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 ROSStanPluto
    using StanSample ✓

    # Graphics related
    using GLMakie ✓

    # Common data files and functions
    using RegressionAndOtherStories ✓
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

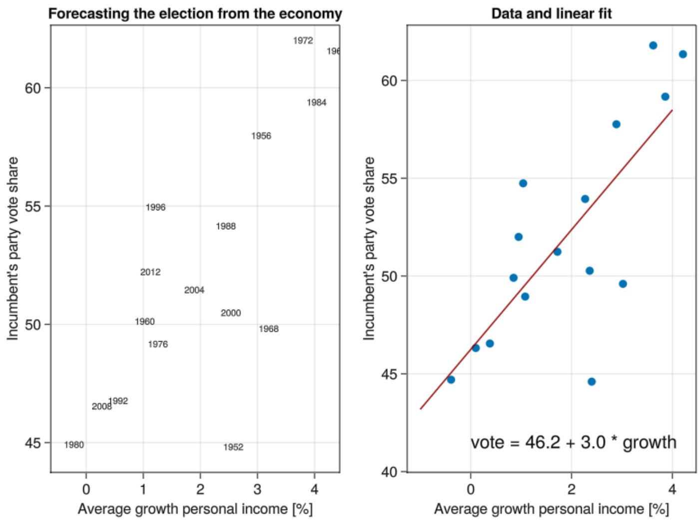
```
mad(residuals(hibbs_lm))
```

3.635681268522063

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

▸ [46.2476, 3.06053]

```
coef(hibbs_lm)
```




```
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
```

```
stan7_1 = "
data {
    int<lower=1> N;       // total number of
    observations
    vector[N] growth;     // Independent
    variable: growth
    vector[N] vote;       // Dependent
    variable: votes
}
parameters {
    real b;               // Coefficient
    independent variable
    real a;               // Intercept
    real<lower=0> sigma; // dispersion
    parameter
}
model {
    vector[N] mu;

    // priors including constants
    a ~ normal(50, 20);
    b ~ normal(2, 10);
    sigma ~ exponential(1);

    mu = a + b * growth;

    // likelihood including constants
    vote ~ normal(mu, sigma);
}";
```

	parameters	mean	mcse	std
1	"b"	3.03591	0.0167247	0.652008
2	"a"	46.2957	0.0392767	1.52911
3	"sigma"	3.57778	0.0136109	0.602732

```
let
    data = (N=nrow(hibbs), vote=hibbs.vote,
    growth=hibbs.growth)
    global m7_1s = SampleModel("hibbs",
    stan7_1)
    global rc7_1s = stan_sample(m7_1s; data)
    success(rc7_1s) && describe(m7_1s)
end
```

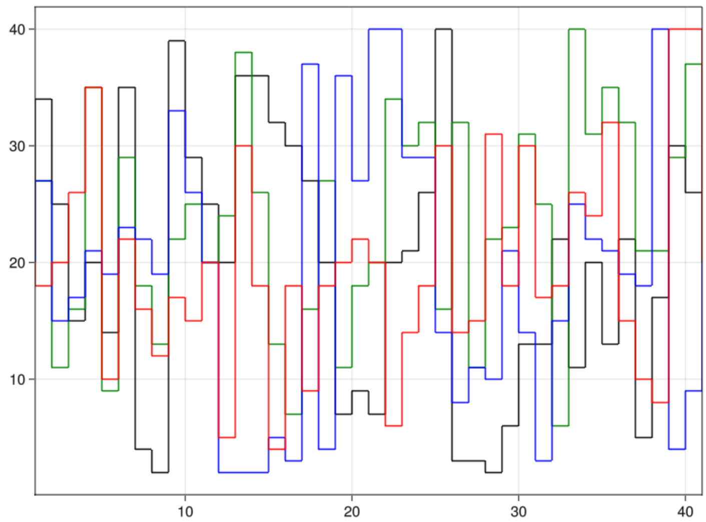
```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
d.
```

Note

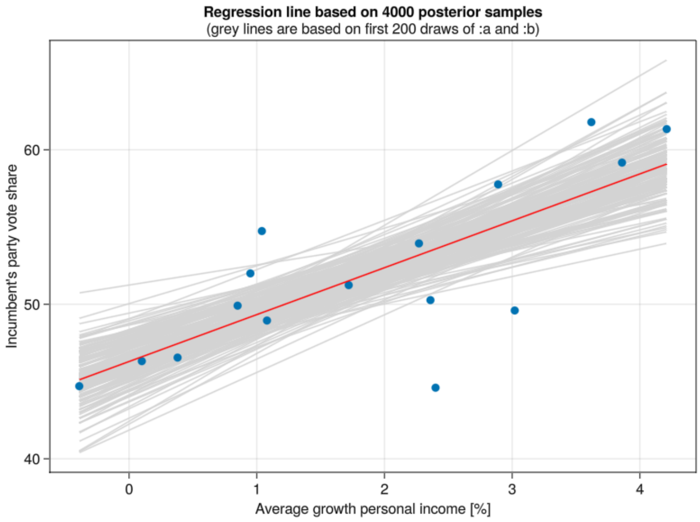
Sometimes I hide or show the output logs. To show them, click on the little circle with 3 dots visible in the top right of the input cell if the cursor is in there. Try it!

	parameters	median	mad_sd	mean	st
1	"a"	46.308	1.479	46.296	1.52
2	"b"	3.047	0.633	3.036	0.65
3	"sigma"	3.496	0.566	3.578	0.60

```
if success(rc7_1s)
    post7_1s = success(rc7_1s) &&
    read_samples(m7_1s, :dataframe)
    ms7_1s = model_summary(post7_1s, [:a,
    :b, :sigma])
end
```




```
trankplot(post7_1s, "b")
```



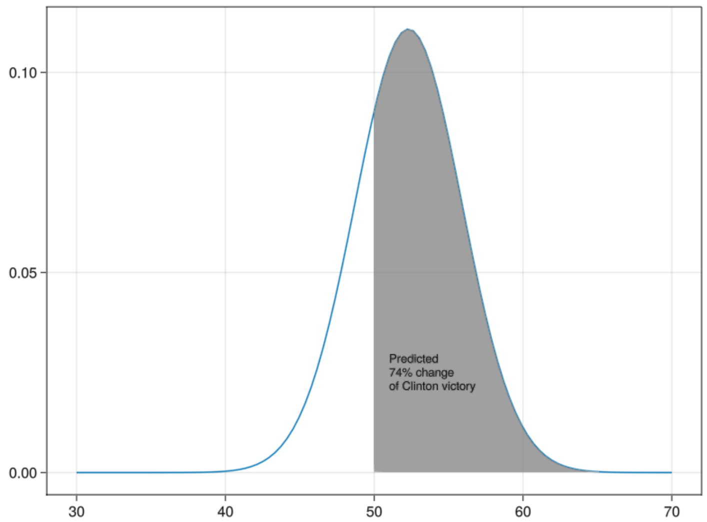

```
let
    growth_range =
    LinRange(minimum(hibbs.growth),
    maximum(hibbs.growth), 200)
    votes = mean.(link(post7_1s, (r,x) ->
    r.a + x * r.b, growth_range))

    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_1s.a[i]
        .+ post7_1s.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
```

```
52.3
```




```
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	mcse	std
1	"b"	2.94595	0.0163709	0.633977
2	"a"	45.3194	0.0365307	1.47054
3	"sigma"	3.39667	0.0139647	0.58763

```
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_2s = SampleModel("fake",
    stan7_1)
    global rc7_2s = stan_sample(m7_2s; data)
    success(rc7_2s) && describe(m7_2s)
end
```

```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
d.
```

	parameters	median	mad_sd	mean	st
1	"b"	2.957	0.603	2.946	0.63
2	"a"	45.336	1.403	45.319	1.47
3	"sigma"	3.318	0.555	3.397	0.58

```
if success(rc7_2s)
    post7_2s = read_samples(m7_2s,
    :dataframe)
    ms7_2s = model_summary(post7_2s,
    names(post7_2s))
end
```

	parameters	median	mad_sd	mean	st
1	"a"	46.308	1.479	46.296	1.52
2	"b"	3.047	0.633	3.036	0.65
3	"sigma"	3.496	0.566	3.578	0.60

```
ms7_1s
```

sim (generic function with 1 method)

```julia
function sim(sm::SampleModel)
    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))
    data_sim = (N=n, vote=y, growth=x)
    rc = stan_sample(sm; data=data_sim)
    post = read_samples(sm, :dataframe)
    ms = model_summary(post, 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
```

```julia
m7_2_1s = SampleModel("fake_sim", stan7_1);
```

```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
ated.
```

```
(b̂ = 2.606, b_se = 0.548, cover_68 = 1, cover_9
```

```julia
sim(m7_2_1s)
```

```
50.21592438279586
```

```
false
```

```julia
isdefined(Main, :StanSample)
```

	parameters	median	mad_sd	mean	st
1	"a"	46.308	1.479	46.296	1.52
2	"b"	3.047	0.633	3.036	0.65
3	"sigma"	3.496	0.566	3.578	0.60

```
ms7_1s
```

Or use the underlying DataFrame directly.

1.479

```
ms7_1s["a", "mad_sd"]
```

1.479

```
ms7_1s[:a, :mad_sd]
```

[1.479, 0.633, 0.566]

```
ms7_1s[:, :mad_sd]
```

```
ms7_1s[:c, :mad_sd]
```

Parameter `c` is not in ["a", "b", "sigma"].

```
ms7_1s[:a, :mad]
```

Statistic `mad` is not in ["parameters", "mediar

DataFrameRow (2 columns)

	median	mad_sd
	Float64	Float64
3	3.496	0.566

```
ms7_1s[3, [:median, :mad_sd]]
```

String

```
eltype(ms7_1s.parameters)
```

	variable	mean	min	median	max
1	:b̂	3.117	2.017	3.277	3.696
2	:b_se	0.704	0.587	0.704	0.824
3	:cover_68	0.8	0	1.0	1
4	:cover_95	1.0	1	1.0	1

```
let
    n_fake = 10  # 1000
    df = DataFrame()
    cover_68 = Float64[]
    cover_95 = Float64[]
    m7_2_1s = SampleModel("fake_sim_1",
    stan7_1)

    for i in 1:n_fake
        res = sim(m7_2_1s)
        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.

```
stan7_3 = "
data {
    int N;
    vector[N] y;
}
parameters {
    real a;
    real sigma;
}
model {
    y ~ normal(a, sigma);
}
";
```

```
[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
```

	parameters	mean	mcse	std
1	"a"	3.30376	0.0264402	1.26012
2	"sigma"	5.42711	0.0221087	0.954644

```
begin
    m7_3_0s = SampleModel("fake_0", stan7_3)
    rc7_3_0s = stan_sample(m7_3_0s;
     data=data_0)
    success(rc7_3_0s) && describe(m7_3_0s)
end
```

	parameters	mean	mcse	std
1	"a"	8.1841	0.0203487	0.957577
2	"sigma"	5.12561	0.014233	0.709505

```
begin
    m7_3_1s = SampleModel("fake_1", stan7_3)
    rc7_3_1s = stan_sample(m7_3_1s;
     data=data_1)
    success(rc7_3_1s) && describe(m7_3_1s)
end
```

Note

In above cells, the logs are hidden.

	parameters	median	mad_sd	mean	st
1	"a"	3.293	1.23	3.304	1.26
2	"sigma"	5.302	0.873	5.427	0.95

```
if success(rc7_3_0s)
    post7_3_0s = read_samples(m7_3_0s,
    :dataframe)
    sm7_3_0s = model_summary(post7_3_0s,
    [:a, :sigma])
end
```

	parameters	median	mad_sd	mean	st
1	"a"	8.187	0.92	8.184	0.95
2	"sigma"	5.041	0.684	5.126	0.71

```
if success(rc7_3_1s)
    post7_3_1s = read_samples(m7_3_1s,
    :dataframe)
    sm7_3_1s = model_summary(post7_3_1s,
    [:a, :sigma])
end
```

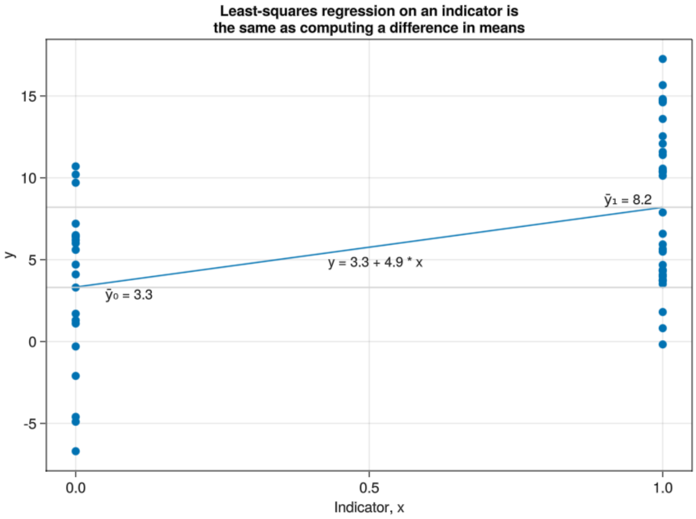
```
stan7_3_2 = "
data {
    int N;
    vector[N] y;
    vector[N] x;
}
parameters {
    real a;
    real b;
    real sigma;
}
model {
    vector[N] mu;
    mu = a + b * x;
    y ~ normal(mu, sigma);
}
";
```

	parameters	mean	mcse	std	5%
1	"a"	3.32	0.03	1.12	1.43
2	"b"	4.89	0.03	1.44	2.56
3	"sigma"	5.07	0.01	0.53	4.28

```
let
    n = n₀ + n₁
    y = vcat(y₀, y₁)
    x = vcat(zeros(Int, n₀), ones(Int, n₁))
    global fake = DataFrame(x=x, y=y)
    data = (N = n, x = x, y = y)
    global m7_3_2s = SampleModel("fake_2",
    stan7_3_2)
    global rc7_3_2s = stan_sample(m7_3_2s;
    data)
    success(rc7_3_2s) && describe(m7_3_2s,
    [:a, :b, :sigma])
end
```

	parameters	median	mad_sd	mean	st
1	"a"	3.327	1.117	3.317	1.11
2	"b"	4.871	1.45	4.892	1.44
3	"sigma"	5.026	0.531	5.073	0.53

```
if success(rc7_3_2s)
    post7_3_2s = read_samples(m7_3_2s,
    :dataframe)
    sm7_3_2s = model_summary(post7_3_2s,
    [:a, :b, :sigma])
end
```




```julia
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 = sm7_3_2s[:a, :median]
    b̂ = sm7_3_2s[:b, :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
```