See chapter 7 in Regression and Other Stories.

Widen the notebook.

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

```
using Pkg ✓ , DrWatson ✓
```

A typical set of Julia packages to include in notebooks.

```
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;
```

7.1 Example: Predicting presidential vote from the economy.

```
hdi =
```

	rank	state	hdi	canada
1	1	"Connecticut"	0.962	2
2	2	"Massachusetts"	0.961	2
3	3	"New Jersey"	0.961	2
4	4	"Washington, D.C."	0.96	4
5	5	"Maryland"	0.96	3
6	6	"Hawaii"	0.959	2
7	7	"New York"	0.959	1
8	8	"New Hampshire"	0.958	1
9	9	"Minnesota"	0.958	1
10	10	"Rhode Island"	0.958	3
⋮ more				
51	51	"Mississippi"	0.799	5

```
hdi = CSV.read(ros_datadir("HDI", "hdi.csv"), DataFrame)
```

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)
```

```
hibbs_lm =
StatsModels.TableRegressionModel{LinearModel{GLM

vote ~ 1 + growth

Coefficients:
────────────────────────────────────────────────────
                Coef.  Std. Error      t  Pr(>|t
────────────────────────────────────────────────────
(Intercept)  46.2476     1.62193   28.51    <1e-1
growth        3.06053    0.696274   4.40    0.000
────────────────────────────────────────────────────
```

```
hibbs_lm = lm(@formula(vote ~ growth),
hibbs)
```

```
[-8.99292, 2.66743, 1.0609, 2.20753, -5.89044, 4
```

```
residuals(hibbs_lm)
```

```
2.2744434224582912
```

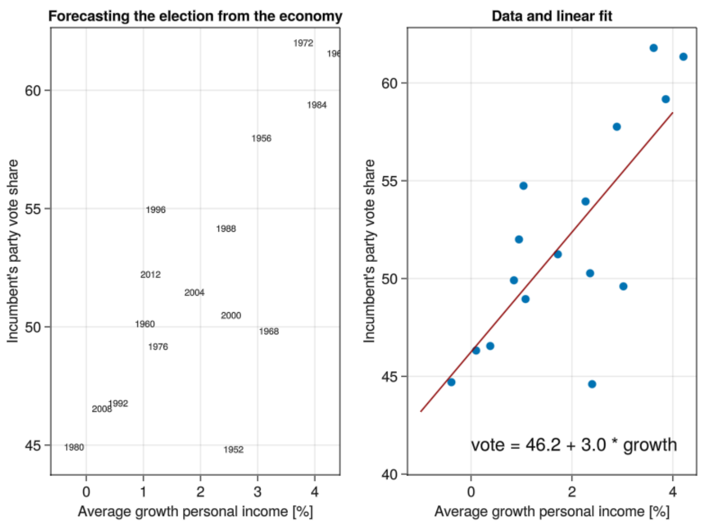
```
mad(residuals(hibbs_lm))
```

3.635681268522063

```
std(residuals(hibbs_lm))
```

[46.2476, 3.06053]

```
coef(hibbs_lm)
```




```julia
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)
```

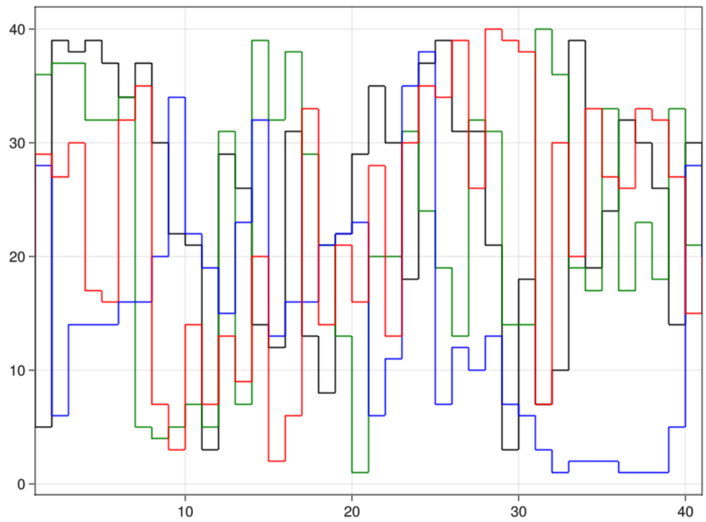
```julia
@model function ppl7_1(growth, vote)
    a ~ Normal(50, 20)
    b ~ Normal(2, 10)
    σ ~ Exponential(1)
    μ = a .+ b .* growth
    for i in eachindex(vote)
        vote[i] ~ Normal(μ[i], σ)
    end
end
```

	parameters	mean	std	naive_se
1	:a	46.2751	1.58511	0.0250628
2	:b	3.05431	0.673448	0.0106482
3	:σ	3.57802	0.602371	0.0095243

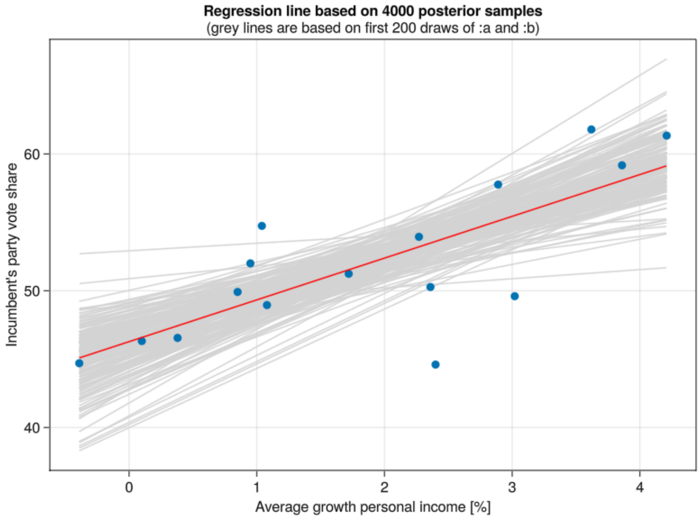
```julia
begin
    m7_1t = ppl7_1(hibbs.growth, hibbs.vote)
    chns7_1t = sample(m7_1t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns7_1t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	46.236	1.464	46.275	1.58
2	"b"	3.067	0.649	3.054	0.67
3	"σ"	3.505	0.59	3.578	0.60

```julia
begin
    post7_1t = DataFrame(chns7_1t)[:, 3:5]
    ms7_1t = model_summary(post7_1t,
    names(post7_1t))
end
```




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

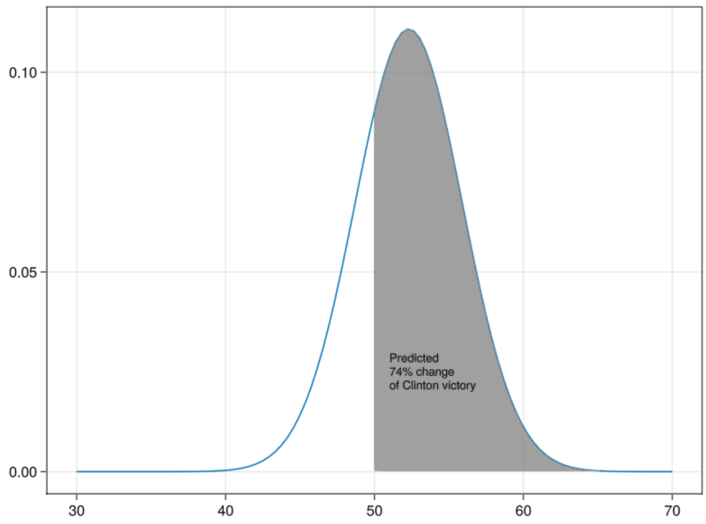



```
let
    growth_range =
    LinRange(minimum(hibbs.growth),
    maximum(hibbs.growth), 200)
    votes = mean.(link(post7_1t, (r,x) ->
    r.a + x * r.b, growth_range))

    fig = Figure()
    xlabel = "Average growth personal
    income [%]"
    ylabel="Incumbent's party vote share"
    ax = Axis(fig[1, 1]; 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 1:200
        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
```

0.7385523916379624

```
let
    println(46.3 + 3 * 2.0) # 52.3, σ = 3.6
    (from ms7_1s above)
    probability_of_Clinton_winning = 1 -
    cdf(Normal(52.3, 3.6), 50)
end
```




```
let
    f = Figure()
    ax = Axis(f[1, 1]; title = "")
    x_range = LinRange(30, 70, 100)
    y = pdf.(Normal(52.3, 3.6), x_range)
    lines!(x_range, y)

    x1 = range(50, 70; length=200)
    band!(x1, fill(0, length(x1)), pdf.
    (Normal(52.3, 3.6), x1);
        color = (:grey, 0.75), label =
        "Label")

    annotations!("Predicted\n74% change\nof
    Clinton victory", position=(51, 0.02),
    textsize=13)
    f
end
```

7.2 Checking the model-fitting procedure using simulation.

	parameters	mean	std	naive_se
1	:a	45.0094	1.69352	0.0267769
2	:b	3.41034	0.724007	0.0114475
3	:σ	3.94573	0.675559	0.0106815

```
let
    a = 46.3
    b = 3.0
    sigma = 3.9
    x = hibbs.growth
    n = length(x)

    y = a .+ b .* x + rand(Normal(0,
    sigma), n)
    fake = DataFrame(x = x, y = y)

    data = (N=nrow(fake), vote=fake.y,
    growth=fake.x)
    global m7_2t = ppl7_1(fake.x, fake.y)
    global chns7_2t = sample(m7_2t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns7_2t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	45.011	1.638	45.009	1.69
2	"b"	3.409	0.72	3.41	0.72
3	"σ"	3.852	0.624	3.946	0.67

```
begin
    post7_2t = DataFrame(chns7_2t)[:, 3:5]
    ms7_2t = model_summary(post7_2t,
    names(post7_2t))
end
```

sim (generic function with 1 method)

```julia
function sim(ppl)
    a = 46.3
    b = 3.0
    sigma = 3.9
    x = hibbs.growth
    n = length(x)

    y = a .+ b .* x + rand(Normal(0,
    sigma), n)
    #println(mean(y))
    m7_2t = ppl(x, y)
    chns7_2t = sample(m7_2t, NUTS(),
    MCMCThreads(), 1000, 4)
    post7_2t = DataFrame(chns7_2t)[:, 3:5]
    ms = model_summary(post7_2t, Symbol.
    ([:a, :b, :sigma]))
    b̂ = ms[:b, :mean]
    b_se = ms[:b, :std]

    (
        b̂ = b̂,
        b_se = b_se,
        cover_68 = Int(abs(b - b̂) < b_se),
        cover_95 = Int(abs(b - b̂) < 2b_se)
    )
end
```

(b̂ = 3.073, b_se = 0.673, cover_68 = 1, cover_9

```julia
sim(ppl7_1)
```

	variable	mean	min	median	max
1	:b̂	2.99808	1.073	2.913	4.895
2	:b_se	0.67236	0.46	0.66	0.962
3	:cover_68	0.56	0	1.0	1
4	:cover_95	0.91	0	1.0	1

```
let
    n_fake = 100 # 1000
    df = DataFrame()
    cover_68 = Float64[]
    cover_95 = Float64[]

    for i in 1:n_fake
        res = sim(ppl7_1)
        append!(df, DataFrame(;res...))
    end
    describe(df)
end
```

Note

In above cell, I have hidden the logs. To show them, click on the little circle with 3 dots.

7.3 Formulating comparisons as regression models.

```
▸ [3.305, 1.12992]
```

```
begin
    r₀ = [-0.3, 4.1, -4.9, 3.3, 6.4, 7.2, 10.7, -4.6, 4.7, 6.0, 1.1, -6.7, 10.2, 9.7, 5.6,
        1.7, 1.3, 6.2, -2.1, 6.5]
    [mean(r₀), std(r₀)/sqrt(length(r₀))]
end
```

```
(diff = 4.89914, se_0 = 1.12992, se_1 = 0.89368
```

```
begin
    Random.seed!(3)
    n₀ = 20
    y₀ = r₀
    fake_0 = DataFrame(y₀ = r₀)
    data_0 = (N = nrow(fake_0), y =
    fake_0.y₀)

    n₁ = 30
    y₁ = rand(Normal(8.0, 5.0), n₁)
    data_1 = (N = n₁, y = y₁)

    se_0 = std(y₀)/sqrt(n₀)
    se_1 = std(y₁)/sqrt(n₁)

    (diff=mean(y₁)-mean(y₀), se_0=se_0,
    se_1=se_1, se=sqrt(se_0^2 + se_1^2))
end
```

```
ppl7_3 (generic function with 2 methods)
```

```
@model function ppl7_3(y)
    a ~ Uniform(0, 10)
    σ ~ Uniform(0, 10)
    y ~ Normal(a, σ)
end
```

	parameters	mean	std	naive_se
1	:a	3.36139	1.20521	0.0190561
2	:σ	5.41613	0.932735	0.0147478

```
begin
    m7_3at = ppl7_3(data_0.y)
    chns7_3at = sample(m7_3at, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns7_3at)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	3.313	1.194	3.361	1.2(
2	"σ"	5.301	0.888	5.416	0.9:

```
begin
    post7_3at = DataFrame(chns7_3at)[:, 3:4]
    ms7_3at = model_summary(post7_3at,
    names(post7_3at))
end
```

	parameters	mean	std	naive_se
1	:a	8.12938	0.887754	0.0140366
2	:σ	5.10509	0.699261	0.0110563

```
begin
    m7_3bt = ppl7_3(data_1.y)
    chns7_3bt = sample(m7_3bt, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns7_3bt)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	8.145	0.899	8.129	0.8{
2	"σ"	5.023	0.68	5.105	0.6{

```
begin
    post7_3bt = DataFrame(chns7_3bt)[:, 3:4]
    ms7_3bt = model_summary(post7_3bt,
    names(post7_3bt))
end
```

```
ppl7_3c (generic function with 2 methods)
```

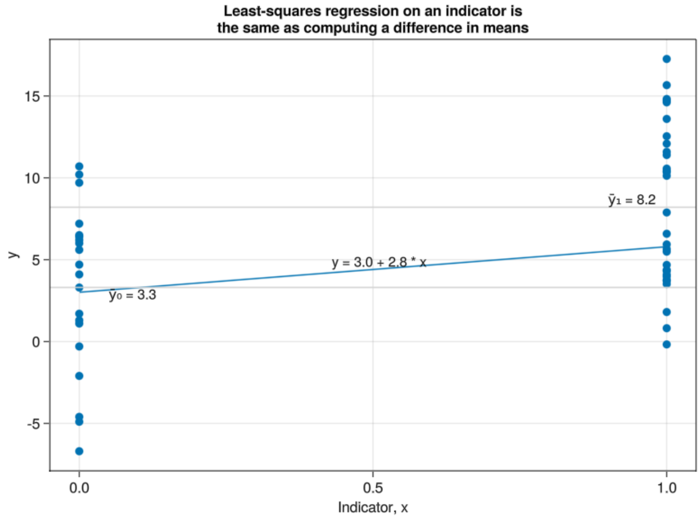
```julia
@model function ppl7_3c(x, y)
    a ~ Normal()
    b ~ Normal()
    σ ~ Exponential(1)
    μ = a .+ b .* x
    for i in eachindex(y)
        y[i] ~ Normal(μ[i], σ)
    end
end
```

	parameters	mean	std	naive_se
1	:a	2.99498	0.658544	0.0104125
2	:b	2.7782	0.784782	0.0124085
3	:σ	5.14012	0.545518	0.0086253!

```julia
let
    n = n₀ + n₁
    y = vcat(y₀, y₁)
    x = vcat(zeros(Int, n₀), ones(Int, n₁))
    global fake = DataFrame(x=x, y=y)
    global m7_3ct = ppl7_3c(fake.x, fake.y)
    global chns7_3ct = sample(m7_3ct,
    NUTS(), MCMCThreads(), 1000, 4)
    describe(chns7_3ct)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	3.011	0.666	2.995	0.65
2	"b"	2.781	0.766	2.778	0.78
3	"σ"	5.101	0.538	5.14	0.54

```julia
begin
    post7_3ct = DataFrame(chns7_3ct)[:, 3:6]
    sm7_3ct = model_summary(post7_3ct, [:a,
    :b, :σ])
end
```




```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Least-squares regression on an indicator is\nthe same as computing a difference in means",
    xlabel="Indicator, x", ylabel="y")
    x_range = LinRange(0, 1, 100)
    â, b̂, σ̂ = sm7_3ct[:, :median]

    y = â .+ b̂ .* x_range
    lines!(x_range, y)
    x = vcat(zeros(Int, n₀), ones(Int, n₁))
    scatter!(fake.x, fake.y)
    ȳ₀ = mean(y₀)
    ȳ₁ = mean(y₁)
    hlines!(ax, [ȳ₀, ȳ₁]; color=:lightgrey)
    annotations!("ȳ₀ = $(round(ȳ₀, digits=1))", position=(0.05, 2.4), textsize=15)
    annotations!("ȳ₁ = $(round(ȳ₁, digits=1))", position=(0.9, 8.2), textsize=15)
    annotations!("y = $(round(â, digits=1)) + $(round(b̂, digits=1)) * x", position=(0.43, 4.4), textsize=15)
    f
end
```

```
8.204138555696407
```

```
mean(y₁)
```