See chapter 9 in Regression and Other Stories.

```
html"""
<style>
    main {
        margin: 0 auto;
        max-width: 2000px;
        padding-left: max(160px, 10%);
        padding-right: max(160px, 10%);
    }
</style>
"""

```

```
using Pkg ✓ , DrWatson ✓
```

```
begin
    # Specific to this notebook
    using GLM ✓

    # Specific to ROSTuringPluto
    using Optim ✓
    using Logging ✓
    using Turing ✓

    # Graphics related
    using GLMakie ✓

    # Common data files and functions
    using RegressionAndOtherStories ✓
    import RegressionAndOtherStories: link

    Logging.disable_logging(Logging.Warn)
end;
```

```
Replacing docs for `RegressionAndOtherStories.tr
DataFrame, AbstractString}` in module `Regressic
```

9.1 Propagating uncertainty in inference using posterior simulations.

hibbs =

	year	growth	vote	inc_party_candidat
1	1952	2.4	44.6	"Stevenson"
2	1956	2.89	57.76	"Eisenhower"
3	1960	0.85	49.91	"Nixon"
4	1964	4.21	61.34	"Johnson"
5	1968	3.02	49.6	"Humphrey"
6	1972	3.62	61.79	"Nixon"
7	1976	1.08	48.95	"Ford"
8	1980	-0.39	44.7	"Carter"
9	1984	3.86	59.17	"Reagan"
10	1988	2.27	53.94	"Bush, Sr."
⋮ more				
16	2012	0.95	52.0	"Obama"

```
hibbs =
CSV.read(ros_datadir("ElectionsEconomy",
"hibbs.csv"), DataFrame)
```

UndefVarError: hibbs_lm not defined

1. top-level scope @ *Local: 18*

```
let
    fig = Figure()
    hibbs.label = string.(hibbs.year)
    xlabel = "Average growth personal
    income [%]"
    ylabel = "Incumbent's party vote share"
    let
        title = "Forecasting the election
        from the economy"
        ax = Axis(fig[1, 1]; title, xlabel,
        ylabel)
        for (ind, yr) in
        enumerate(hibbs.year)
            annotations!("$(yr)"; position=
            (hibbs.growth[ind],
            hibbs.vote[ind]), textsize=10)
        end
    end
    let
        x = LinRange(-1, 4, 100)
        title = "Data and linear fit"
        ax = Axis(fig[1, 2]; title, xlabel,
        ylabel)
        scatter!(hibbs.growth, hibbs.vote)
        lines!(x, coef(hibbs_lm)[1] .+
        coef(hibbs_lm)[2] .* x;
        color=:darkred)
        annotations!("vote = 46.2 + 3.0 *
        growth"; position=(0, 41))
    end
    fig
end
```

ppl7_1 (generic function with 2 methods)

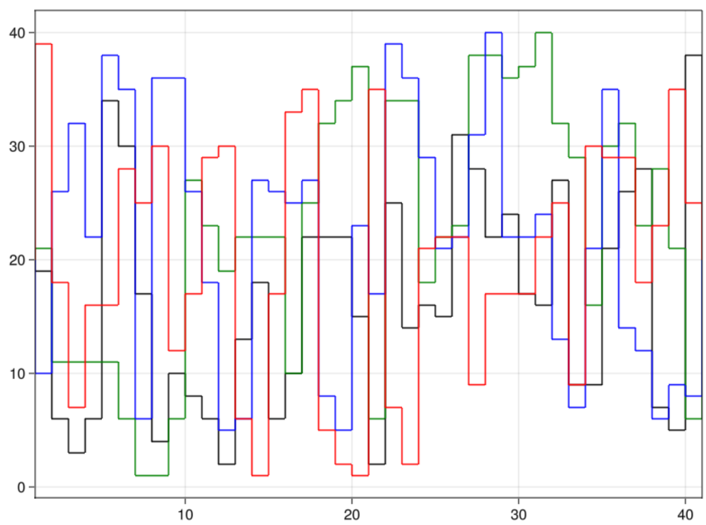
```
@model function ppl7_1(x, y)
    a ~ Normal(50, 20)
    b ~ Normal(2, 10)
    σ ~ Exponential(1)
    μ = a .+ b .* x
    for i in eachindex(y)
        y[i] ~ Normal(μ[i], σ)
    end
end
```

	parameters	mean	std	naive_se
1	:a	46.2708	1.55608	0.0246038
2	:b	3.05463	0.663403	0.0104893
3	:σ	3.57383	0.620971	0.0098184

```
begin
    m7_1t = ppl7_1(hibbs.growth, hibbs.vote)
    chns7_1t = sample(m7_1t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns7_1t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	46.273	1.554	46.271	1.55
2	"b"	3.049	0.66	3.055	0.66

```
begin
    post7_1t = DataFrame(chns7_1t)[:, 3:5]
    ms7_1t = model_summary(post7_1t, [:a,
    :b, :sigma])
end
```



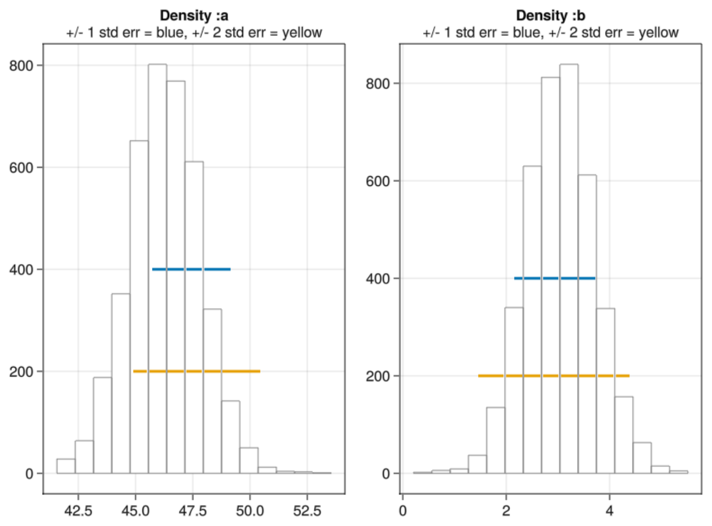

```
trankplot(post7_1t, "b")
```

```
sims = 4000×3 Matrix{Float64}:
        46.9655  2.92837  4.19874
        48.2228  2.33301  2.98444
        48.531   2.08068  2.57154
        48.4329  2.33537  3.76951
        45.4511  3.68536  2.99814
        43.5968  3.45706  3.77984
        45.6401  2.84532  3.23996
         ⋮
        46.0638  3.33481  3.41148
        44.1948  3.35263  3.12854
        45.7716  2.73001  4.02867
        47.5026  2.25395  3.93305
        45.5372  3.39224  3.18888
        48.2702  2.82634  2.85088
```

```
sims = Array(post7_1t)
```

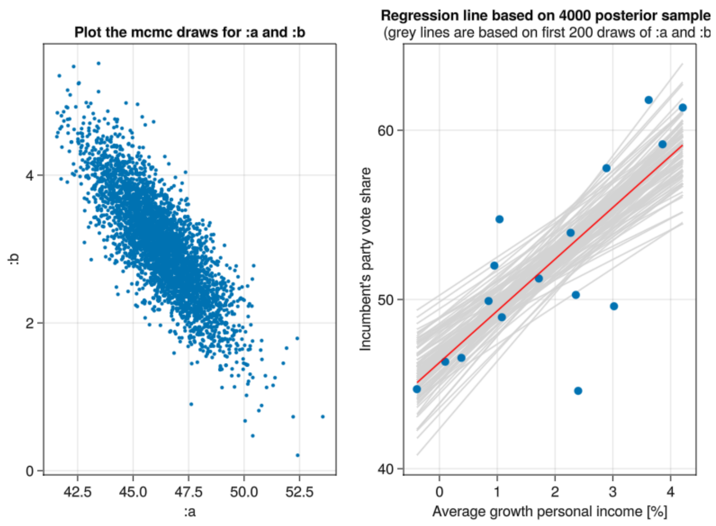
```
1×3 Matrix{Float64}:
 46.2726  3.04927  3.50379
```

```
median(sims; dims=1)
```




```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Density :a",
    subtitle="+/- 1 std err = blue, +/- 2
    std err = yellow")
    hist!(post7_1t.a; bins=15, color =
    :white, strokewidth = 1, strokecolor =
    :grey)
    hlines!(ax, 400; xmin=0.36, xmax=0.62,
    linewidth=3)
    hlines!(ax, 200; xmin=0.30, xmax=0.72,
    linewidth=3)

    ax = Axis(f[1, 2]; title="Density :b",
    subtitle="+/- 1 std err = blue, +/- 2
    std err = yellow")
    hist!(post7_1t.b; bins=15, color =
    :white, strokewidth = 1, strokecolor =
    :grey)
    hlines!(ax, 400; xmin=0.38, xmax=0.65,
    linewidth=3)
    hlines!(ax, 200; xmin=0.26, xmax=0.76,
    linewidth=3)
    f
end
```




```
let
    growth_range =
    LinRange(minimum(hibbs.growth),
    maximum(hibbs.growth), 200)
    votes = mean.(link(post7_1t, (r,x) ->
    r.a + x * r.b, growth_range))

    xlabel = "Average growth personal
    income [%]"
    ylabel = "Incumbent's party vote share"

    fig = Figure()

    ax = Axis(fig[1, 1]; title="Plot the
    mcmc draws for :a and :b", xlabel=":a",
    ylabel=":b")
    scatter!(post7_1t.a, post7_1t.b;
    markersize=4)

    xlabel = "Average growth personal
    income [%]"
    ylabel="Incumbent's party vote share"
    ax = Axis(fig[1, 2]; title="Regression
    line based on 4000 posterior samples",
        subtitle = "(grey lines are based
        on first 200 draws of :a and :b)",
        xlabel, ylabel)
    for i in 1:100
        lines!(growth_range, post7_1t.a[i]
        .+ post7_1t.b[i] .* growth_range,
        color = :lightgrey)
    end
    scatter!(hibbs.growth, hibbs.vote)
    lines!(growth_range, votes, color =
    :red)
    fig
```

```
end
```

9.2 Prediction and uncertainty.

	x	y
1	-2.0	50
2	-1.0	44
3	0.0	50
4	1.0	47
5	2.0	56

```
let
    x = LinRange(-2, 2, 5)
    y = [50, 44, 50, 47, 56]
    global sexratio = DataFrame(x = x, y =
    y)
end
```

ppl9_1 (generic function with 2 methods)

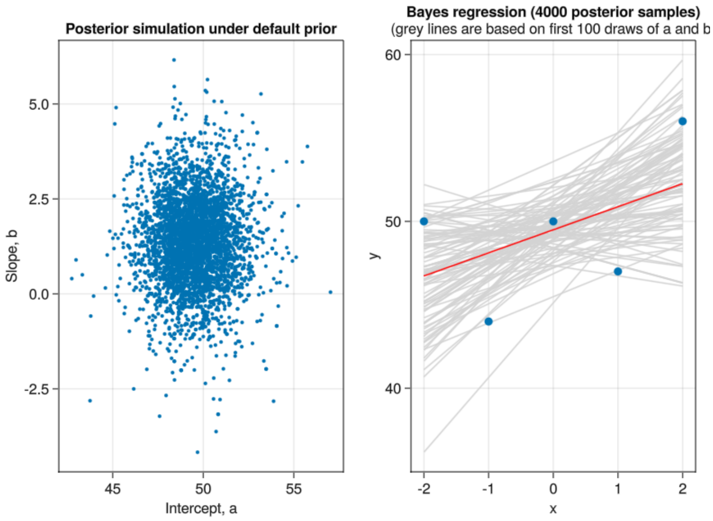
```
@model function ppl9_1(x, y)
    a ~ Normal(50, 5)
    b ~ Normal(0, 5)
    σ ~ Exponential(1)
    μ = a .+ b .* x
    for i in eachindex(y)
        y[i] ~ Normal(μ[i], σ)
    end
end
```

	parameters	mean	std	naive_se
1	:a	49.4958	1.50676	0.0238239
2	:b	1.38353	1.13322	0.0179178
3	:σ	3.51317	1.00488	0.0158886

```
begin
    m9_1t = ppl9_1(sexratio.x, sexratio.y)
    chns9_1t = sample(m9_1t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns9_1t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	49.491	1.413	49.496	1.50
2	"b"	1.397	1.058	1.384	1.13

```
begin
    post9_1t = DataFrame(chns9_1t)[:, 3:5]
    ms9_1t = model_summary(post9_1t, [:a,
    :b, :sigma])
end
```




```
let
    x_range = LinRange(minimum(sexratio.x),
    maximum(sexratio.x), 200)
    y = mean.(link(post9_1t, (r,x) -> r.a +
    x * r.b, x_range))

    xlabel = "x"
    ylabel = "y"

    fig = Figure()

    ax = Axis(fig[1, 1]; title="Posterior
    simulation under default prior",
    xlabel="Intercept, a", ylabel="Slope,
    b")
    scatter!(post9_1t.a, post9_1t.b;
    markersize=4)

    ax = Axis(fig[1, 2]; title="Bayes
    regression (4000 posterior samples)",
        subtitle = "(grey lines are based
        on first 100 draws of a and b)",
        xlabel, ylabel)
    for i in 1:100
        lines!(x_range, post9_1t.a[i] .+
        post9_1t.b[i] .* x_range, color =
        :lightgrey)
    end
    scatter!(sexratio.x, sexratio.y)
    lines!(x_range, y, color = :red)
    fig
end
```

ppl9_2 (generic function with 2 methods)

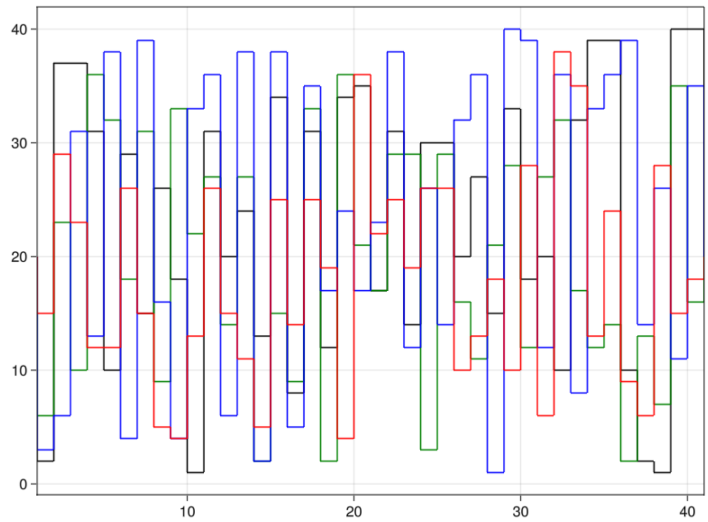
```
@model function ppl9_2(x, y)
    a ~ Normal(48.8, 0.2)
    b ~ Normal(0, 0.2)
    σ ~ Exponential(1)
    μ = a .+ b .* x
    for i in eachindex(y)
        y[i] ~ Normal(μ[i], σ)
    end
end
```

	parameters	mean	std	naive_s
1	:a	48.8133	0.197691	0.003125
2	:b	0.0509051	0.198607	0.003140
3	:σ	3.57855	0.922843	0.014591

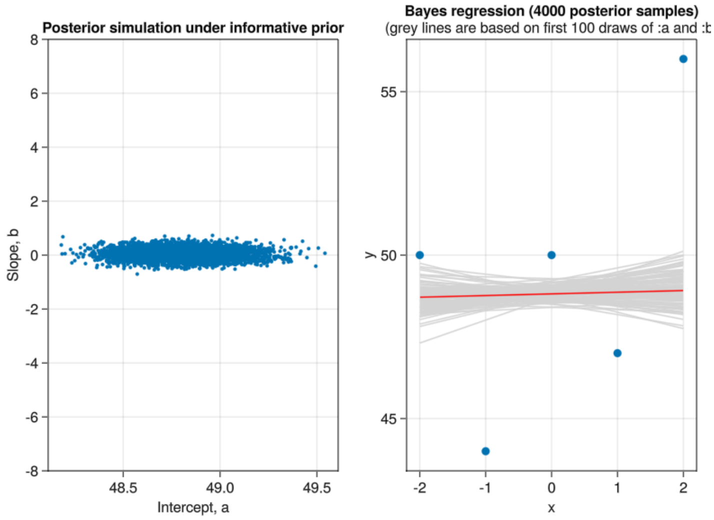
```
begin
    m9_2t = ppl9_2(sexratio.x, sexratio.y)
    chns9_2t = sample(m9_2t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns9_2t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	48.816	0.198	48.813	0.19
2	"b"	0.052	0.199	0.051	0.19

```
begin
    post9_2t = DataFrame(chns9_2t)[:, 3:5]
    ms9_12 = model_summary(post9_2t, [:a,
    :b, :sigma])
end
```




```
trankplot(post9_2t, "b")
```




```
let
    x_range = LinRange(minimum(sexratio.x),
    maximum(sexratio.x), 200)
    y = mean.(link(post9_2t, (r,x) -> r.a +
    x * r.b, x_range))

    xlabel = "x"
    ylabel = "y"

    fig = Figure()

    ax = Axis(fig[1, 1]; title="Posterior
    simulation under informative prior",
    xlabel="Intercept, a", ylabel="Slope,
    b")
    ylims!(ax, -8, 8)
    scatter!(post9_2t.a, post9_2t.b;
    markersize=4)

    ax = Axis(fig[1, 2]; title="Bayes
    regression (4000 posterior samples)",
        subtitle = "(grey lines are based
        on first 100 draws of :a and :b)",
        xlabel, ylabel)
    for i in 1:100
        lines!(x_range, post9_2t.a[i] .+
        post9_2t.b[i] .* x_range, color =
        :lightgrey)
    end
    scatter!(sexratio.x, sexratio.y)
    lines!(x_range, y, color = :red)
    fig
end
```

9.3 Prior information and Bayesian synthesis.

Prior based on a previously-fitted model using economic and political condition.

```
begin
    theta_hat_prior = 0.524
    se_prior = 0.041
end;
```

Survey of 400 people, of whom 190 say they will vote for the Democratic candidate.

```
begin
    n = 400
    y = 190
end;
```

Data estimate.

```
theta_hat_data = 0.475
```

```
theta_hat_data = y/n
```

```
se_data = 0.02496873044429772
```

```
se_data = √((y/n)*(1-y/n)/n)
```

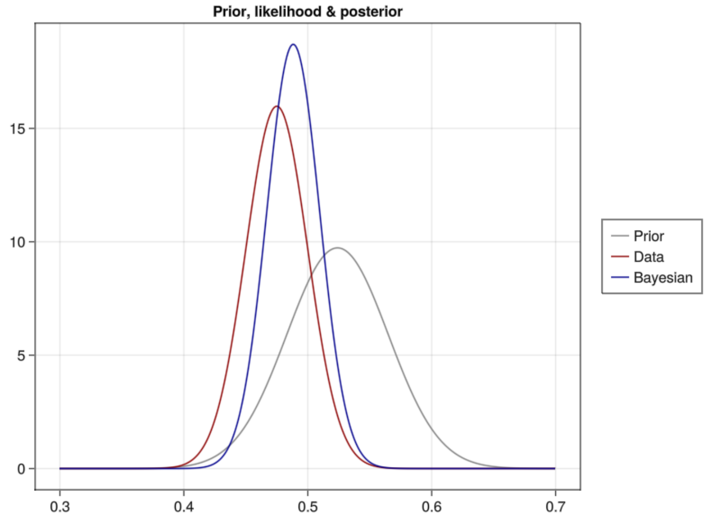
Bayes estimate.

```
theta_hat_bayes = 0.48825635323153693
```

```
theta_hat_bayes =
(theta_hat_prior/se_prior^2 +
    theta_hat_data/se_data^2) /(1/se_prior^2
     + 1/se_data^2)
```

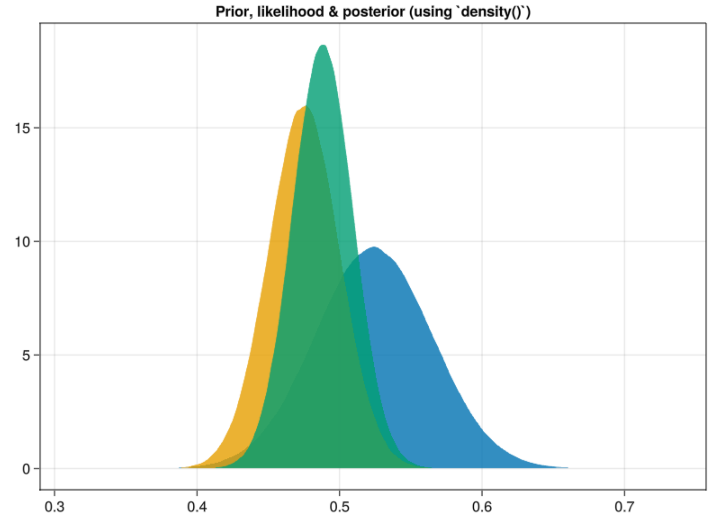
```
se_bayes = 0.02132543263776925
```

```
se_bayes = sqrt(1/(1/se_prior^2 +
1/se_data^2))
```




```
let
    x = 0.3:0.001:0.7
    f = Figure()
    ax = Axis(f[1, 1], title="Prior,
    likelihood & posterior")
    prior = lines!(f[1, 1], x, pdf.
    (Normal(theta_hat_prior, se_prior), x),
    color=:gray)
    data = lines!(x, pdf.
    (Normal(theta_hat_data, se_data),
    x),color=:darkred)
    bayes = lines!(x, pdf.
    (Normal(theta_hat_bayes, se_bayes), x),
    color=:darkblue)
    Legend(f[1, 2], [prior, data, bayes],
    ["Prior", "Data", "Bayesian"])

    current_figure()
end
```




```
let
    f = Figure()
    ax = Axis(f[1, 1], title="Prior,
    likelihood & posterior (using
    `density()`)")
    density!(rand(Normal(theta_hat_prior,
    se_prior), Int(1e6)), lab="prior")
    density!(rand(Normal(theta_hat_data,
    se_data), Int(1e6)), lab="likelihood")
    density!(rand(Normal(theta_hat_bayes,
    se_bayes), Int(1e6)), lab="bayes")
    current_figure()
end
```

9.4 Example of Bayesian inference: beauty and sex ratio.