See Chapter 6 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;
```

6.1 Regression models.

6.2 Fitting a simple regression to fake data.

	x	y
1	1.0	0.175189
2	2.0	1.70123
3	3.0	1.22936
4	4.0	0.79866
5	5.0	1.31689
6	6.0	2.30556
7	7.0	3.02729
8	8.0	3.24706
9	9.0	2.04172
10	10.0	2.74121
⋮	more	
20	20.0	6.72016

```
let
    n = 20
    x = LinRange(1, n, 20)
    a = 0.2
    b = 0.3
    sigma = 0.5
    y = a .+ b .* x .+ rand(Normal(0,
    sigma), n)
    global fake = DataFrame(x=x, y=y)
end
```

```
ppl6_1 (generic function with 2 methods)
```

```
@model function ppl6_1(x, y)
    a ~ Uniform(-2, 2)
    b ~ Uniform(-2, 2)
    σ ~ Uniform(0, 10)
    μ = a .+ b .* x
    for i in eachindex(y)
        y[i] ~ Normal(μ[i], σ)
    end
end
```

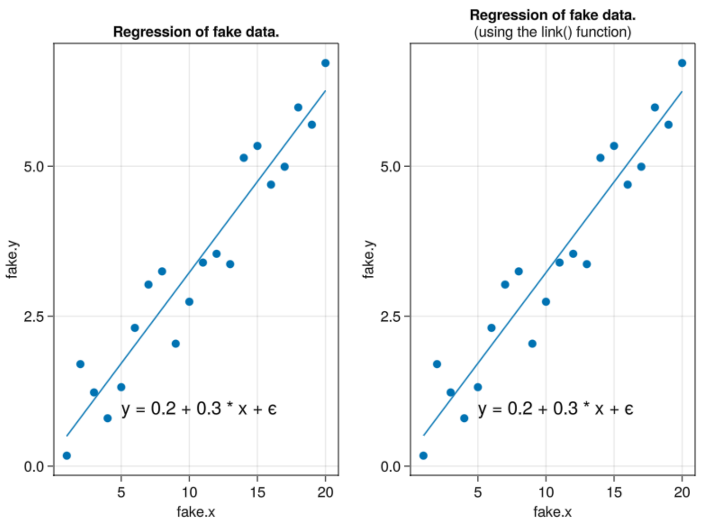
▸ [

	parameters	mean	std	naive_
1	:a	0.203286	0.281582	0.004452
2	:b	0.302298	0.0232234	0.000367
3	:σ	0.603167	0.110236	0.001742

```
begin
    m6_1t = ppl6_1(fake.x, fake.y)
    chns6_1t = sample(m6_1t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns6_1t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	0.199	0.279	0.203	0.28
2	"b"	0.303	0.023	0.302	0.02
3	"σ"	0.587	0.102	0.603	0.11

```
begin
    post6_1t = DataFrame(chns6_1t)[:, 3:5]
    ms6_1t = model_summary(post6_1t,
    names(post6_1t))
end
```



```julia
let
    f = Figure()

    ax = Axis(f[1, 1]; title="Regression of fake data.", xlabel="fake.x", ylabel="fake.y")
    scatter!(fake.x, fake.y)
    x = 1:0.01:20
    y = ms6_1t["a", "median"] .+ ms6_1t["b", "median"] .* x
    lines!(x, y)
    â, b̂, ô  = round.(ms6_1t[:, "median"]; digits=2)
    annotations!("y = $(â) + $(b̂) * x + ϵ"; position=(5, 0.8))

    ax = Axis(f[1, 2]; title="Regression of fake data.", subtitle="(using the link() function)",
        xlabel="fake.x", ylabel="fake.y")
    scatter!(fake.x, fake.y)
    xrange = LinRange(1, 20, 200)
    y = mean.(link(post6_1t, (r,x) -> r.a + x * r.b, xrange))
    lines!(xrange, y)
    annotations!("y = $(â) + $(b̂) * x + ϵ"; position=(5, 0.8))

    current_figure()
end
```

	parameters	simulated	median	mad_sd
1	:a	0.2	0.199	0.279
2	:b	0.3	0.303	0.023
3	:σ	0.5	0.587	0.102

```
DataFrame(parameters = Symbol.
(names(post6_1t)), simulated = [0.2, 0.3,
0.5], median = ms6_1t[:, "median"], mad_sd
= ms6_1t[:, "mad_sd"])
```

6.3 Interpret coefficients as comparisons, not effects.

	earnk	height	male
1	50.0	74	1
2	60.0	66	0
3	30.0	64	0
4	25.0	65	0
5	50.0	63	0
6	62.0	68	0
7	51.0	63	0
8	9.0	64	0
9	29.0	62	0
10	32.0	73	1
⋮ more			
1816	6.0	68	1

```julia
begin
    earnings = CSV.read(ros_datadir("Earnings", "earnings.csv"), DataFrame)
    earnings[:, [:earnk, :height, :male]]
end
```

	variable	mean	min	median	max	n
1	:earnk	21.1473	0.0	16.0	400.0	0
2	:height	66.5688	57	66.0	82	0
3	:male	0.371696	0	0.0	1	0

```julia
describe(earnings[:, [:earnk, :height, :male]])
```

```
ppl6_2 (generic function with 2 methods)
```

```
@model function ppl6_2(male, height, earnk)
    a ~ Normal()
    b ~ Normal()
    c ~ Normal()
    σ ~ Exponential(1)
    μ = a .+ b .* height .+ c .* male
    for i in eachindex(earnk)
        earnk[i] ~ Normal(μ[i], σ)
    end
end
```

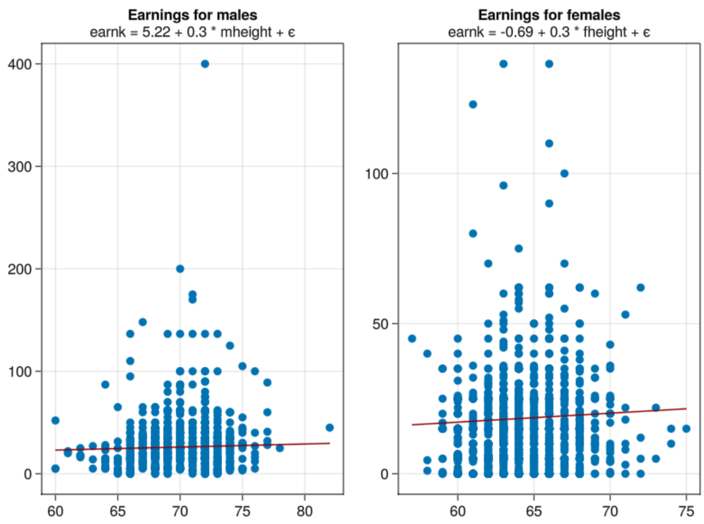
▸ [

	parameters	mean	std	naive
1	:a	-0.676695	0.995374	0.01573
2	:b	0.297586	0.0173254	0.00027
3	:c	5.89894	0.741121	0.01171
4	:σ	21.5295	0.354245	0.0056(

```
begin
    m6_2t = ppl6_2(earnings.male,
    earnings.height, earnings.earnk)
    chns6_2t = sample(m6_2t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns6_2t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	-0.689	1.01	-0.677	0.9(
2	"b"	0.298	0.017	0.298	0.01
3	"c"	5.911	0.736	5.899	0.74
4	"σ"	21.524	0.348	21.53	0.35

```
begin
    post6_2t = DataFrame(chns6_2t)[:, 3:6]
    ms6_2t = model_summary(post6_2t,
    names(post6_2t))
end
```



```
let
    â, b̂, ĉ, σ̂ = round.(ms6_2t[:, "median"]; digits=2)

    fig = Figure()

    ax = Axis(fig[1, 1]; title="Earnings for males", subtitle="earnk = $(round(ĉ + â; digits=2)) + $(b̂) * mheight + ϵ")
    m = sort(earnings[earnings.male .== 1, [:height, :earnk]])
    scatter!(m.height, m.earnk)
    mheight_range = LinRange(minimum(m.height), maximum(m.height), 200)
    earnk = mean.(link(post6_2t, (r,x) -> r.c + r.a + x * r.b, mheight_range))

    lines!(mheight_range, earnk; color=:darkred)

    ax = Axis(fig[1, 2]; title="Earnings for females", subtitle="earnk = $(â) + $(b̂) * fheight + ϵ")
    f = sort(earnings[earnings.male .== 0, [:height, :earnk]])
    scatter!(f.height, f.earnk)
    fheight_range = LinRange(minimum(f.height), maximum(f.height), 200)
    earnk = mean.(link(post6_2t, (r,x) -> r.a + x * r.b, fheight_range))
    lines!(fheight_range, earnk; color=:darkred)

    fig
    end
```

```
R2 = 0.08694354978286623
```

```
R2 = 1 - ms6_2t["σ", "mean"]^2 /
std(earnings.earnk)^2
```

6.4 Historical origins of regression.

```
heights =
```

	daughter_height	mother_height
1	52.5	59.5
2	52.5	59.5
3	53.5	59.5
4	53.5	59.5
5	55.5	59.5
6	55.5	59.5
7	55.5	59.5
8	55.5	59.5
9	56.5	58.5
10	56.5	58.5
⋮ more		
5524	73.5	63.5

```
heights =
CSV.read(ros_datadir("PearsonLee",
"heights.csv"), DataFrame)
```

```
ppl6_3 (generic function with 2 methods)
```

```julia
@model function ppl6_3(m_height, d_height)
    a ~ Normal()
    b ~ Normal()
    σ ~ Exponential(1)
    μ = a .+ b .* m_height
    for i in eachindex(d_height)
        d_height[i] ~ Normal(μ[i], σ)
    end
end
```

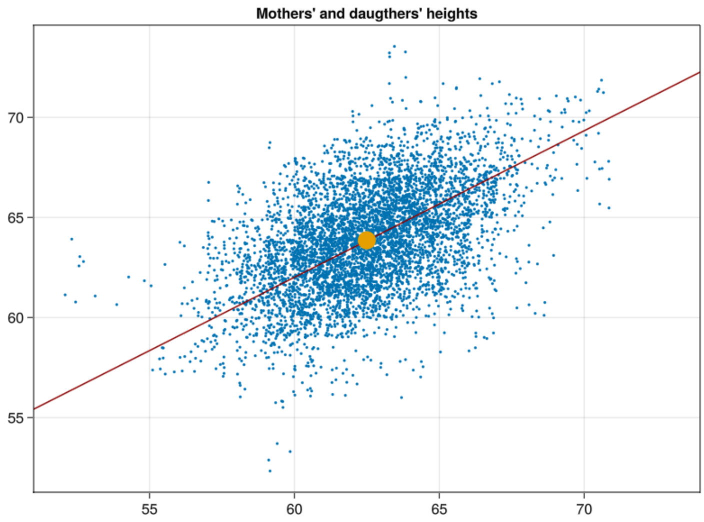
▸ [

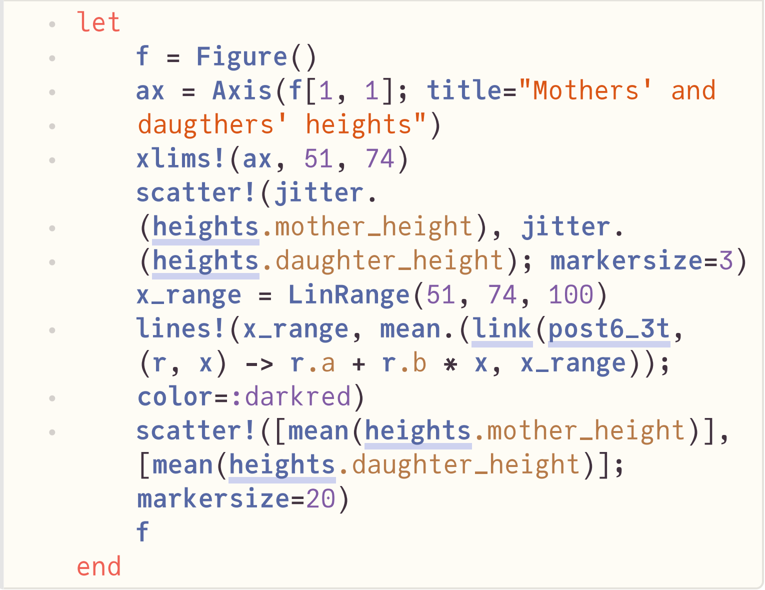
	parameters	mean	std	naive_
1	:a	18.0761	0.627493	0.009921
2	:b	0.732217	0.0100345	0.000158
3	:σ	2.30724	0.0223726	0.000353

```julia
begin
    m6_3t = ppl6_3(heights.mother_height,
    heights.daughter_height)
    chns6_3t = sample(m6_3t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns6_3t)
end
```

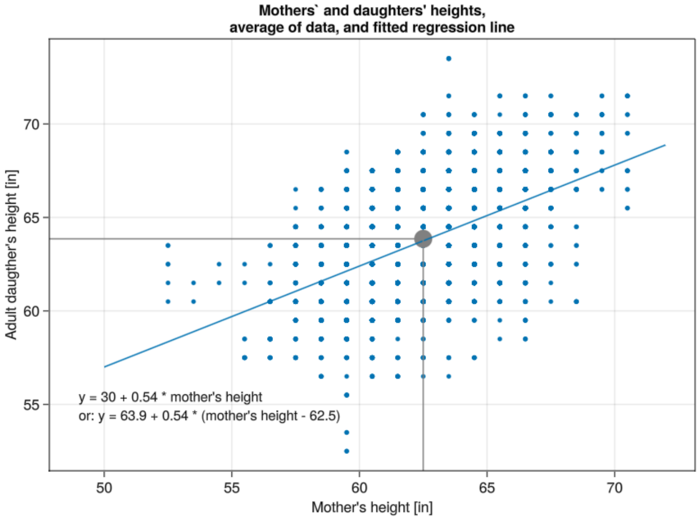
	parameters	median	mad_sd	mean	st
1	"a"	18.087	0.614	18.076	0.62
2	"b"	0.732	0.01	0.732	0.01
3	"σ"	2.307	0.022	2.307	0.02

```julia
begin
    post6_3t = DataFrame(chns6_3t)[:, 3:5]
    ms6_3t = model_summary(post6_3t,
    names(post6_3t))
end
```

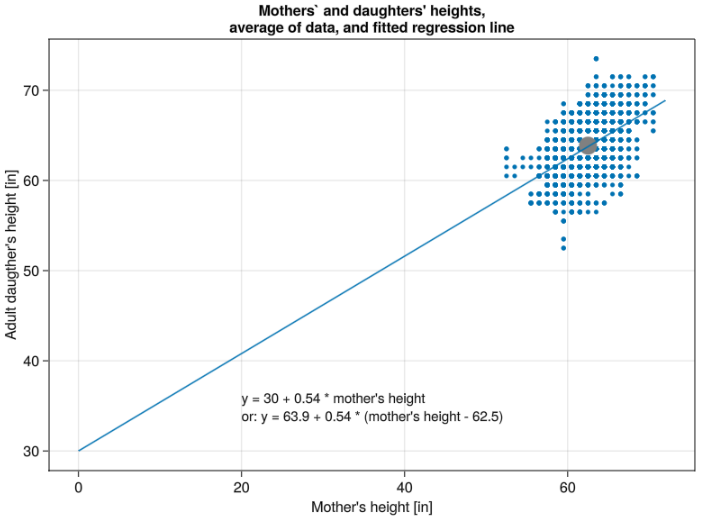




```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Mothers' and
    daugthers' heights")
    xlims!(ax, 51, 74)
    scatter!(jitter.
    (heights.mother_height), jitter.
    (heights.daughter_height); markersize=3)
    x_range = LinRange(51, 74, 100)
    lines!(x_range, mean.(link(post6_3t,
    (r, x) -> r.a + r.b * x, x_range));
    color=:darkred)
    scatter!([mean(heights.mother_height)],
    [mean(heights.daughter_height)];
    markersize=20)
    f
end
```



```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Mothers` and
    daughters' heights,\naverage of data,
    and fitted regression line",
        xlabel="Mother's height [in]",
        ylabel="Adult daugther's height
        [in]")
    scatter!(heights.mother_height,
    heights.daughter_height; markersize=5)
    xrange = LinRange(50, 72, 100)
    y = 30 .+ 0.54 .* xrange
    m̄ = mean(heights.mother_height)
    d̄ = mean(heights.daughter_height)
    scatter!([m̄], [d̄]; markersize=20,
    color=:gray)
    lines!(xrange, y)
    vlines!(ax, m̄; ymax=0.55, color=:grey)
    hlines!(ax, d̄; xmax=0.58, color=:grey)
    annotations!("y = 30 + 0.54 * mother's
    height", position=(49, 55), textsize=15)
    annotations!("or: y = 63.9 + 0.54 *
    (mother's height - 62.5)", position=
    (49, 54), textsize=15)
    f
end
```



```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Mothers` and
    daughters' heights,\naverage of data,
    and fitted regression line",
        xlabel="Mother's height [in]",
        ylabel="Adult daugther's height
        [in]")
    scatter!(heights.mother_height,
    heights.daughter_height; markersize=5)
    xrange = LinRange(0, 72, 100)
    y = 30 .+ 0.54 .* xrange
    m̄ = mean(heights.mother_height)
    d̄ = mean(heights.daughter_height)
    scatter!([m̄], [d̄]; markersize=20,
    color=:gray)
    lines!(xrange, y)
    annotations!("y = 30 + 0.54 * mother's
    height", position=(20, 35), textsize=15)
    annotations!("or: y = 63.9 + 0.54 *
    (mother's height - 62.5)", position=
    (20, 33), textsize=15)
    f
end
```

```
ppl6_4 (generic function with 2 methods)
```

```
@model function ppl6_4(m_height, d_height)
    a ~ Normal(25, 3)
    b ~ Normal(0, 0.5)
    σ ~ Exponential(1)
    μ = a .+ b .* m_height
    for i in eachindex(d_height)
        d_height[i] ~ Normal(μ[i], σ)
    end
end
```

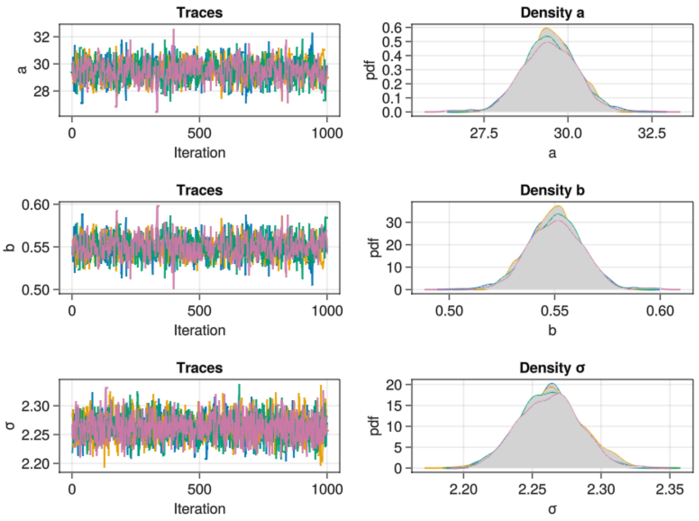
▸ [

	parameters	mean	std	naive_s
1	:a	29.4845	0.760218	0.0120201
2	:b	0.54995	0.0121652	0.0001923
3	:σ	2.26237	0.0216619	0.0003425

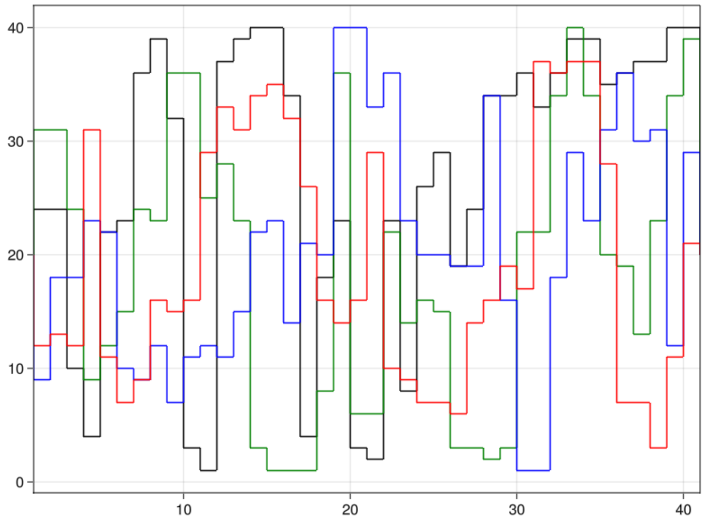
```julia
begin
    m6_4t = ppl6_4(heights.mother_height,
    heights.daughter_height)
    chns6_4t = sample(m6_4t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns6_4t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	29.473	0.734	29.485	0.76
2	"b"	0.55	0.012	0.55	0.01
3	"σ"	2.262	0.022	2.262	0.02

```julia
begin
    post6_4t = DataFrame(chns6_4t)[:, 3:5]
    ms6_4t = model_summary(post6_4t,
    names(post6_4t))
end
```



```
plot_chains(post6_4t, [:a, :b, :σ])
```



```
trankplot(post6_4t, "b")
```

Above trankplot and the low `ess` numbers a couple of cells earlier do not look healthy.

6.5 The paradox of regression to the mean.

	midterm	final
1	70.8711	55.178
2	21.7327	44.2992
3	67.0099	53.225
4	27.4136	50.4825
5	42.3918	68.7687
6	80.811	81.8612
7	34.2082	52.7463
8	52.8503	47.5438
9	62.3295	69.732
10	63.4512	69.7803
⋮ more		
1000	72.4838	65.602

```
let
    n = 1000
    true_ability = rand(Normal(50, 10), n)
    noise_1 = rand(Normal(0, 10), n)
    noise_2 = rand(Normal(0, 10), n)
    midterm = true_ability + noise_1
    final = true_ability + noise_2
    global exams =
    DataFrame(midterm=midterm, final=final)
end
```

```
ppl6_5 (generic function with 2 methods)
```

```
@model function ppl6_5(midterm, final)
    a ~ Normal()
    b ~ Normal()
    σ ~ Exponential(1)
    μ = a .+ b .* midterm
    for i in eachindex(final)
        final[i] ~ Normal(μ[i], σ)
    end
end
```

▸ [

	parameters	mean	std	naive_
1	:a	7.31704	0.870082	0.013757
2	:b	0.830372	0.0178971	0.000282
3	:σ	13.4318	0.307599	0.004863

```
begin
    m6_5t = ppl6_5(exams.midterm,
    exams.final)
    chns6_5t = sample(m6_5t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns6_5t)
end
```

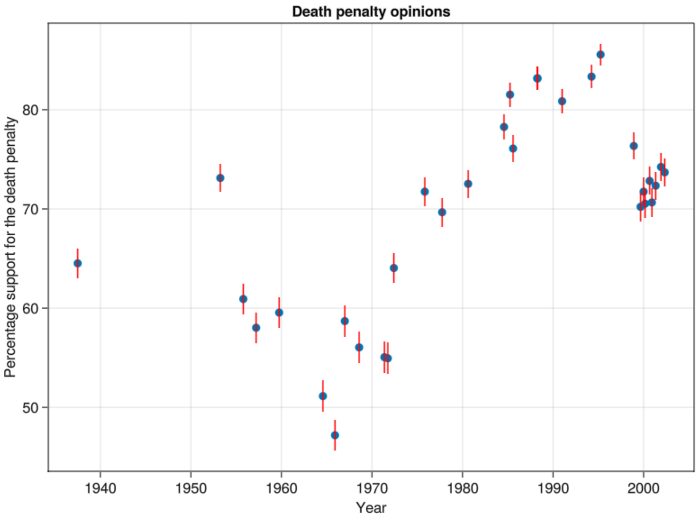
	parameters	median	mad_sd	mean	st
1	"a"	7.31	0.866	7.317	0.87
2	"b"	0.831	0.018	0.83	0.01
3	"σ"	13.43	0.308	13.432	0.30

```
begin
    post6_5t = DataFrame(chns6_5t)[:, 3:5]
    ms6_5t = model_summary(post6_5t,
    names(post6_5t))
end
```

```
df_poll =
```

	poll1	poll2	poll3	poll4	poll5
1	2002	10.0	70.0	25.0	5.0
2	2002	5.0	72.0	25.0	3.0
3	2001	10.0	68.0	26.0	6.0
4	2001	5.0	65.0	27.0	8.0
5	2001	2.0	67.0	25.0	8.0
6	2000	8.0	67.0	28.0	5.0
7	2000	6.0	66.0	26.0	8.0
8	2000	2.0	66.0	28.0	6.0
9	1999	5.0	71.0	22.0	7.0
10	1995	9.0	77.0	13.0	10.0
⋮ more					
32	1937	12.0	60.0	33.0	7.0

```
df_poll = CSV.read(ros_datadir("Death",
"polls.csv"), DataFrame)
```



```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Death penalty
    opinions", xlabel="Year",
    ylabel="Percentage support for the
    death penalty")
    scatter!(df_poll.year, df_poll.support
    .* 100)
    err_lims = [100(sqrt(df_poll.support[i]*
    (1-df_poll.support[i])/1000)) for i in
    1:nrow(df_poll)]
    errorbars!(df_poll.year, df_poll.support
     .* 100, err_lims, color = :red)
    f
end
```

Used in later notebooks.

	STATE	TOTLDF	DOR	DORAVG	HRS
1	"AL"	296.0	33.47	32.65	11.61
2	"AR"	77.0	15.4	15.65	9.7
3	"AZ"	231.0	41.5	39.42	7.92
4	"CA"	528.0	9.21	9.14	8.8
5	"FL"	851.0	30.19	30.18	10.91
6	"GA"	323.0	19.63	19.12	12.78
7	"ID"	31.0	48.48	44.16	3.55
8	"IL"	238.0	11.26	10.98	8.18
9	"IN"	79.0	11.81	10.93	5.61
10	"KY"	59.0	10.67	10.24	7.03
⋮ more					
26	"WY"	5.0	9.98	11.63	4.58

```
begin
    death_raw=CSV.read(ros_datadir("Death",
    "dataforandy.csv"), DataFrame;
    missingstring="NA")
    death =
    death_raw[completecases(death_raw), :]
end
```

```
let
    st_abbr = death[:, 1]
    ex_rate = death[:, 8] ./ 100
    err_rate = death[:, 7] ./ 100
    hom_rate = death[:, 5] ./ 100000
    ds_per_homicide = death[:, 3] ./ 1000
    ds = death[:, 2]
    hom = ds ./ ds_per_homicide
    ex = ex_rate .* ds
    err = err_rate .* ds
    pop = hom ./ hom_rate
    std_err_rate = sqrt.( (err .+ 1) .* (ds
    .+ 1 .- err) ./ ((ds .+ 2).^2 .* (ds .+
    3)) )
end;
```