot_ecdf(ecdf_pval1, distx, disty, m, n, a;
    testname="case of tie shifting\n")

@show a = median(distx) - median(disty)
ecdf_pval2 = @time sim_brunner_mumzel(;
    distx = distx, disty = disty + a, m, n)
P2 = plot_ecdf(ecdf_pval2, distx, disty, m, n, a;
    testname="case of matching medians\n")

plot(P1, P2; size=(800, 450), topmargin=4Plots.mm)
```

```
(distx, std(distx)) = (Uniform{Float64}(a=-1.0, b=1.0), 0.5773502691896257)
(disty, std(disty)) = (Exponential{Float64}(θ=1.0), 1.0)
a = tieshift(distx, disty) = -0.8414056600399943
  3.157133 seconds (798.40 k allocations: 227.154 MiB, 2.15% gc time, 28.78% compilation tim
e)
a = median(distx) - median(disty) = -0.6931471805599453
  2.948860 seconds (1.40 M allocations: 437.444 MiB, 1.55% gc time)
```

Out[55]:



Loading web-font STIX-Web/Size1/Regular

In [56]:

```
distx, disty = Uniform(-1, 1), Exponential()
@show distx, std(distx)
@show disty, std(disty)

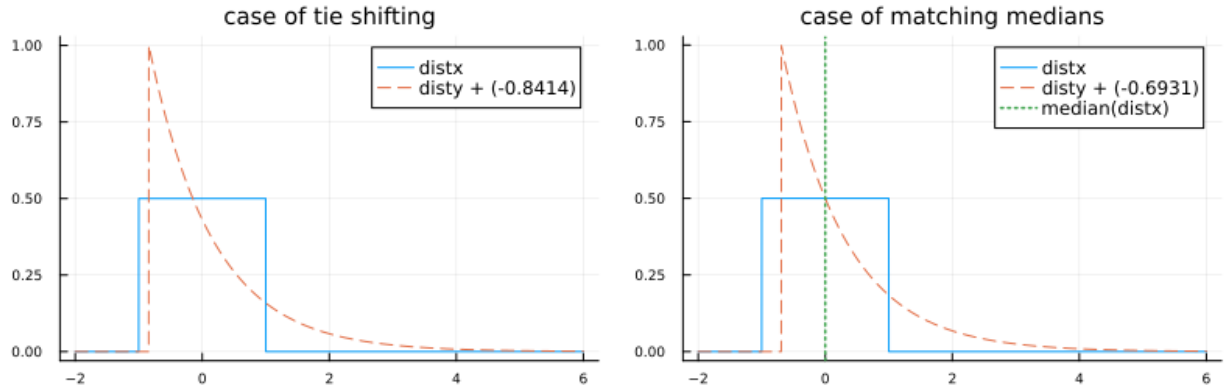
a = @show tieshift(distx, disty)
P1 = plot(distx, -2, 6; label="distx")
plot!(disty + a, -2, 6; label="disty + ($(round(a; digits=4)))", ls=:dash)
title!("case of tie shifting")

a = @show median(distx) - median(disty)
P2 = plot(distx, -2, 6; label="distx")
plot!(disty + a, -2, 6; label="disty + ($(round(a; digits=4)))", ls=:dash)
vline!([median(distx)]; label="median(distx)", ls=:dot, lw=1.5)
title!("case of matching medians")

plot(P1, P2; size=(800, 250))
```

```
(distx, std(distx)) = (Uniform{Float64}(a=-1.0, b=1.0), 0.5773502691896257)
(disty, std(disty)) = (Exponential{Float64}(θ=1.0), 1.0)
tieshift(distx, disty) = -0.8414056600399943
median(distx) - median(disty) = -0.6931471805599453
```

Out[56]:



Loading web-font STIX-Web/Size1/Regular

In [57]:

```
distx, disty = Uniform(-1, 1), Exponential(4)
m, n, = 100, 100

@show distx, std(distx)
@show disty, std(disty)

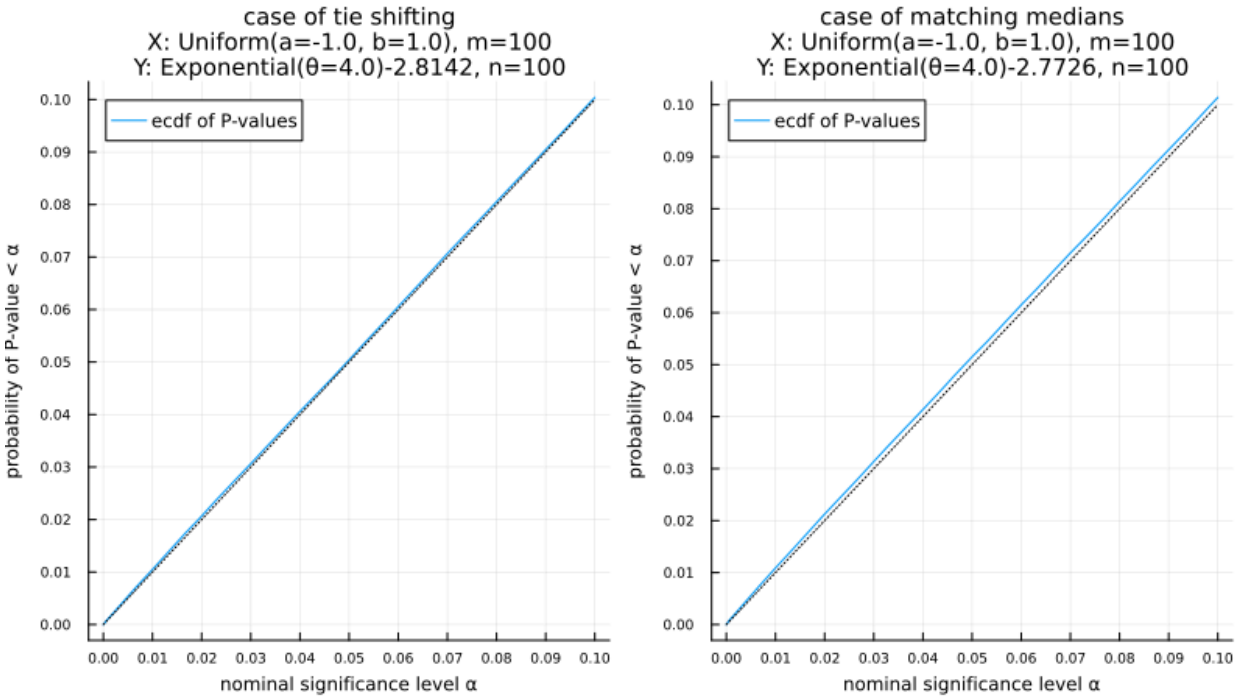
@show a = tieshift(distx, disty)
ecdf_pval1 = @time sim_brunner_mumzel(;
    distx = distx, disty = disty + a, m, n)
P1 = plot_ecdf(ecdf_pval1, distx, disty, m, n, a;
    testname="case of tie shifting\n")

@show a = median(distx) - median(disty)
ecdf_pval2 = @time sim_brunner_mumzel(;
    distx = distx, disty = disty + a, m, n)
P2 = plot_ecdf(ecdf_pval2, distx, disty, m, n, a;
    testname="case of matching medians\n")

plot(P1, P2; size=(800, 450), topmargin=4Plots.mm)
```

```
(distx, std(distx)) = (Uniform{Float64}(a=-1.0, b=1.0), 0.5773502691896257)
(disty, std(disty)) = (Exponential{Float64}(θ=4.0), 4.0)
a = tieshift(distx, disty) = -2.814168911097314
  2.906576 seconds (639.61 k allocations: 215.939 MiB)
a = median(distx) - median(disty) = -2.772588722239781
  2.929456 seconds (645.53 k allocations: 217.653 MiB, 1.19% gc time)
```

Out[57]:



Loading web-font STIX-Web/Size1/Regular

In [58]:

```
distx, disty = Uniform(-1, 1), Exponential(4)
@show distx, std(distx)
@show disty, std(disty)

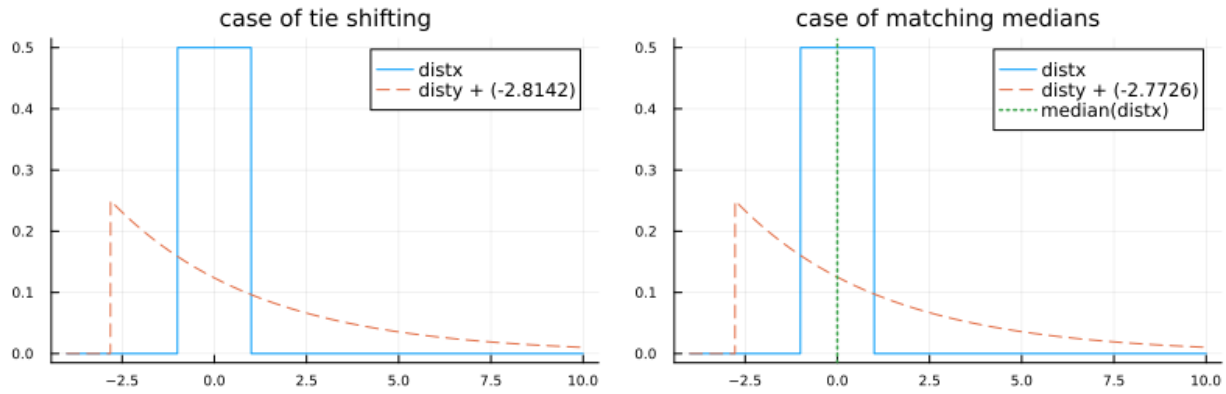
a = @show tieshift(distx, disty)
P1 = plot(distx, -4, 10; label="distx")
plot!(disty + a, -4, 10; label="disty + ($(round(a; digits=4)))", ls=:dash)
title!("case of tie shifting")

a = @show median(distx) - median(disty)
P2 = plot(distx, -4, 10; label="distx")
plot!(disty + a, -4, 10; label="disty + ($(round(a; digits=4)))", ls=:dash)
vline!([median(distx)]; label="median(distx)", ls=:dot, lw=1.5)
title!("case of matching medians")

plot(P1, P2; size=(800, 250))
```

```
(distx, std(distx)) = (Uniform{Float64}(a=-1.0, b=1.0), 0.5773502691896257)
(disty, std(disty)) = (Exponential{Float64}(θ=4.0), 4.0)
tieshift(distx, disty) = -2.814168911097314
median(distx) - median(disty) = -2.772588722239781
```

Out[58]:



Loading web-font STIX-Web/Size1/Regular

In [59]:

```
distx, disty = Uniform(-1, 1), Exponential(0.5773502691896257)
m, n, = 100, 100

@show distx, std(distx)
@show disty, std(disty)

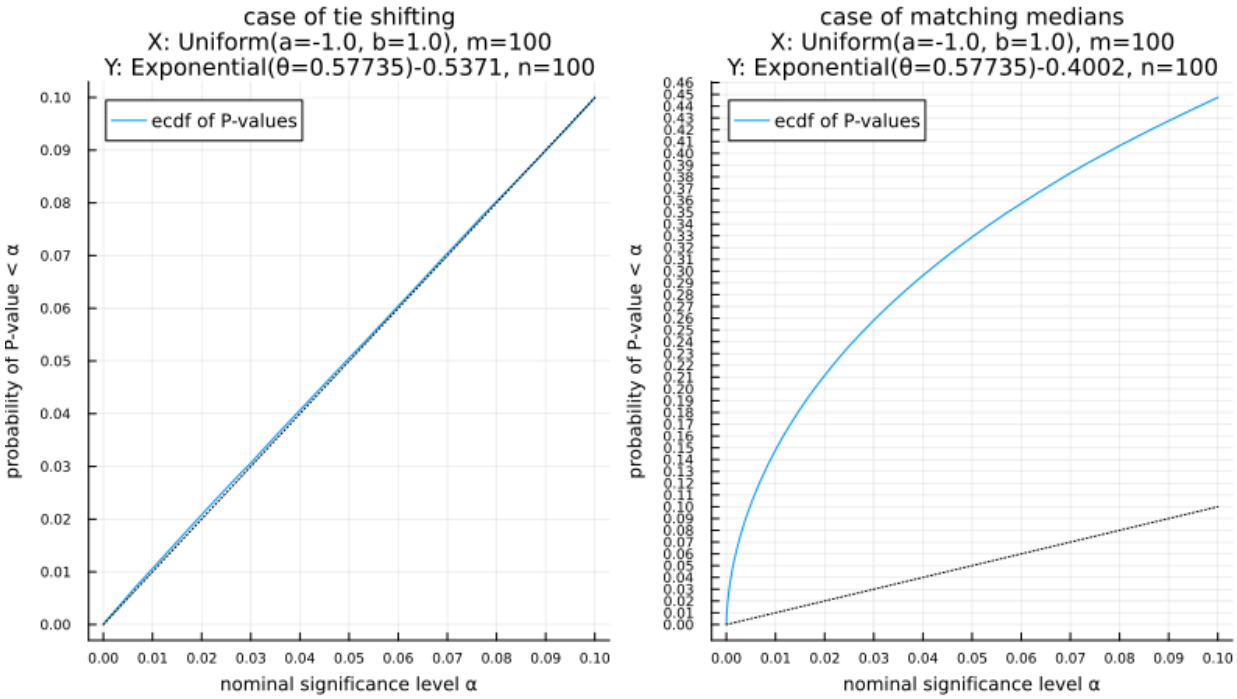
@show a = tieshift(distx, disty)
ecdf_pval1 = @time sim_brunner_mumzel(;
    distx = distx, disty = disty + a, m, n)
P1 = plot_ecdf(ecdf_pval1, distx, disty, m, n, a;
    testname="case of tie shifting\n")

@show a = median(distx) - median(disty)
ecdf_pval2 = @time sim_brunner_mumzel(;
    distx = distx, disty = disty + a, m, n)
P2 = plot_ecdf(ecdf_pval2, distx, disty, m, n, a;
    testname="case of matching medians\n")

plot(P1, P2; size=(800, 450), topmargin=4Plots.mm)
```

```
(distx, std(distx)) = (Uniform{Float64}(a=-1.0, b=1.0), 0.5773502691896257)
(disty, std(disty)) = (Exponential{Float64}(θ=0.5773502691896257), 0.5773502691896257)
a = tieshift(distx, disty) = -0.5370568188698567
  3.047695 seconds (641.75 k allocations: 216.560 MiB, 1.90% gc time)
a = median(distx) - median(disty) = -0.40018871128431455
  3.046602 seconds (1.40 M allocations: 437.657 MiB, 2.13% gc time)
```

Out[59]:



Loading web-font STIX-Web/Size1/Regular

In [60]:

```
distx, disty = Uniform(-1, 1), Exponential(0.5773502691896257)
@show distx, std(distx)
@show disty, std(disty)

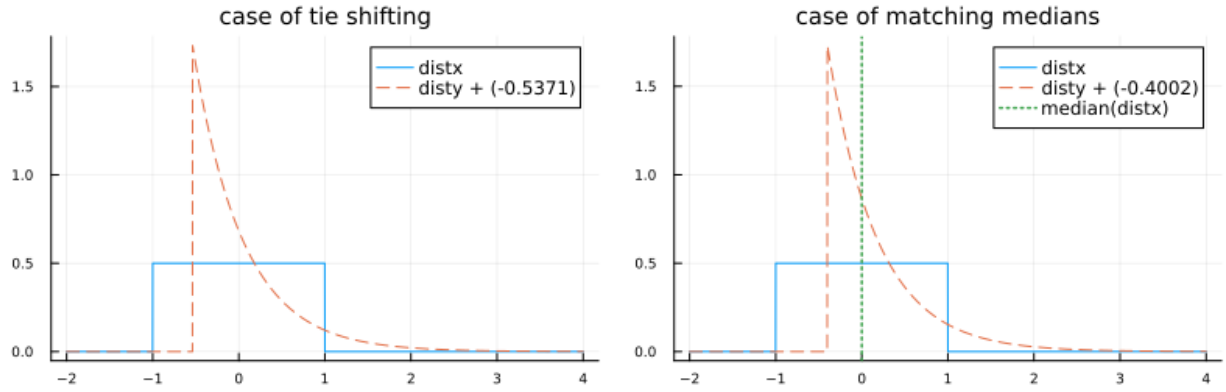
a = @show tieshift(distx, disty)
P1 = plot(distx, -2, 4; label="distx")
plot!(disty + a, -2, 4; label="disty + ($(round(a; digits=4)))", ls=:dash)
title!("case of tie shifting")

a = @show median(distx) - median(disty)
P2 = plot(distx, -2, 4; label="distx")
plot!(disty + a, -2, 4; label="disty + ($(round(a; digits=4)))", ls=:dash)
vline!([median(distx)]; label="median(distx)", ls=:dot, lw=1.5)
title!("case of matching medians")

plot(P1, P2; size=(800, 250))
```

```
(distx, std(distx)) = (Uniform{Float64}(a=-1.0, b=1.0), 0.5773502691896257)
(disty, std(disty)) = (Exponential{Float64}(θ=0.5773502691896257), 0.5773502691896257)
tieshift(distx, disty) = -0.5370568188698567
median(distx) - median(disty) = -0.40018871128431455
```

Out[60]:



4.3 BM検定による互角シフトの信頼区間とWelchのt検定による平均の差の信頼区間の比較

In [61]:

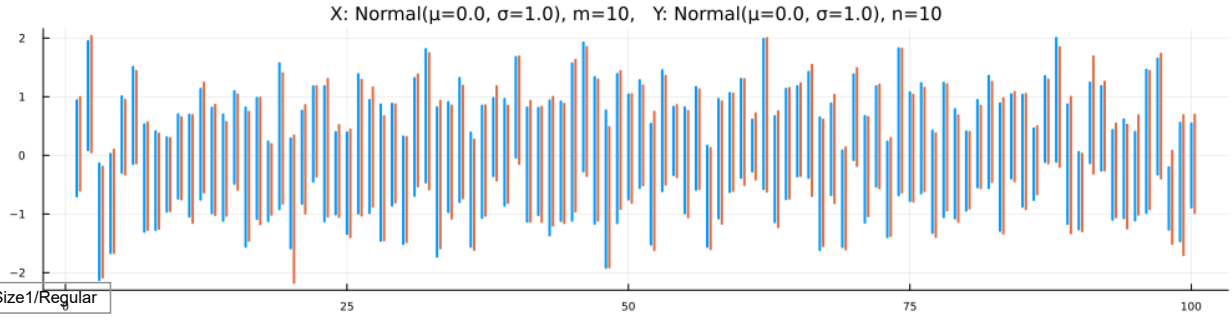
```
function plot_confints(;
        distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10,
        L = 100, kwargs...)
    a = tieshift(distx, disty)
    Δμ = mean(distx) - mean(disty)
    BM = fill(zeros(2), 0)
    W = fill(zeros(2), 0)
    for _ in 1:L
        X = rand(distx, m)
        Y = rand(disty, n)
        push!(BM, brunner_munzel(X, Y .+ a).confint_shift)
        push!(W, confint_welch(X, Y .+ Δμ))
    end
    P = plot()
    for i in 1:L
        plot!(fill(i, 2), [first(BM[i]), last(BM[i])]; label="", c=1, lw=2)
        plot!(fill(i+0.3, 2), [first(W[i]), last(W[i])]; label="", c=2, lw=2)
    end
    title!("X: $(distname(distx)), m=$m,   Y: $(distname(disty)), n=$n")
    plot!(size=(1000, 250))
end
```

Out[61]: plot_confints (generic function with 1 method)

In [62]:

```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 1), m = 10, n = 10)
```

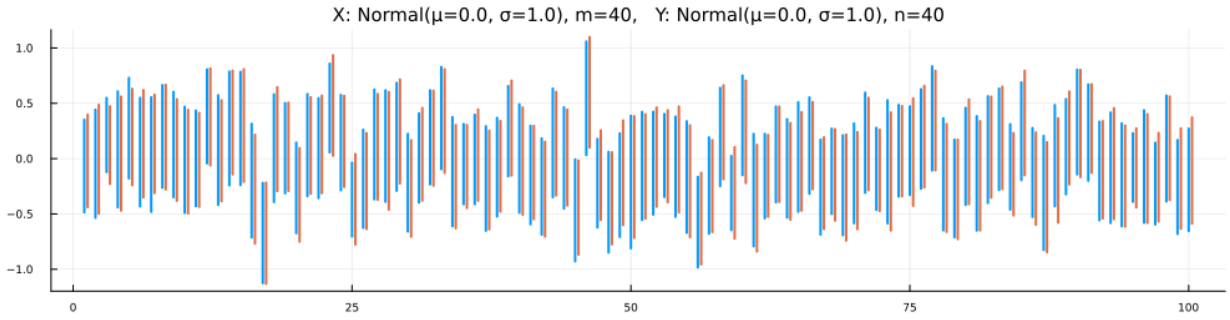
Out[62]:



Loading web-font STIX-Web/Size1/Regular

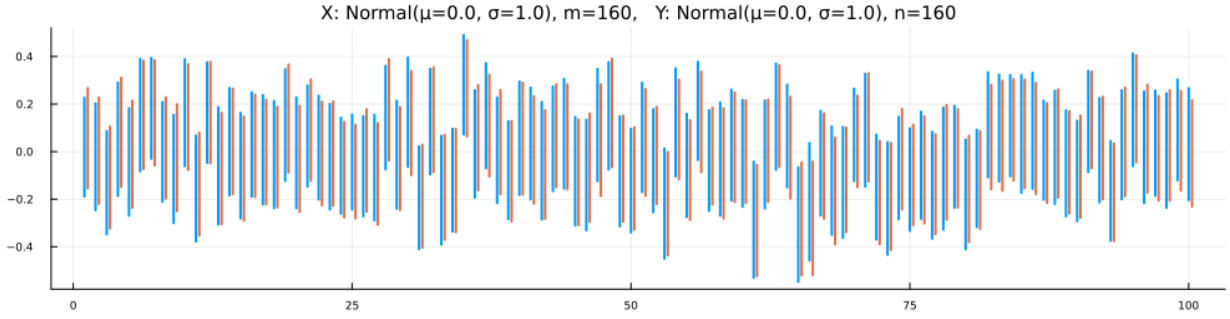
In [63]:
```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 1), m = 40, n = 40)
```

Out[63]:



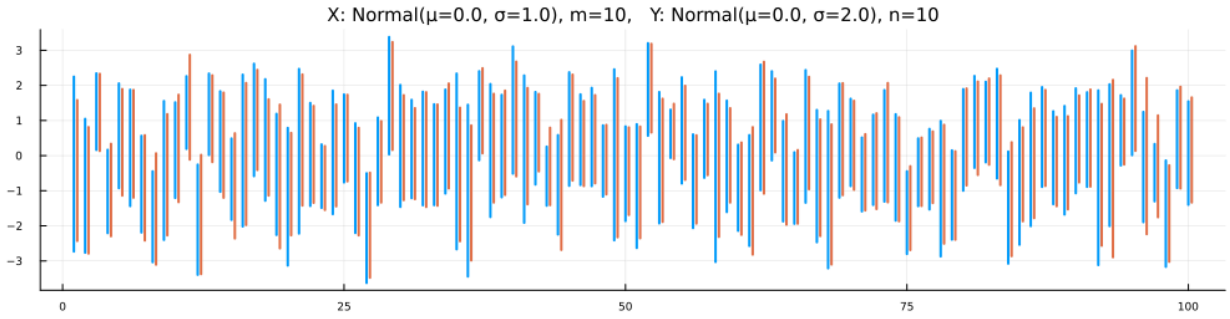
In [64]:
```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 1), m = 160, n = 160)
```

Out[64]:



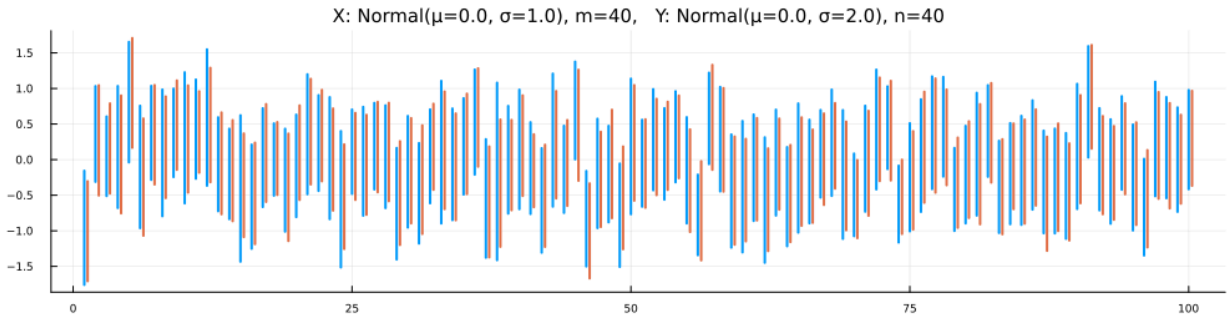
In [65]:
```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10)
```

Out[65]:



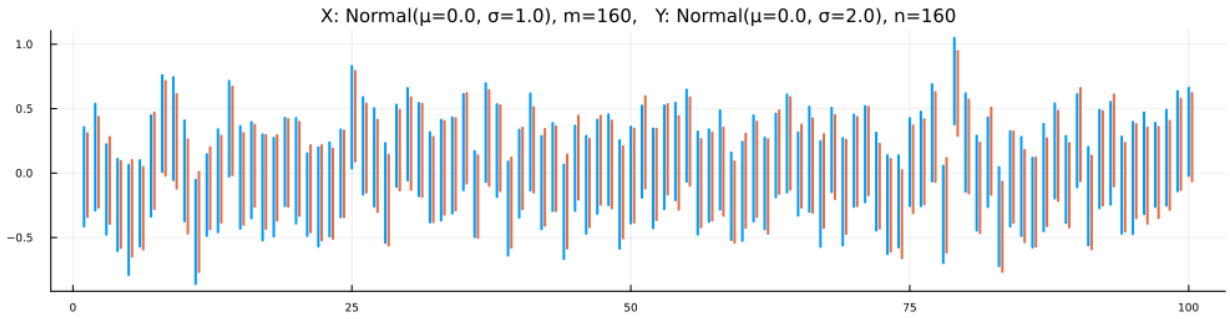
In [66]:
```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 2), m = 40, n = 40)
```

Out[66]:



In [67]:
```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 2), m = 160, n = 160)
```

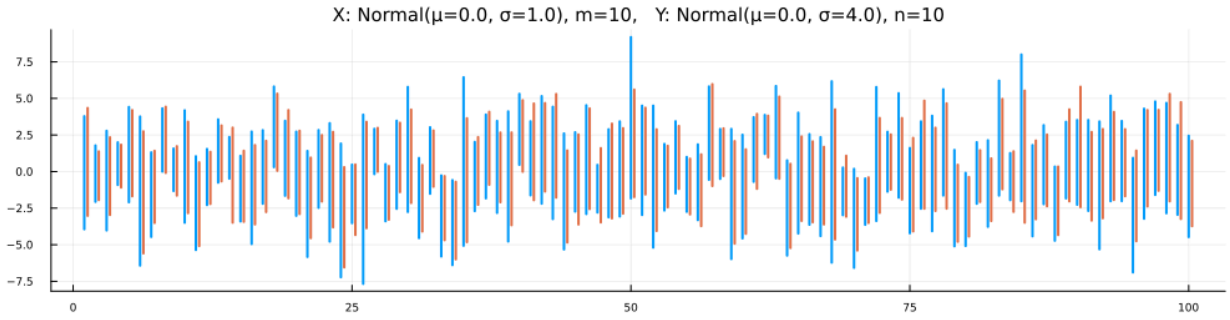
Out[67]:



Loading web-font STIX-Web/Size1/Regular

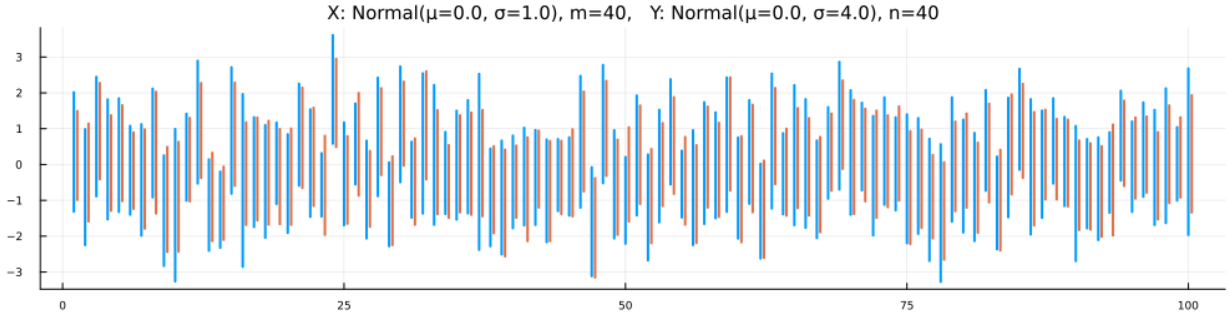
In [68]:
```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 4), m = 10, n = 10)
```

Out[68]:



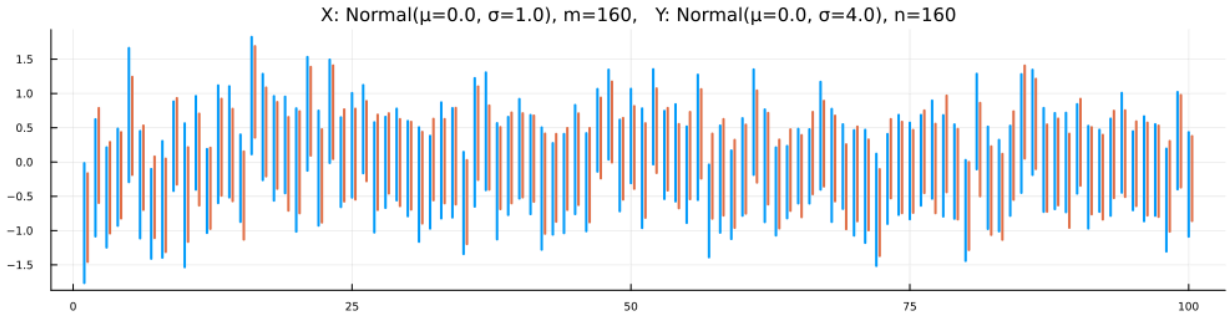
In [69]:
```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 4), m = 40, n = 40)
```

Out[69]:



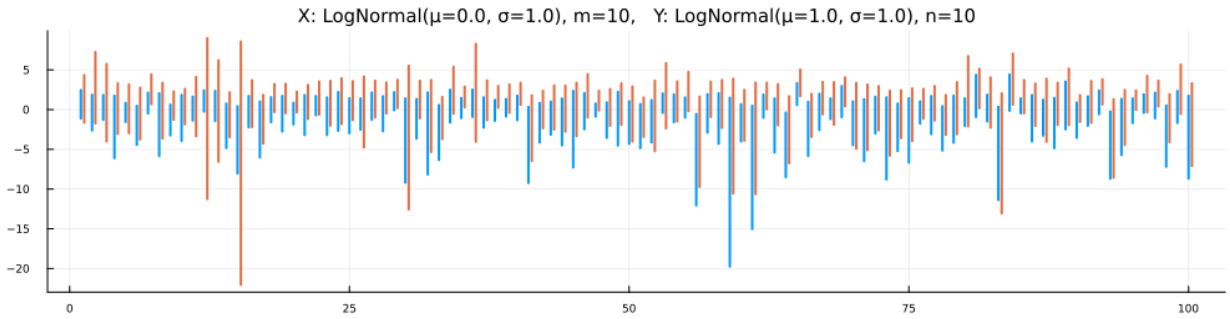
In [70]:
```
plot_confints(distx = Normal(0, 1), disty = Normal(0, 4), m = 160, n = 160)
```

Out[70]:



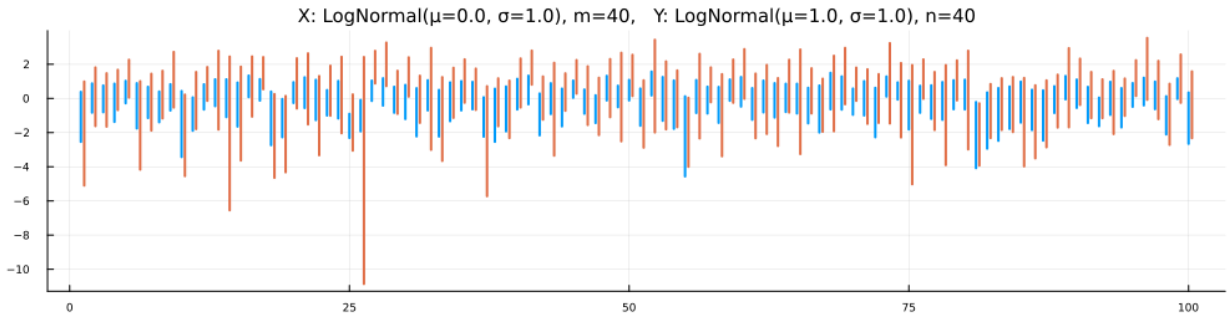
In [71]:
```
plot_confints(distx = LogNormal(0), disty = LogNormal(1), m = 10, n = 10)
```

Out[71]:



In [72]:
```
plot_confints(distx = LogNormal(0), disty = LogNormal(1), m = 40, n = 40)
```

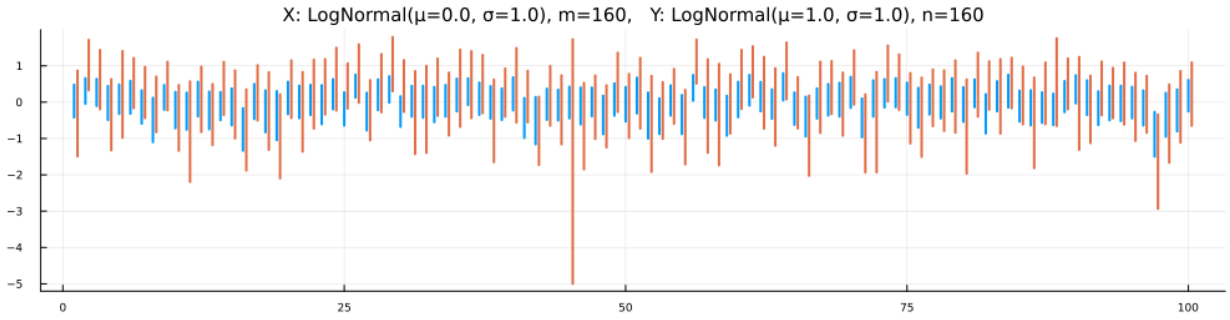
Out[72]:



Loading web-font STIX-Web/Size1/Regular

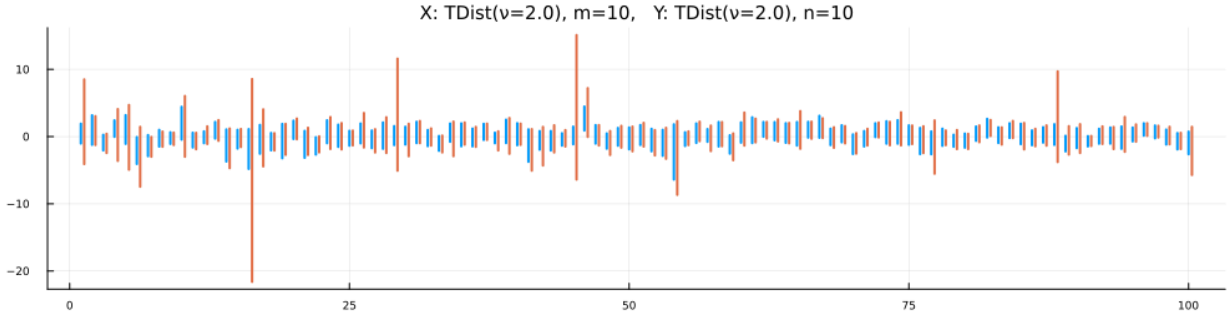
In [73]:
```
1 plot_confints(distx = LogNormal(0), disty = LogNormal(1), m = 160, n = 160)
```

Out[73]:



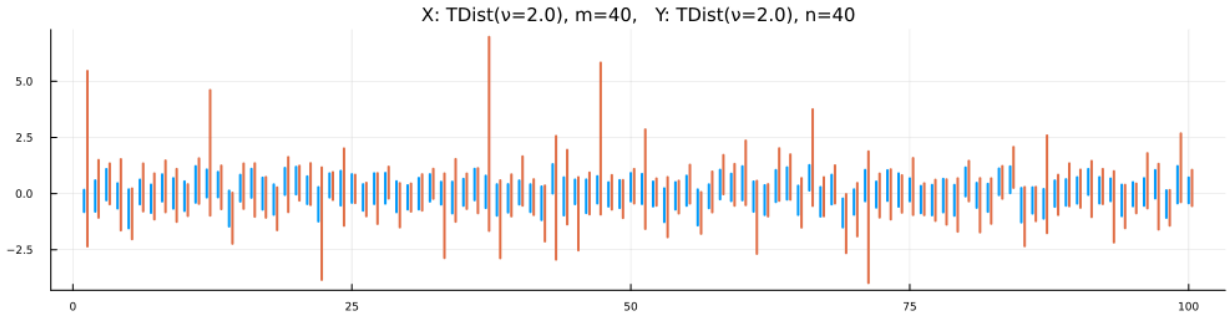
In [74]:
```
1 plot_confints(distx = TDist(2), disty = TDist(2), m = 10, n = 10)
```

Out[74]:



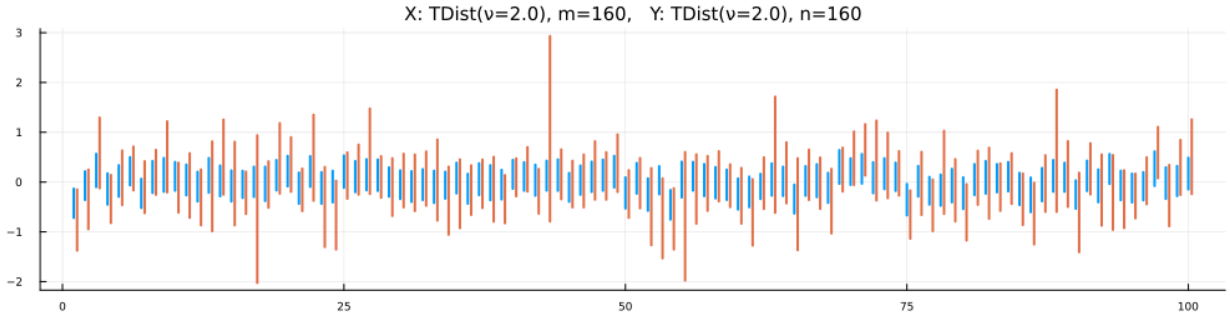
In [75]:
```
1 plot_confints(distx = TDist(2), disty = TDist(2), m = 40, n = 40)
```

Out[75]:



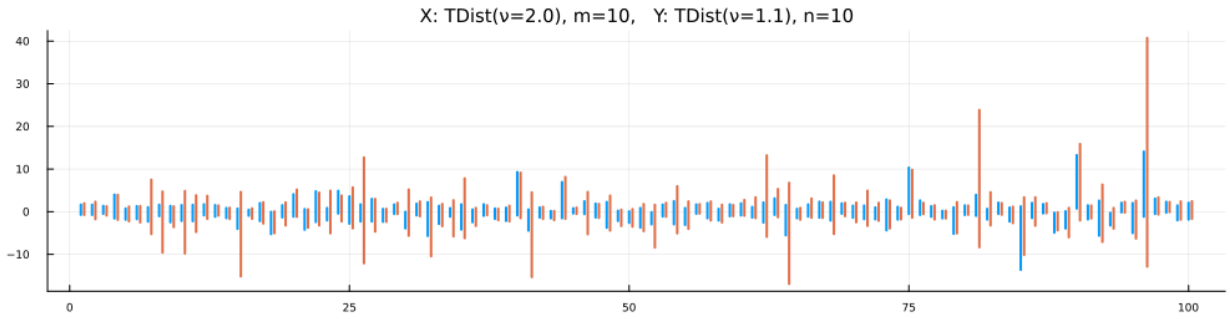
In [76]:
```
1 plot_confints(distx = TDist(2), disty = TDist(2), m = 160, n = 160)
```

Out[76]:



In [77]:
```
1 plot_confints(distx = TDist(2), disty = TDist(1.1), m = 10, n = 10)
```

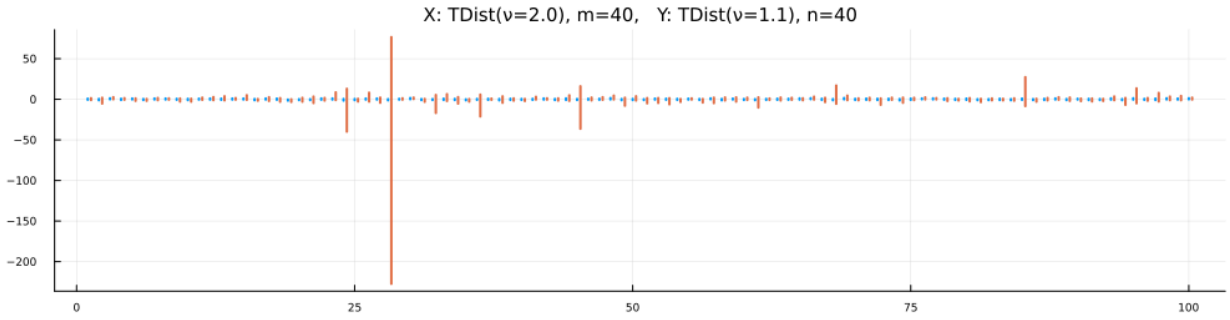
Out[77]:



Loading web-font STIX-Web/Size1/Regular

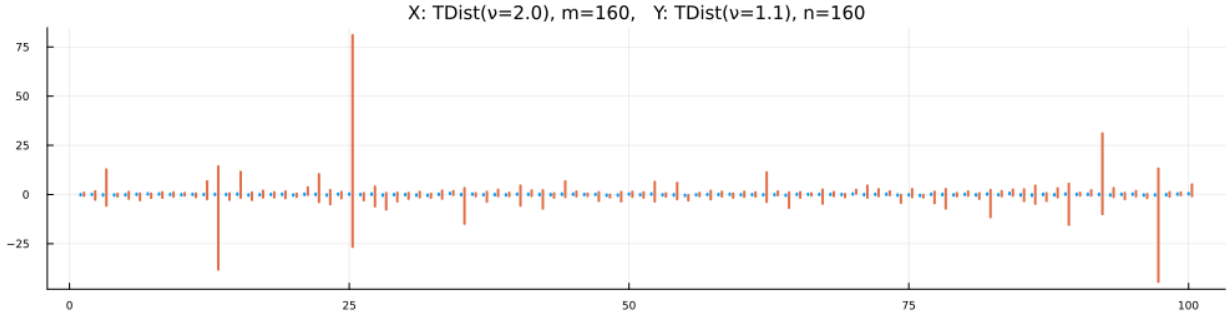
In [78]:
```
plot_confints(distx = TDist(2), disty = TDist(1.1), m = 40, n = 40)
```

Out[78]:



In [79]:
```
plot_confints(distx = TDist(2), disty = TDist(1.1), m = 160, n = 160)
```

Out[79]:



In [80]:
```
function plot_limits(;
        distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10,
        L = 1000, kwargs...)

    @show distx, m
    @show disty, n

    a = tieshift(distx, disty)
    Δμ = mean(distx) - mean(disty)

    BM = fill(zeros(2), 0)
    W = fill(zeros(2), 0)
    for _ in 1:L
        X = rand(distx, m)
        Y = rand(disty, n)
        push!(BM, brunner_munzel(X, Y .+ a).confint_shift)
        push!(W, confint_welch(X, Y .+ Δμ))
    end

    lower = [(first(BM[i]), first(W[i])) for i in 1:L]
    upper = [(last(BM[i]), last(W[i])) for i in 1:L]

    P1 = scatter(lower; label="", msc=:auto, ms=2, ma=0.5)
    plot!(identity; label="")
    plot!(xguide="Brunner-Munzel", yguide="Welch")
    title!("lower")

    P2 = scatter(upper; label="", msc=:auto, ms=2, ma=0.5)
    plot!(identity; label="")
    plot!(xguide="Brunner-Munzel", yguide="Welch")
    title!("upper")

    plot(P1, P2; size=(640, 320))
end
```

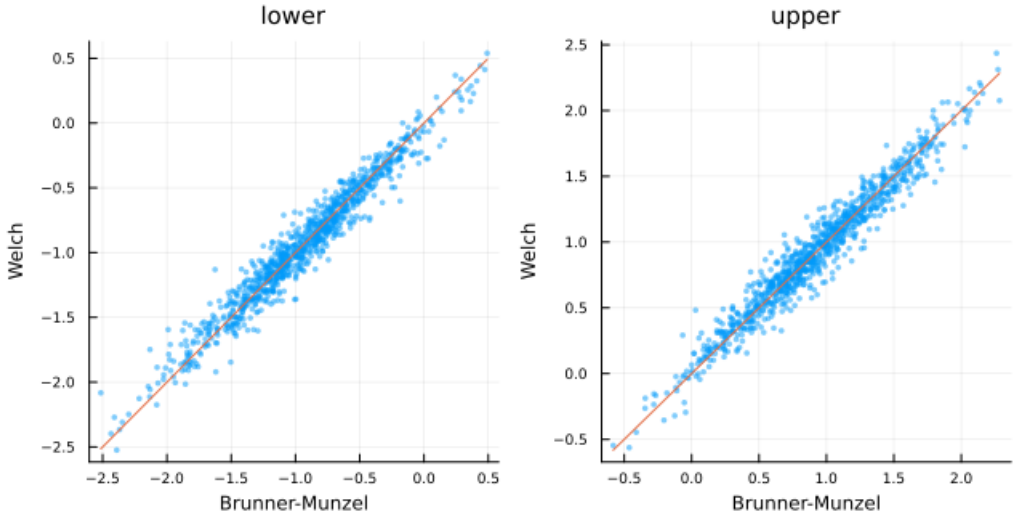
Out[80]:
```
plot_limits (generic function with 1 method)
```

Loading web-font STIX-Web/Size1/Regular

In [81]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 1), m = 10, n = 10)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 10)
(disty, n) = (Normal{Float64}(μ=0.0, σ=1.0), 10)
```

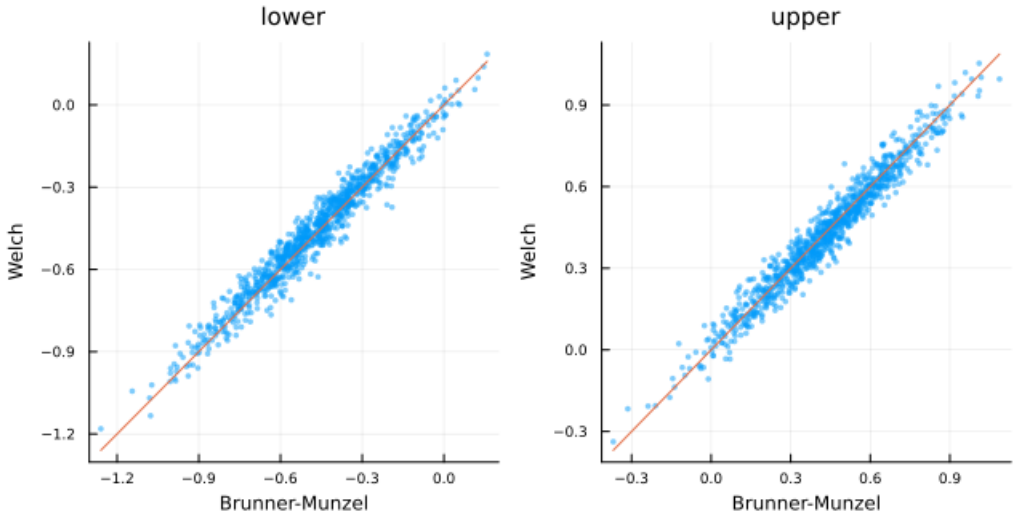
Out[81]:



In [82]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 1), m = 40, n = 40)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 40)
(disty, n) = (Normal{Float64}(μ=0.0, σ=1.0), 40)
```

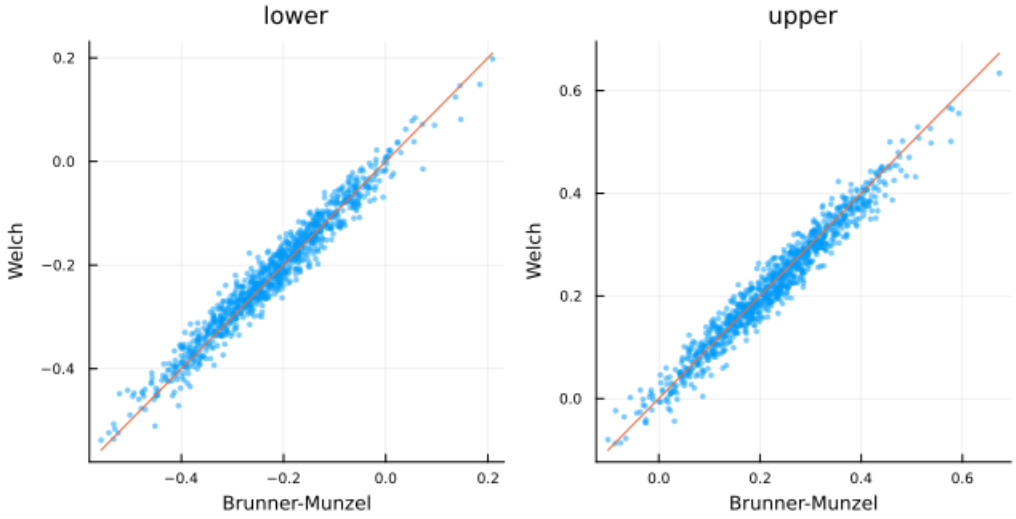
Out[82]:



In [83]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 1), m = 160, n = 160)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 160)
(disty, n) = (Normal{Float64}(μ=0.0, σ=1.0), 160)
```

Out[83]:

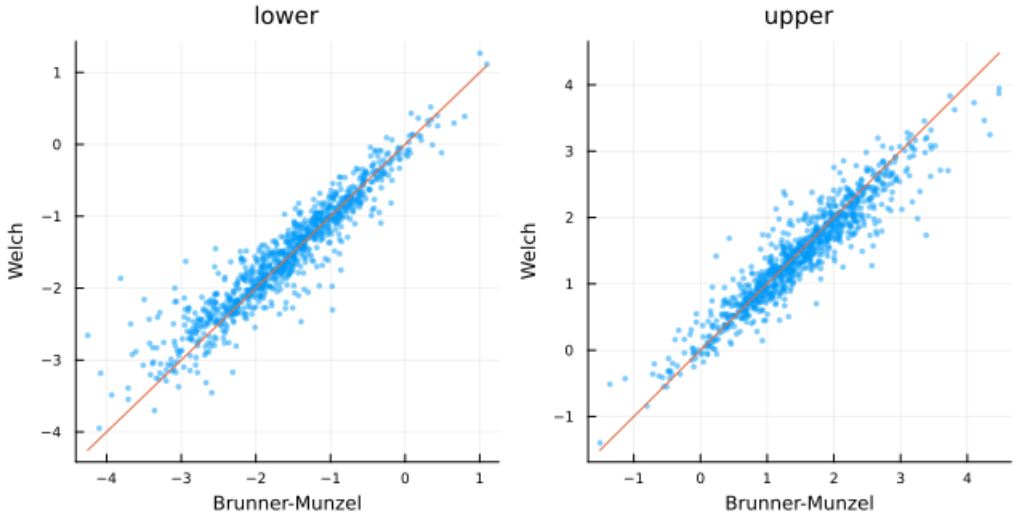


Loading web-font STIX-Web/Size1/Regular

In [84]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 10)
(disty, n) = (Normal{Float64}(μ=0.0, σ=2.0), 10)
```

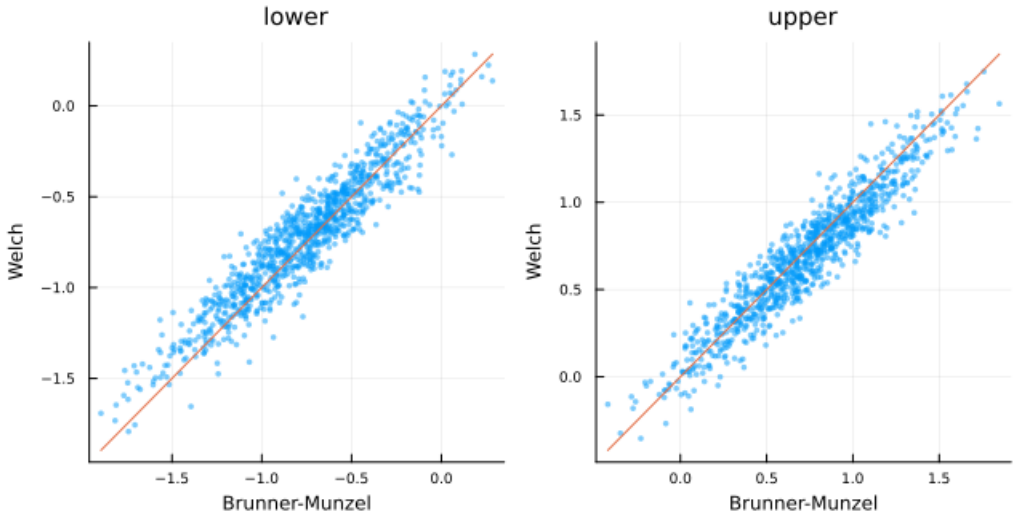
Out[84]:



In [85]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 2), m = 40, n = 40)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 40)
(disty, n) = (Normal{Float64}(μ=0.0, σ=2.0), 40)
```

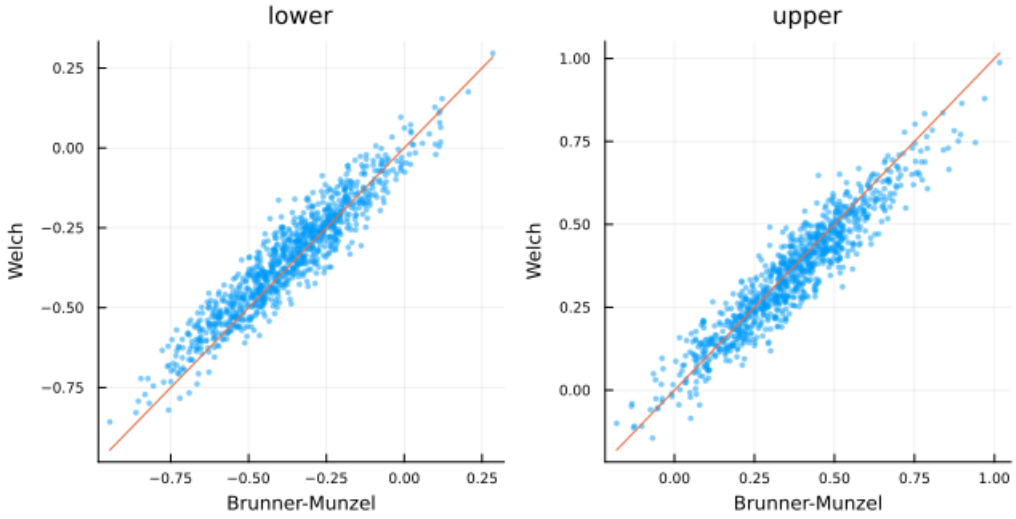
Out[85]:



In [86]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 2), m = 160, n = 160)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 160)
(disty, n) = (Normal{Float64}(μ=0.0, σ=2.0), 160)
```

Out[86]:

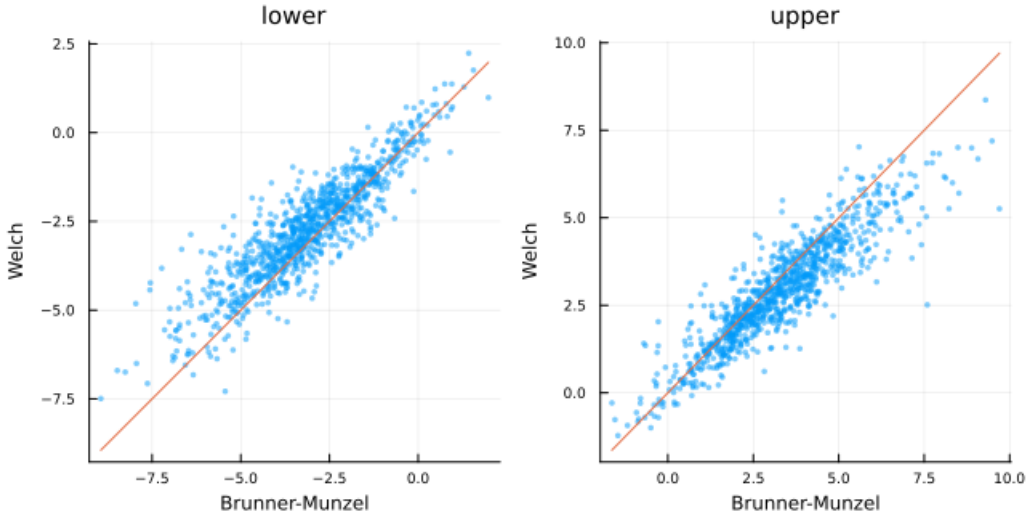


Loading web-font STIX-Web/Size1/Regular

In [87]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 4), m = 10, n = 10)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 10)
(disty, n) = (Normal{Float64}(μ=0.0, σ=4.0), 10)
```

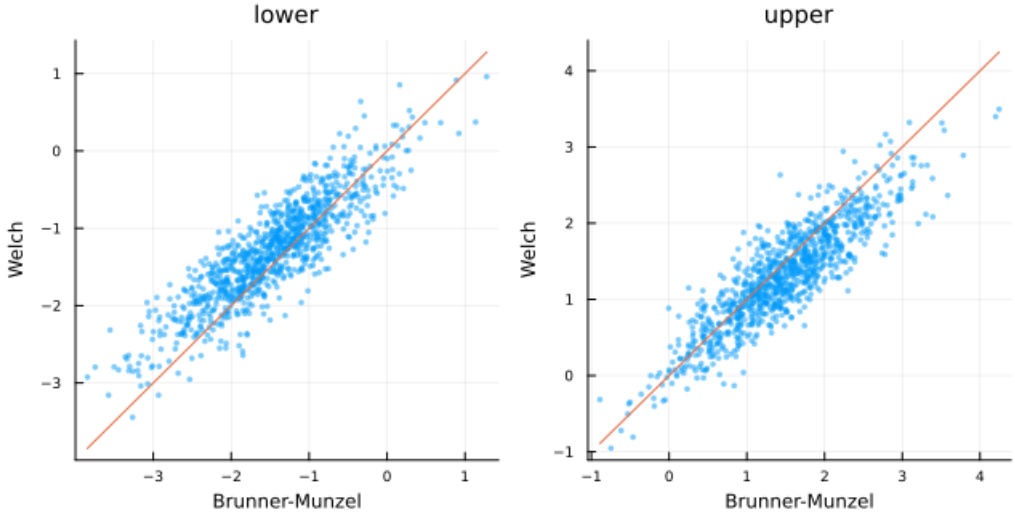
Out[87]:



In [88]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 4), m = 40, n = 40)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 40)
(disty, n) = (Normal{Float64}(μ=0.0, σ=4.0), 40)
```

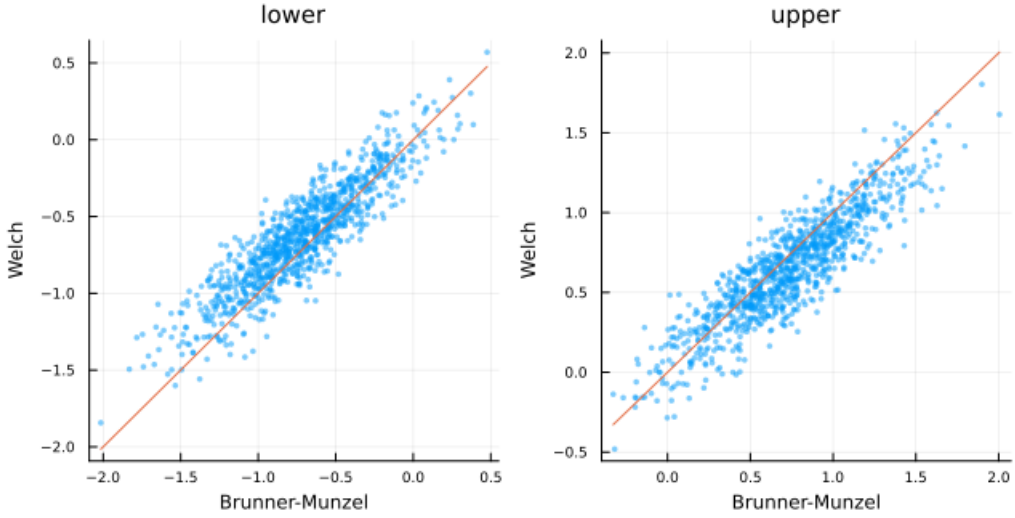
Out[88]:



In [89]:
```
1 plot_limits(distx = Normal(0, 1), disty = Normal(0, 4), m = 160, n = 160)
```

```
(distx, m) = (Normal{Float64}(μ=0.0, σ=1.0), 160)
(disty, n) = (Normal{Float64}(μ=0.0, σ=4.0), 160)
```

Out[89]:

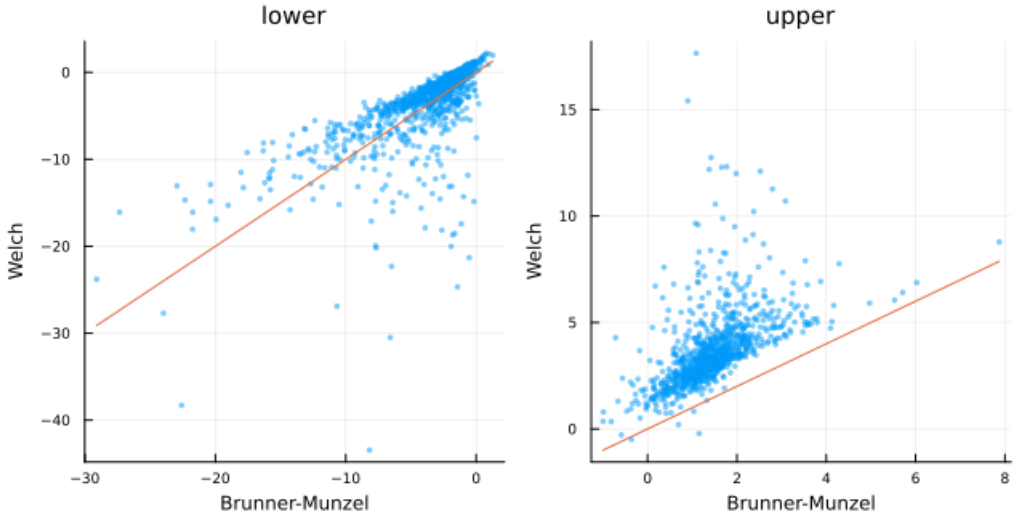


Loading web-font STIX-Web/Size1/Regular

In [90]:
```
1 plot_limits(distx = LogNormal(), disty = LogNormal(1), m = 10, n = 10)
```

```
(distx, m) = (LogNormal{Float64}(μ=0.0, σ=1.0), 10)
(disty, n) = (LogNormal{Float64}(μ=1.0, σ=1.0), 10)
```

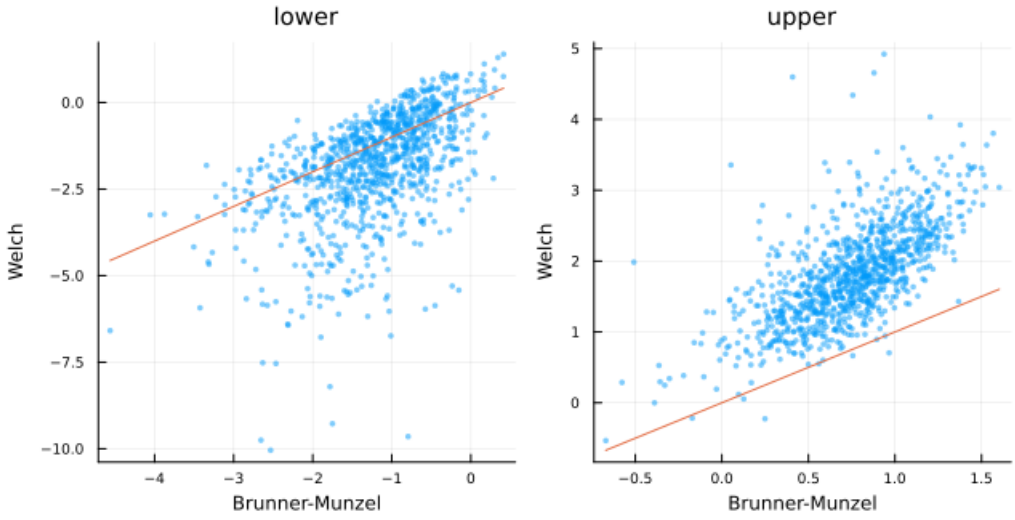
Out[90]:



In [91]:
```
1 plot_limits(distx = LogNormal(), disty = LogNormal(1), m = 40, n = 40)
```

```
(distx, m) = (LogNormal{Float64}(μ=0.0, σ=1.0), 40)
(disty, n) = (LogNormal{Float64}(μ=1.0, σ=1.0), 40)
```

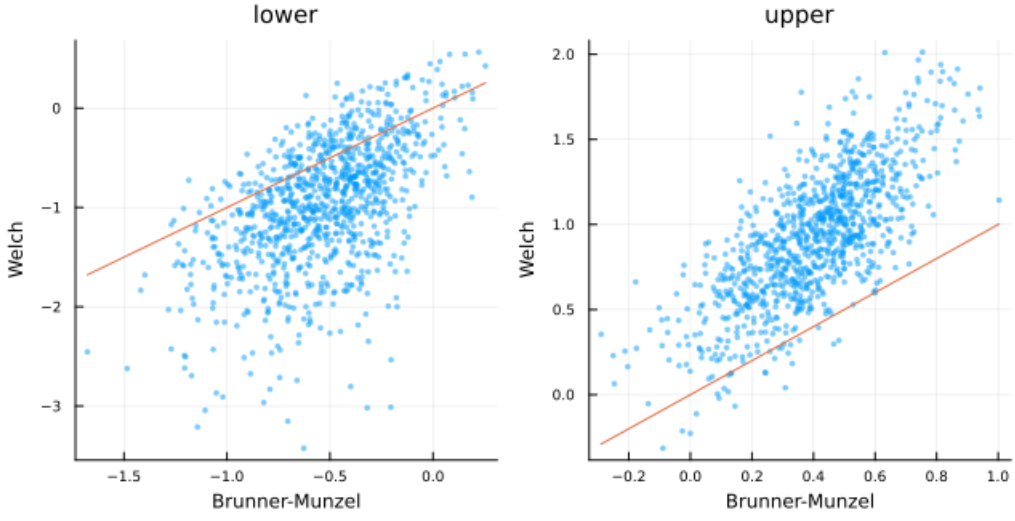
Out[91]:



In [92]:
```
1 @time plot_limits(distx = LogNormal(), disty = LogNormal(1), m = 160, n = 160)
```

```
(distx, m) = (LogNormal{Float64}(μ=0.0, σ=1.0), 160)
(disty, n) = (LogNormal{Float64}(μ=1.0, σ=1.0), 160)
  7.017568 seconds (284.24 k allocations: 108.661 MiB, 0.26% gc time)
```

Out[92]:

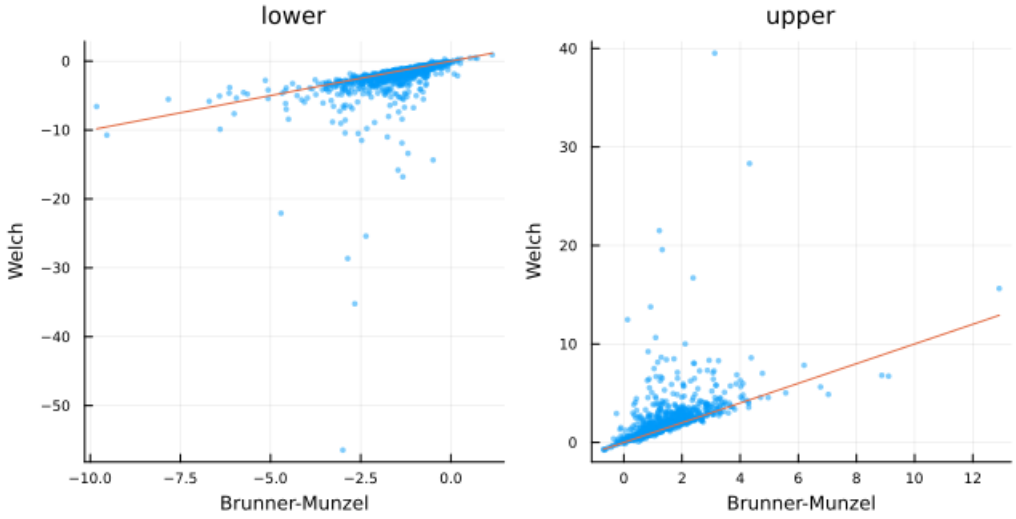


Loading web-font STIX-Web/Size1/Regular

In [93]:
```
1 plot_limits(distx = TDist(2), disty = TDist(2), m = 10, n = 10)
```

```
(distx, m) = (TDist{Float64}(ν=2.0), 10)
(disty, n) = (TDist{Float64}(ν=2.0), 10)
```

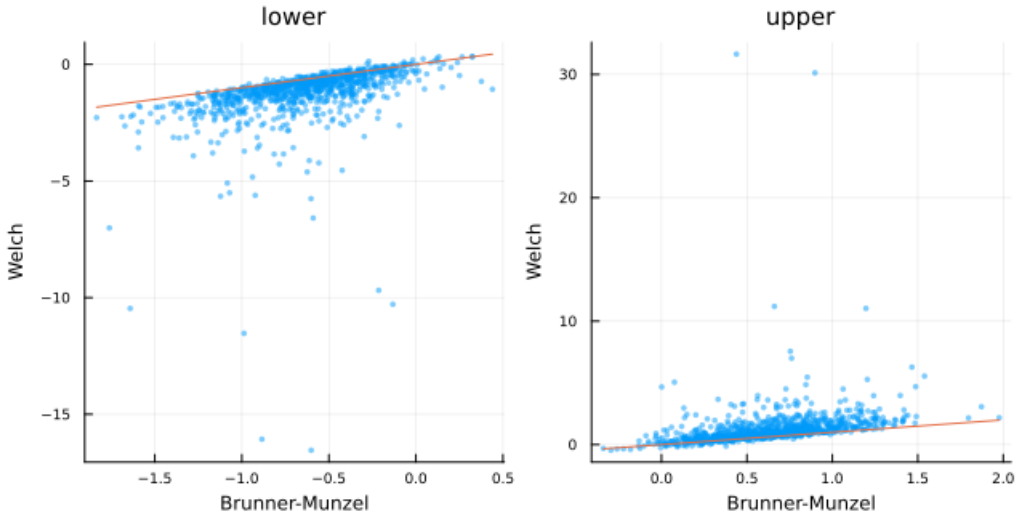
Out[93]:



In [94]:
```
1 plot_limits(distx = TDist(2), disty = TDist(2), m = 40, n = 40)
```

```
(distx, m) = (TDist{Float64}(ν=2.0), 40)
(disty, n) = (TDist{Float64}(ν=2.0), 40)
```

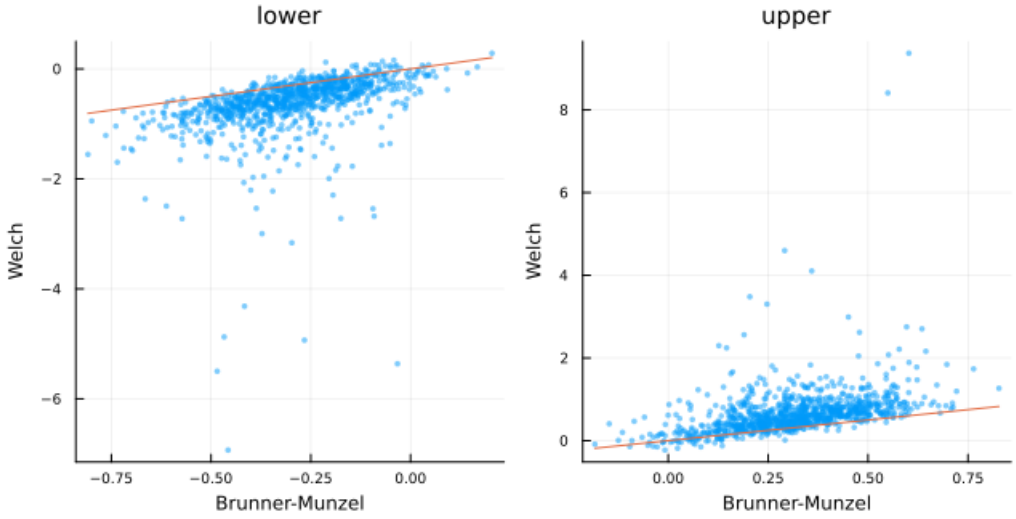
Out[94]:



In [95]:
```
1 plot_limits(distx = TDist(2), disty = TDist(2), m = 160, n = 160)
```

```
(distx, m) = (TDist{Float64}(ν=2.0), 160)
(disty, n) = (TDist{Float64}(ν=2.0), 160)
```

Out[95]:

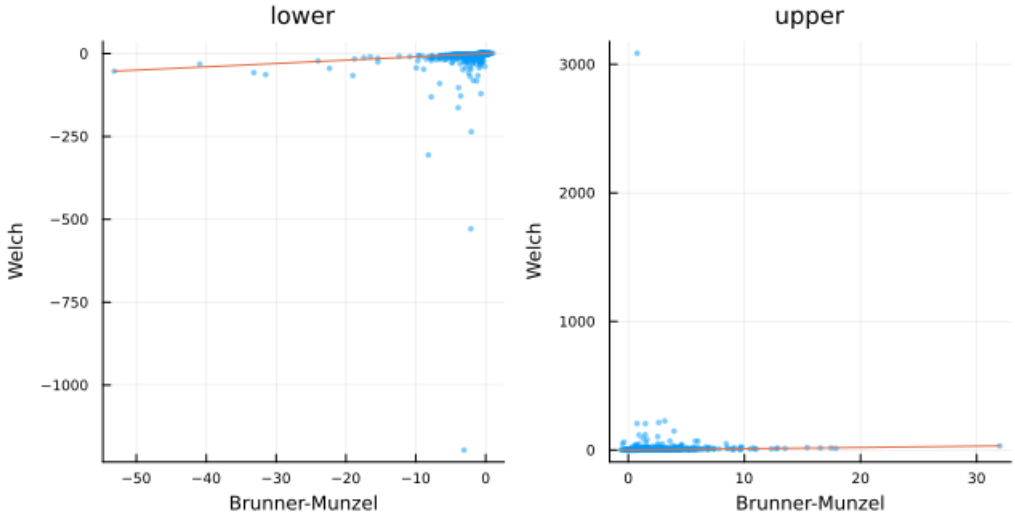


Loading web-font STIX-Web/Size1/Regular

In [96]:
```
1 plot_limits(distx = TDist(2), disty = TDist(1.1), m = 10, n = 10)
```

```
(distx, m) = (TDist{Float64}(ν=2.0), 10)
(disty, n) = (TDist{Float64}(ν=1.1), 10)
```

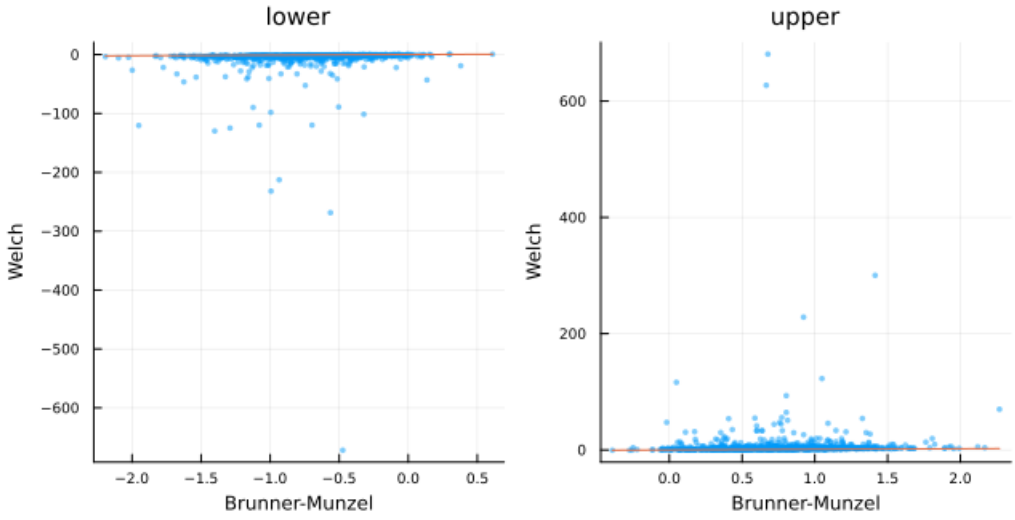
Out[96]:



In [97]:
```
1 plot_limits(distx = TDist(2), disty = TDist(1.1), m = 40, n = 40)
```

```
(distx, m) = (TDist{Float64}(ν=2.0), 40)
(disty, n) = (TDist{Float64}(ν=1.1), 40)
```

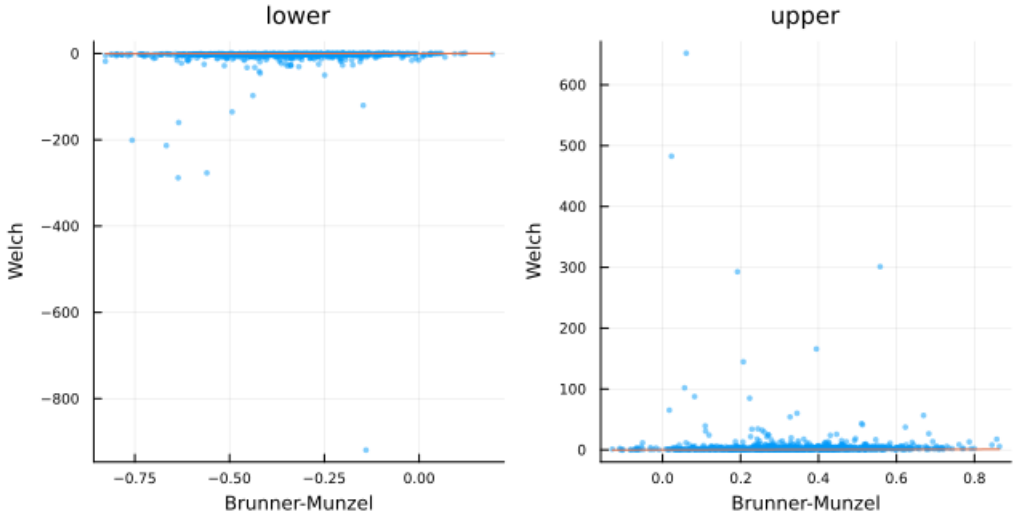
Out[97]:



In [98]:
```
1 plot_limits(distx = TDist(2), disty = TDist(1.1), m = 160, n = 160)
```

```
(distx, m) = (TDist{Float64}(ν=2.0), 160)
(disty, n) = (TDist{Float64}(ν=1.1), 160)
```

Out[98]:



Loading web-font STIX-Web/Size1/Regular

5 小サンプルでのpermutation版の検定とBM検定とWelchのt検定の比較

In [99]:
```
function sim_brunner_mumzel_perm(;
        distx = Normal(0, 1), disty = Normal(0, 2), m = 5, n = 5,
        L = 10^2)
    pval_bm_perm = Vector{Float64}(undef, L)
    tmpX = [Vector{Float64}(undef, m) for _ in 1:nthreads()]
    tmpY = [Vector{Float64}(undef, n) for _ in 1:nthreads()]
    tmpXandY = [Vector{Float64}(undef, m+n) for _ in 1:nthreads()]
    tmpTval = [Vector{Float64}(undef, binomial(m+n, m)) for _ in 1:nthreads()]
    tmpHx = [Vector{Float64}(undef, m) for _ in 1:nthreads()]
    tmpHy = [Vector{Float64}(undef, n) for _ in 1:nthreads()]
    tmpccomb = [Vector{Int}(undef, n) for _ in 1:nthreads()]
    @threads for i in 1:L
        tid = threadid()
        X = rand!(distx, tmpX[tid])
        Y = rand!(disty, tmpY[tid])
        Tval = permutation_tvalues_brunner_munzel(X, Y,
            tmpXandY[tid], tmpTval[tid], tmpHx[tid], tmpHy[tid], tmpccomb[tid])
        tval = statistics_brunner_munzel(X, Y, tmpHx[tid], tmpHy[tid]).tvalue
        pval_bm_perm[i] = pvalue_brunner_munzel_perm(X, Y, Tval, tval)
    end
    ecdf(pval_bm_perm)
end
```

Out[99]: sim_brunner_mumzel_perm (generic function with 1 method)

In [100]:
```
@time ecdf_bm_perm = sim_brunner_mumzel_perm(
    distx = Normal(0, 1), disty = Normal(0, 2), m = 7, n = 7, L = 10^4)
```

5.033259 seconds (16.27 M allocations: 4.602 GiB, 15.78% gc time, 9.61% compilation time)

Out[100]:
```
ECDF{Vector{Float64}, Weights{Float64, Float64, Vector{Float64}}}([0.0005827505827505828, 0.
0005827505827505828, 0.0005827505827505828, 0.0005827505827505828, 0.0005827505827505828, 0.
0005827505827505828, 0.0011655011655011655, 0.0011655011655011655, 0.0011655011655011655, 0.
0011655011655011655  …  1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0], Float64[])
```

In [101]:
```
function plot_pvals_with_perm(;
        distx = Normal(0, 1),
        disty = Normal(0, 2),
        m = 7,
        n = 7,
        L = 10^4,
        kwargs...
    )
    a = tieshift(distx, disty)
    @time ecdf_bm_perm = sim_brunner_mumzel_perm(; distx, disty = disty + a, m, n, L)
    @time ecdf_bm = sim_brunner_mumzel(; distx, disty = disty + a, m, n, L)
    Δμ = mean(distx) - mean(disty)
    @time ecdf_w = sim_welch(; distx, disty = disty + Δμ, m, n, L)
    @show a Δμ

    plot(legend=:topleft)
    plot!(α -> ecdf_bm_perm(α), 0, 0.1; label="BM permutation")
    plot!(α -> ecdf_bm(α), 0, 0.1; label="Brunner-Munzel", ls=:dash)
    plot!(α -> ecdf_w(α), 0, 0.1; label="Welch", ls=:dashdot)
    plot!(identity; label="", c=:black, ls=:dot)
    plot!(xtick=0:0.01:0.1, ytick=0:0.01:1)
    plot!(xguide="nominal significance level α",
        yguide="probability of P-value < α")
    a_ = string(round(a; digits=4))
    Δμ_ = string(round(Δμ; digits=4))
    title!("X: $(distname(distx)), m=$m\n\
        Y: $(distname(disty))+(a, Δμ), n=$n\n\
        a=$a_, Δμ=$Δμ_")
    plot!(size=(400, 450), titlefontsize=9)
    plot!(; kwargs...)
end
```

Out[101]: plot_pvals_with_perm (generic function with 1 method)

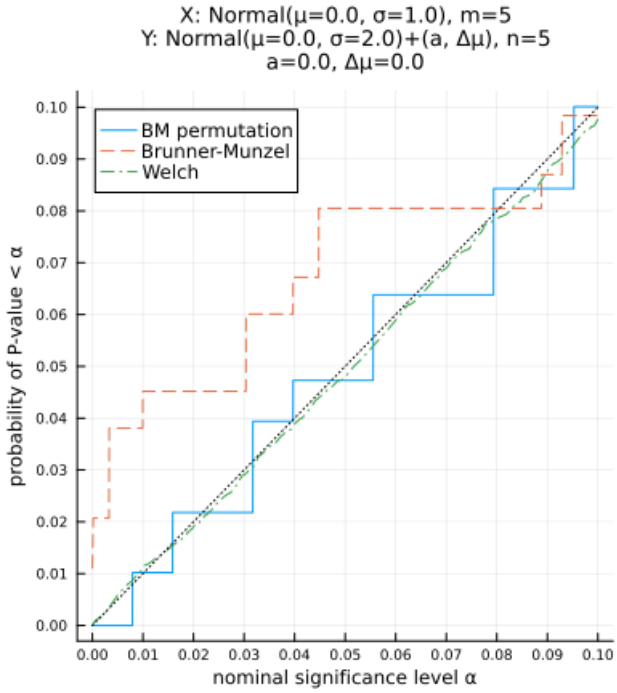
Loading web-font STIX-Web/Size1/Regular

In [102]:

```
plot_pvals_with_perm(
    distx = Normal(0, 1), disty = Normal(0, 2), m = 5, n = 5, L = 10^4)
```

```
  0.363926 seconds (1.22 M allocations: 350.587 MiB, 29.71% gc time)
  0.024036 seconds (4.97 k allocations: 1.724 MiB)
  0.001940 seconds (5.94 k allocations: 2.009 MiB)
a = 7.685641860444171e-14
Δμ = 0.0
```

Out[102]:

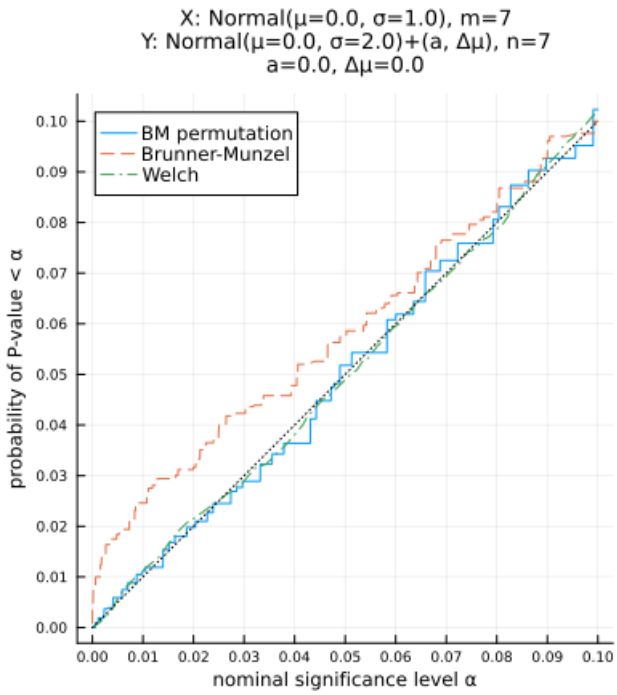


In [103]:

```
plot_pvals_with_perm(
    distx = Normal(0, 1), disty = Normal(0, 2), m = 7, n = 7, L = 10^4)
```

```
  5.505243 seconds (16.26 M allocations: 4.601 GiB, 16.43% gc time)
  0.017051 seconds (5.07 k allocations: 1.753 MiB)
  0.001551 seconds (5.75 k allocations: 1.955 MiB)
a = 7.685641860444171e-14
Δμ = 0.0
```

Out[103]:



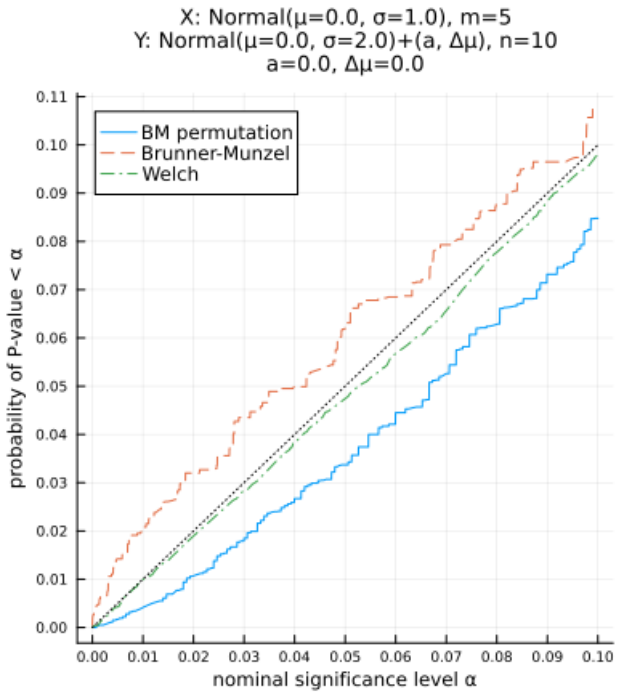
Loading web-font STIX-Web/Size1/Regular

In [104]:

```
1 plot_pvals_with_perm(
2     distx = Normal(0, 1), disty = Normal(0, 2), m = 5, n = 10, L = 10^4)
```

```
  4.904059 seconds (15.70 M allocations: 4.442 GiB, 15.94% gc time)
  0.002604 seconds (5.22 k allocations: 1.797 MiB)
  0.011408 seconds (5.65 k allocations: 1.925 MiB)
a = 7.685641860444171e-14
Δμ = 0.0
```

Out[104]:

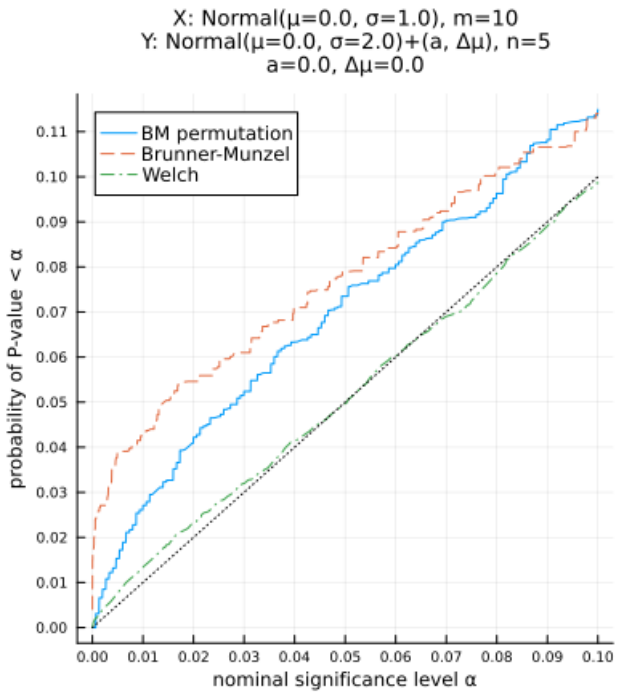


In [105]:

```
1 plot_pvals_with_perm(
2     distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 5, L = 10^4)
```

```
  4.785355 seconds (15.70 M allocations: 4.443 GiB, 15.10% gc time)
  0.001891 seconds (5.21 k allocations: 1.794 MiB)
  0.002406 seconds (6.05 k allocations: 2.042 MiB)
a = 7.685641860444171e-14
Δμ = 0.0
```

Out[105]:



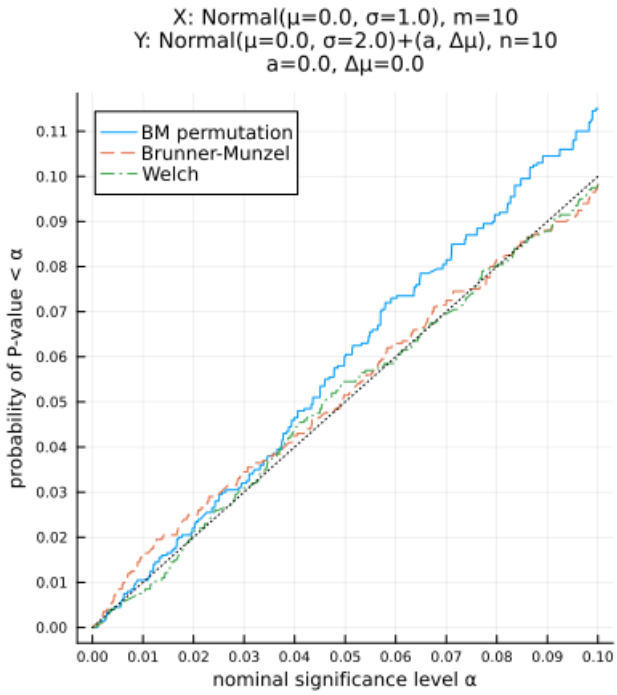
Loading web-font STIX-Web/Size1/Regular

In [106]:

```
plot_pvals_with_perm(
    distx = Normal(0, 1), disty = Normal(0, 2), m = 10, n = 10, L = 2000)
```

```
 67.435797 seconds (203.34 M allocations: 57.586 GiB, 16.83% gc time)
  0.021784 seconds (1.18 k allocations: 391.758 KiB)
  0.015124 seconds (1.31 k allocations: 432.234 KiB)
a = 7.685641860444171e-14
Δμ = 0.0
```

Out[106]:

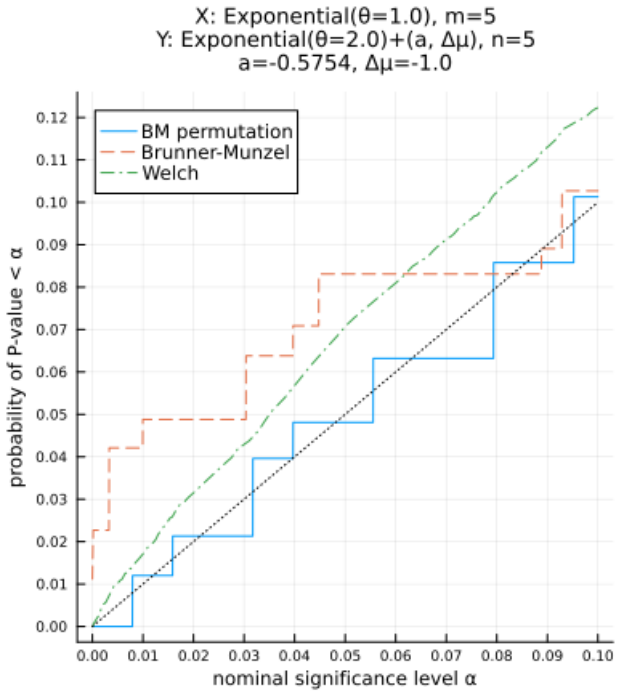


In [107]:

```
plot_pvals_with_perm(
    distx = Exponential(1), disty = Exponential(2),
    m = 5, n = 5, L = 10^4)
```

```
  0.343144 seconds (1.23 M allocations: 351.582 MiB, 221.71% compilation time)
  0.035708 seconds (14.38 k allocations: 2.305 MiB, 1145.51% compilation time)
  0.030994 seconds (14.93 k allocations: 2.644 MiB, 1141.61% compilation time)
a = -0.5753641445892759
Δμ = -1.0
```

Out[107]:



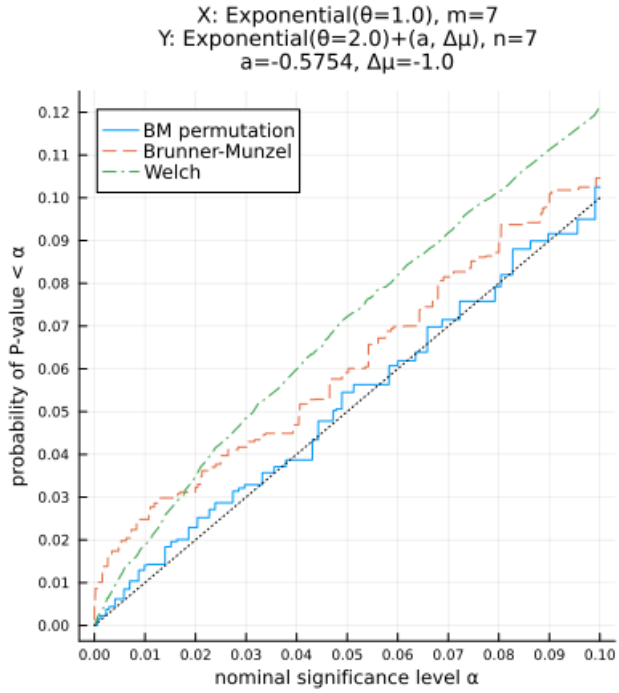
Loading web-font STIX-Web/Size1/Regular

In [108]:

```
plot_pvals_with_perm(
    distx = Exponential(1), disty = Exponential(2),
    m = 7, n = 7, L = 10^4)
```

```
  5.384772 seconds (16.26 M allocations: 4.601 GiB, 16.60% gc time)
  0.020699 seconds (5.12 k allocations: 1.768 MiB)
  0.002418 seconds (5.96 k allocations: 2.017 MiB)
a = -0.5753641445892759
Δμ = -1.0
```

Out[108]:

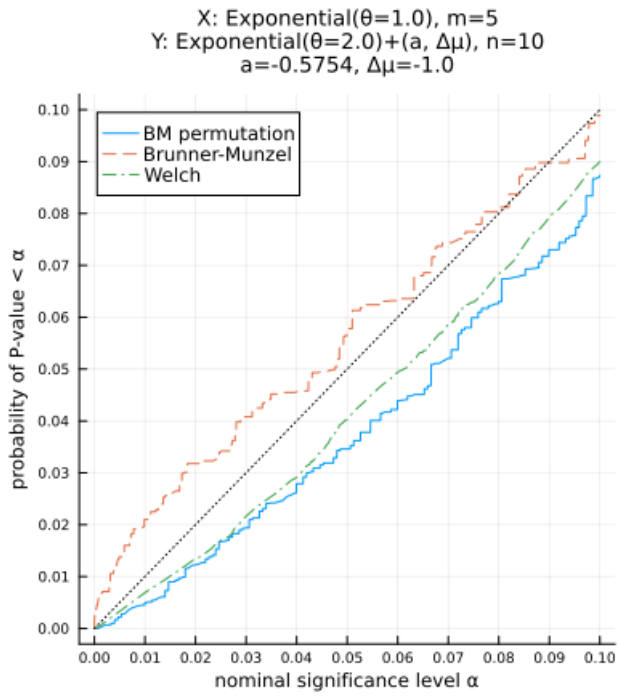


In [109]:

```
plot_pvals_with_perm(
    distx = Exponential(1), disty = Exponential(2),
    m = 5, n = 10, L = 10^4)
```

```
  4.838760 seconds (15.70 M allocations: 4.442 GiB, 17.10% gc time)
  0.001775 seconds (5.13 k allocations: 1.769 MiB)
  0.001546 seconds (5.77 k allocations: 1.960 MiB)
a = -0.5753641445892759
Δμ = -1.0
```

Out[109]:



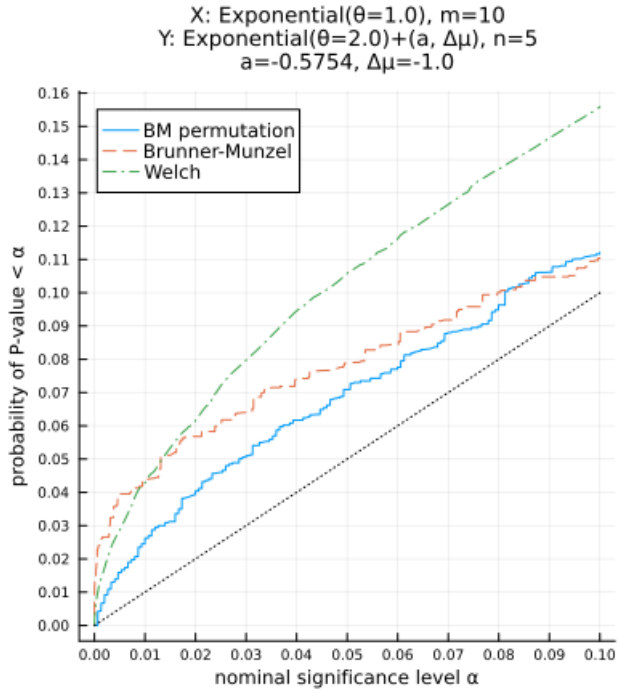
Loading web-font STIX-Web/Size1/Regular

In [110]:

```
plot_pvals_with_perm(
    distx = Exponential(1), disty = Exponential(2),
    m = 10, n = 5, L = 10^4)
```

```
  4.919973 seconds (15.70 M allocations: 4.443 GiB, 18.50% gc time)
  0.015176 seconds (5.34 k allocations: 1.829 MiB)
  0.002276 seconds (6.43 k allocations: 2.149 MiB)
a = -0.5753641445892759
Δμ = -1.0
```

Out[110]:

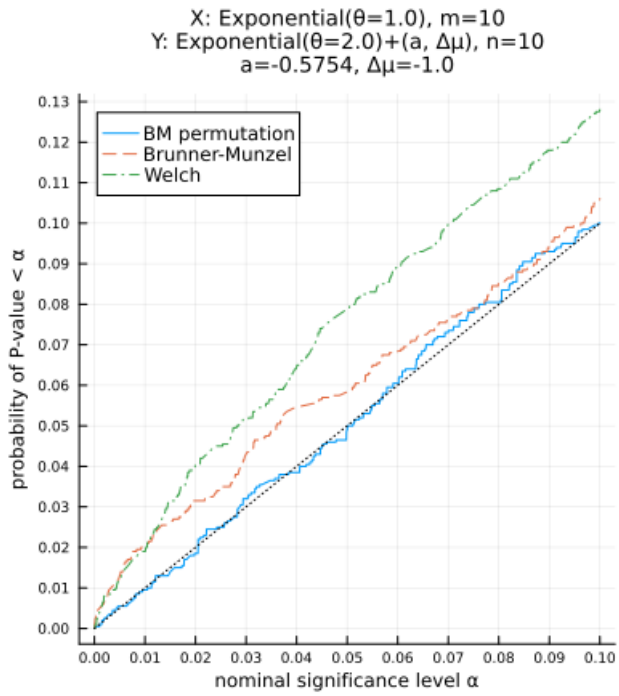


In [111]:

```
plot_pvals_with_perm(
    distx = Exponential(1), disty = Exponential(2),
    m = 10, n = 10, L = 2000)
```

```
 66.374139 seconds (203.34 M allocations: 57.586 GiB, 16.94% gc time)
  0.000406 seconds (1.14 k allocations: 382.258 KiB)
  0.000322 seconds (1.27 k allocations: 422.703 KiB)
a = -0.5753641445892759
Δμ = -1.0
```

Out[111]:



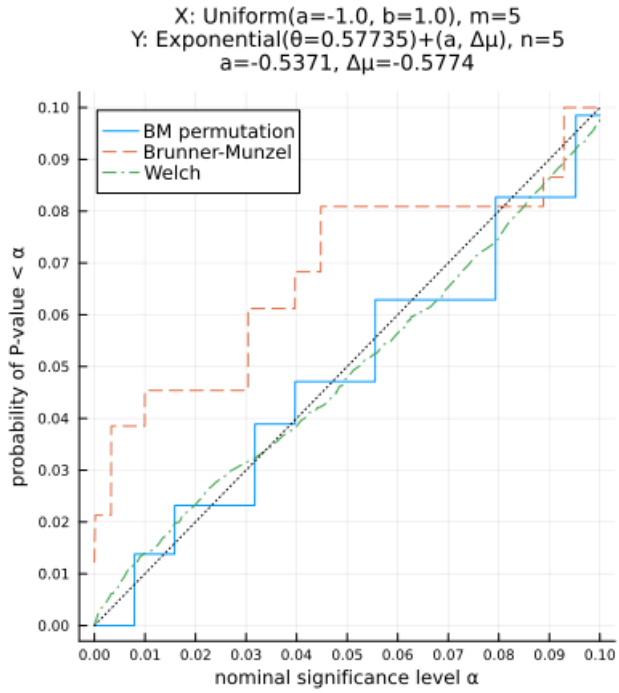
Loading web-font STIX-Web/Size1/Regular

In [112]:

```
plot_pvals_with_perm(
    distx = Uniform(-1, 1), disty = Exponential(0.5773502691896257),
    m = 5, n = 5, L = 10^4)
```

```
  0.369498 seconds (1.23 M allocations: 351.212 MiB, 17.51% gc time, 105.53% compilation tim
e)
  0.024506 seconds (4.75 k allocations: 1.658 MiB)
  0.056172 seconds (14.33 k allocations: 2.446 MiB, 1154.43% compilation time)
a = -0.5370568188698567
Δμ = -0.5773502691896257
```

Out[112]:

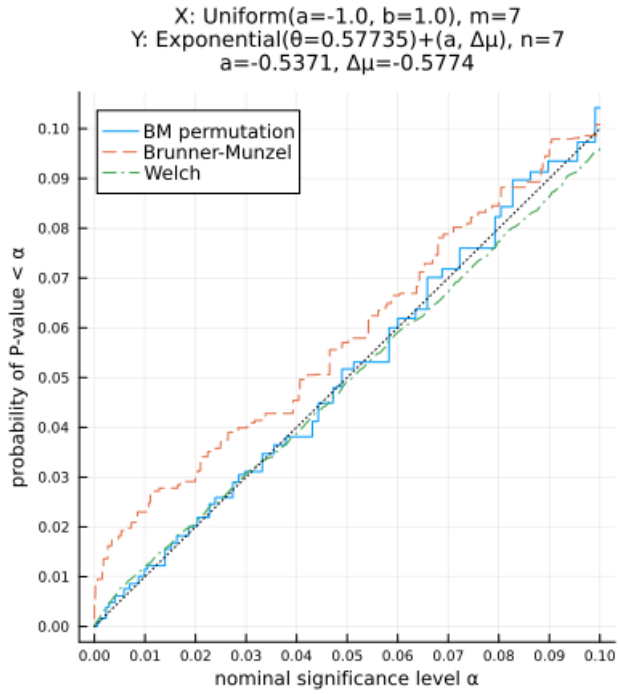


In [113]:

```
plot_pvals_with_perm(
    distx = Uniform(-1, 1), disty = Exponential(0.5773502691896257),
    m = 7, n = 7, L = 10^4)
```

```
  5.337259 seconds (16.25 M allocations: 4.601 GiB, 17.09% gc time)
  0.001646 seconds (4.90 k allocations: 1.703 MiB)
  0.018274 seconds (5.44 k allocations: 1.864 MiB)
a = -0.5370568188698567
Δμ = -0.5773502691896257
```

Out[113]:



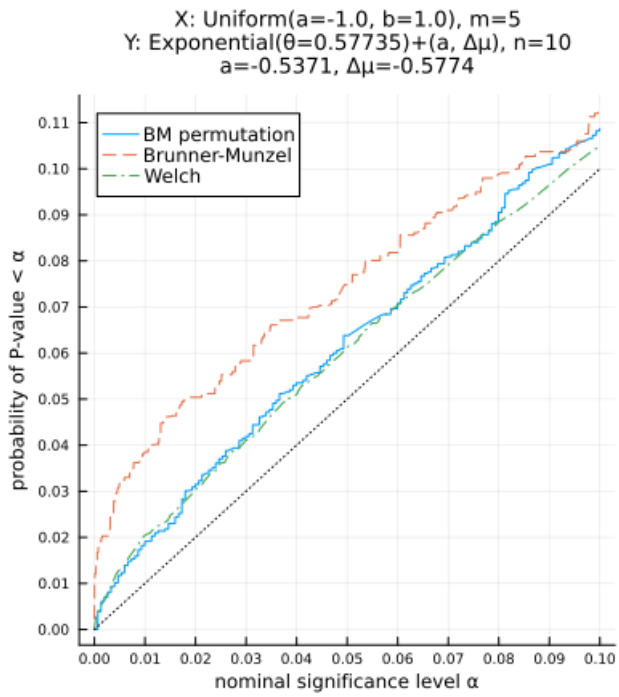
Loading web-font STIX-Web/Size1/Regular

In [114]:

```
plot_pvals_with_perm(
    distx = Uniform(-1, 1), disty = Exponential(0.5773502691896257),
    m = 5, n = 10, L = 10^4)
```

```
  4.874172 seconds (15.70 M allocations: 4.442 GiB, 18.91% gc time)
  0.002489 seconds (5.45 k allocations: 1.862 MiB)
  0.001980 seconds (5.71 k allocations: 1.943 MiB)
a = -0.5370568188698567
Δμ = -0.5773502691896257
```

Out[114]:

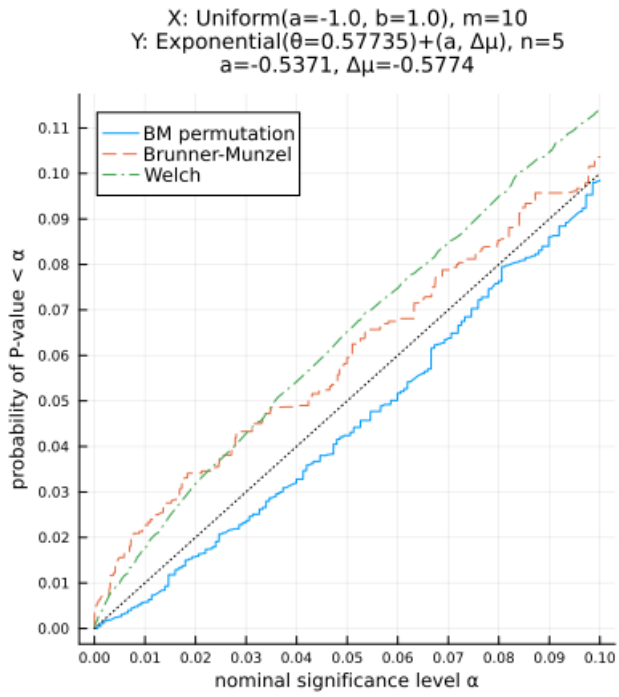


In [115]:

```
plot_pvals_with_perm(
    distx = Uniform(-1, 1), disty = Exponential(0.5773502691896257),
    m = 10, n = 5, L = 10^4)
```

```
  4.875233 seconds (15.70 M allocations: 4.443 GiB, 15.06% gc time)
  0.001710 seconds (5.34 k allocations: 1.832 MiB)
  0.001474 seconds (6.12 k allocations: 2.059 MiB)
a = -0.5370568188698567
Δμ = -0.5773502691896257
```

Out[115]:



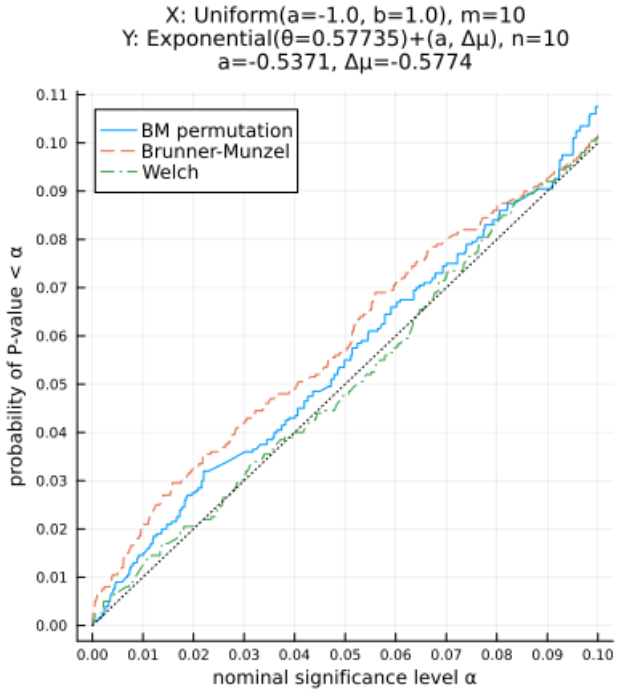
Loading web-font STIX-Web/Size1/Regular

In [116]:

```
plot_pvals_with_perm(
    distx = Uniform(-1, 1), disty = Exponential(0.5773502691896257),
    m = 10, n = 10, L = 2000)
```

```
 66.621568 seconds (203.34 M allocations: 57.586 GiB, 17.00% gc time)
  0.000423 seconds (1.14 k allocations: 380.953 KiB)
  0.000337 seconds (1.30 k allocations: 432.703 KiB)
a = -0.5370568188698567
Δμ = -0.5773502691896257
```

Out[116]:

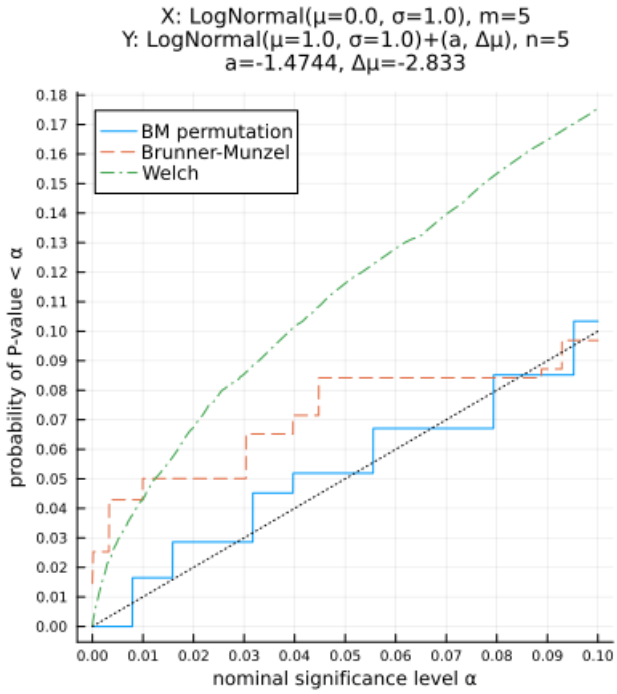


In [117]:

```
plot_pvals_with_perm(
    distx = LogNormal(), disty = LogNormal(1), m = 5, n = 5, L = 10^4)
```

```
  0.356817 seconds (1.23 M allocations: 351.341 MiB, 154.41% compilation time)
  0.002541 seconds (4.76 k allocations: 1.663 MiB)
  0.002543 seconds (6.17 k allocations: 2.077 MiB)
a = -1.4744426128871542
Δμ = -2.8329677996379363
```

Out[117]:



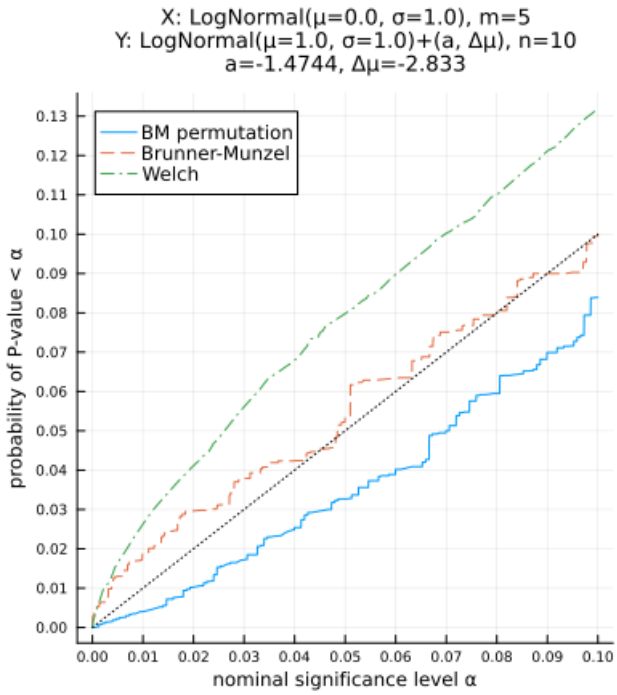
Loading web-font STIX-Web/Size1/Regular

In [118]:

```
plot_pvals_with_perm(
    distx = LogNormal(), disty = LogNormal(1), m = 5, n = 10, L = 10^4)
```

```
  4.881429 seconds (15.70 M allocations: 4.442 GiB, 17.95% gc time)
  0.026256 seconds (5.26 k allocations: 1.808 MiB)
  0.015550 seconds (5.63 k allocations: 1.921 MiB)
a = -1.4744426128871542
Δμ = -2.8329677996379363
```

Out[118]:

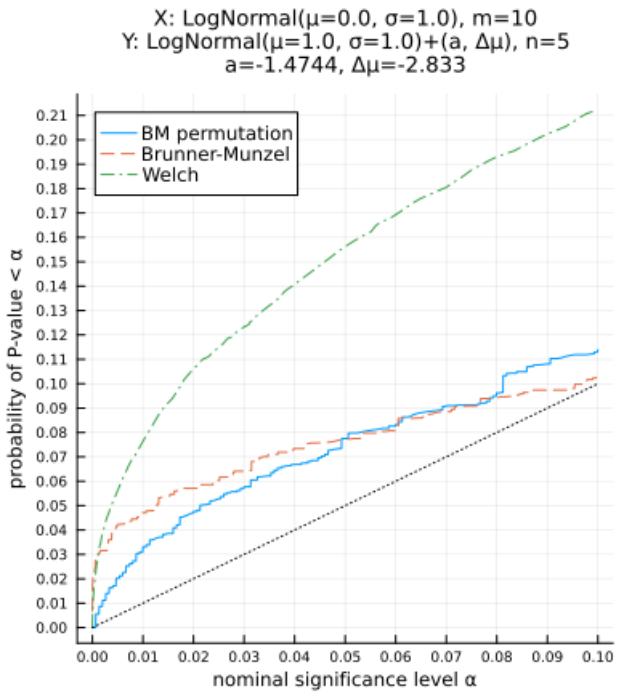


In [119]:

```
plot_pvals_with_perm(
    distx = LogNormal(), disty = LogNormal(1), m = 10, n = 5, L = 10^4)
```

```
  4.735019 seconds (15.70 M allocations: 4.443 GiB, 15.23% gc time)
  0.013115 seconds (5.25 k allocations: 1.805 MiB)
  0.015421 seconds (6.43 k allocations: 2.153 MiB)
a = -1.4744426128871542
Δμ = -2.8329677996379363
```

Out[119]:



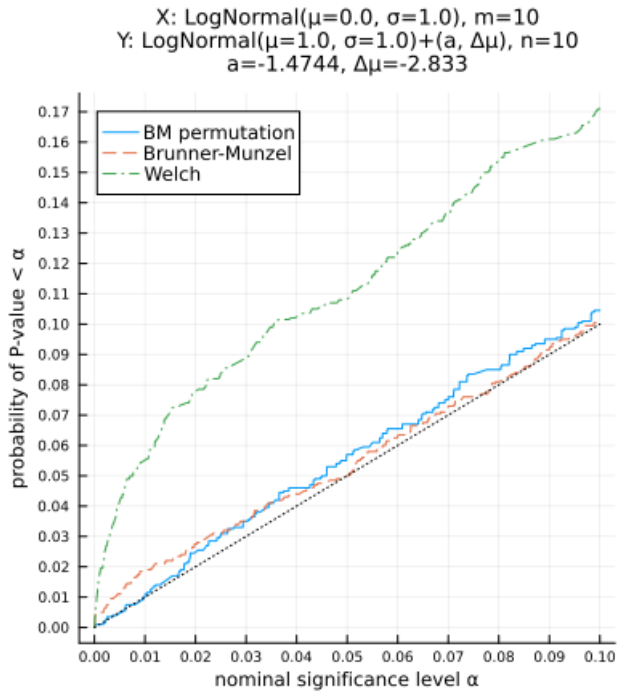
Loading web-font STIX-Web/Size1/Regular

In [120]:

```
plot_pvals_with_perm(
    distx = LogNormal(0), disty = LogNormal(1), m = 10, n = 10, L = 2000)
```

```
 68.285465 seconds (203.34 M allocations: 57.586 GiB, 16.88% gc time)
  0.000460 seconds (1.13 k allocations: 377.789 KiB)
  0.000395 seconds (1.28 k allocations: 426.453 KiB)
a = -1.4744426128871542
Δμ = -2.8329677996379363
```

Out[120]:



In [ ]:

```

```

Loading web-font STIX-Web/Size1/Regular