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 ROSStanPluto
    using StanSample ✓

    # Graphics related
    using GLMakie ✓

    # Common data files and functions
    using RegressionAndOtherStories ✓
end
```

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

6.1 Regression models.

6.2 Fitting a simple regression to fake data.

	x	y
1	1.0	1.18017
2	2.0	0.66976
3	3.0	0.951537
4	4.0	0.941839
5	5.0	1.90931
6	6.0	1.6079
7	7.0	2.41162
8	8.0	3.01127
9	9.0	2.70929
10	10.0	2.80496
⋮ more		
20	20.0	6.03491

```
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
```

```
stan6_1 = "
data {
    int N;
    vector[N] x;
    vector[N] y;
}
parameters {
    real a;
    real b;
    real<lower=0> sigma;
}
model {
    vector[N] mu;
    a ~ uniform(-2, 2);
    b ~ uniform(-2, 2);
    sigma ~ uniform(0, 10);
    mu = a + b * x;
    y ~ normal(mu, sigma);
}";
```

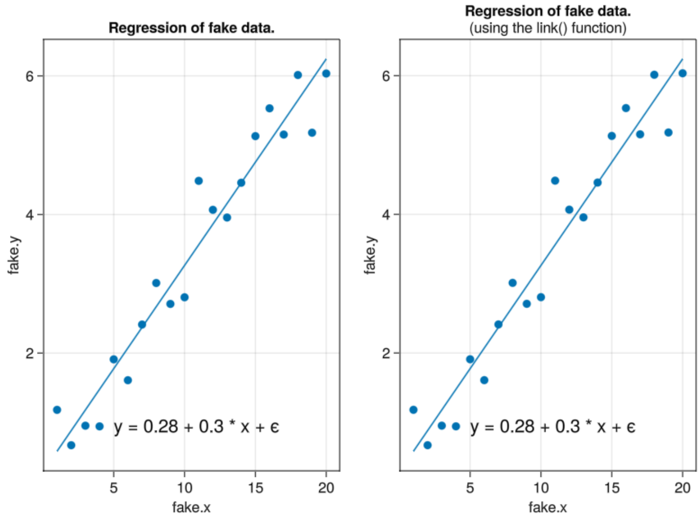
	parameters	mean	mcse	std
1	"a"	0.284945	0.00587742	0.230988
2	"b"	0.29781	0.000481178	0.019324
3	"sigma"	0.474901	0.00220934	0.090591

```
let
    data = (N=nrow(fake), x=fake.x,
    y=fake.y)
    global m6_1s = SampleModel("m6_1s",
    stan6_1)
    global rc6_1s = stan_sample(m6_1s; data)
    success(rc6_1s) && describe(m6_1s)
end
```

```
Informational Message: The current Metropolis
jected because of the following issue:
Exception: normal_lpdf: Scale parameter is 0,
r/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T/
7, column 1 to column 23)
If this warning occurs sporadically, such as f
types like covariance matrices, then the sampl
but if this warning occurs often then your mod
conditioned or misspecified.
```

	parameters	median	mad_sd	mean	st
1	"a"	0.282	0.229	0.285	0.23
2	"b"	0.298	0.018	0.298	0.01
3	"sigma"	0.462	0.082	0.475	0.09

```
if success(rc6_1s)
    post6_1s = read_samples(m6_1s,
    :dataframe)
    ms6_1s = model_summary(post6_1s, [:a,
    :b, :sigma])
end
```




```
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_1s[:a, :mean] .+  ms6_1s[:b, :mean] .* x
    lines!(x, y)
    â = round(ms6_1s[:a, :mean]; digits=2)
    b̂ = round(ms6_1s[:b, :mean]; 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_1s, (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	0.282	0.229
2	:b	0.3	0.298	0.018
3	:sigma	0.5	0.462	0.082

```
DataFrame(parameters = Symbol.(names(post6_1s)), simulated = [0.2, 0.3, 0.5], median = ms6_1s[:, :median], mad_sd = ms6_1s[:, :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

```
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

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

```
stan6_2 = "
data {
    int N;
    vector[N] male;
    vector[N] height;
    vector[N] earnk;
}
parameters {
    real a;
    real b;
    real c;
    real<lower=0> sigma;
}
model {
    vector[N] mu;
    sigma ~ exponential(1);
    mu = a + b * height + c * male;
    earnk ~ normal(mu, sigma);
}";
```

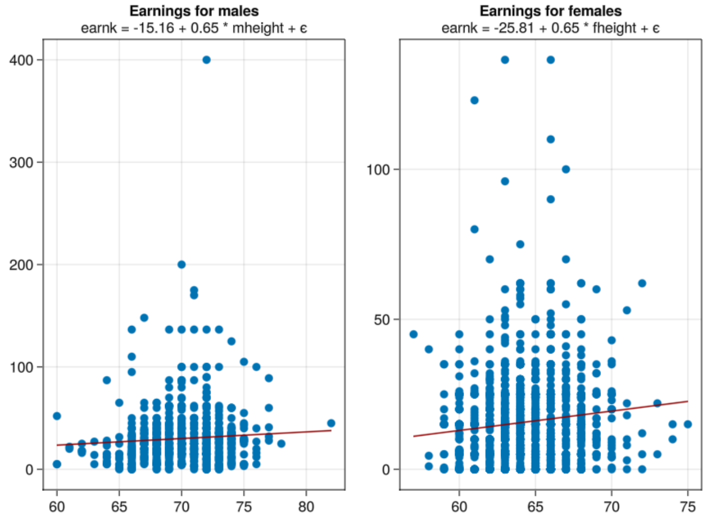
	parameters	mean	mcse	std
1	"a"	-25.8136	0.342803	11.8855
2	"b"	0.645866	0.00533156	0.184075
3	"c"	10.6543	0.0382441	1.48058
4	"sigma"	21.2877	0.00837708	0.35878

```
let
    data = (N=nrow(earnings), height=earnings.height,
    male=earnings.male,
    earnk=earnings.earnk)
    global m6_2s = SampleModel("m6_2s", stan6_2)
    global rc6_2s = stan_sample(m6_2s; data)
    success(rc6_2s) && describe(m6_2s)
end
```

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

	parameters	median	mad_sd	mean	st
1	"a"	-25.412	11.818	-25.814	11.8
2	"b"	0.64	0.184	0.646	0.18
3	"c"	10.678	1.469	10.654	1.48
4	"sigma"	21.274	0.366	21.288	0.35

```
if success(rc6_2s)
    post6_2s = read_samples(m6_2s,
    :dataframe)
    ms6_2s = model_summary(post6_2s, [:a,
    :b, :c, :sigma])
end
```




```
let
    â, b̂, ĉ = round.(ms6_2s[:, :mean];
    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_2s, (r,x) ->
    r.c + r.a + x * r.b, mheight_range))

    #earnk = ms6_2s[:c, "mean"] +
    ms6_2s[:a, "mean"] .+  ms6_2s[:b,
    "mean"] .* mheight
    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_2s, (r,x) ->
    r.a + x * r.b, fheight_range))
```

```
        lines!(fheight_range, earnk;
        color=:darkred)

        fig
    end
```

```
R2 = 0.10735394024175804
```

```
R2 = 1 - ms6_2s[:sigma, :mean]^2 /
std(earnings.earnk)^2
```

6.4 Historical origins of regression.

```
stan6_3 = "
data {
    int N;
    vector[N] m_height;
    vector[N] d_height;
}
parameters {
    real a;
    real b;
    real<lower=0> sigma;
}
model {
    vector[N] mu;
    sigma ~ exponential(1);
    mu = a + b * m_height;
    d_height ~ normal(mu, sigma);
}";
```

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)
```

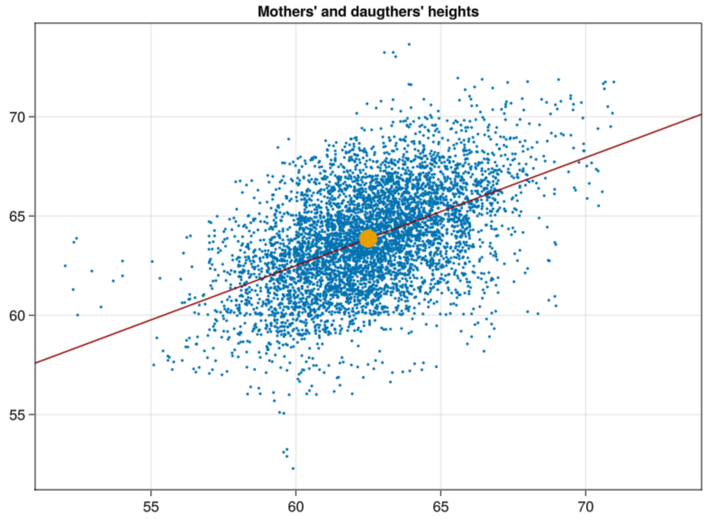
	parameters	mean	mcse	std
1	"a"	29.7912	0.0211552	0.756311
2	"b"	0.545054	0.000338452	0.01210
3	"sigma"	2.26304	0.000512356	0.021351

```
let
    data = (N=nrow(heights),
    m_height=heights.mother_height,
    d_height=heights.daughter_height)
    global m6_3s = SampleModel("m6_3s",
    stan6_3)
    global rc6_3s = stan_sample(m6_3s; data)
    success(rc6_3s) && describe(m6_3s)
end
```

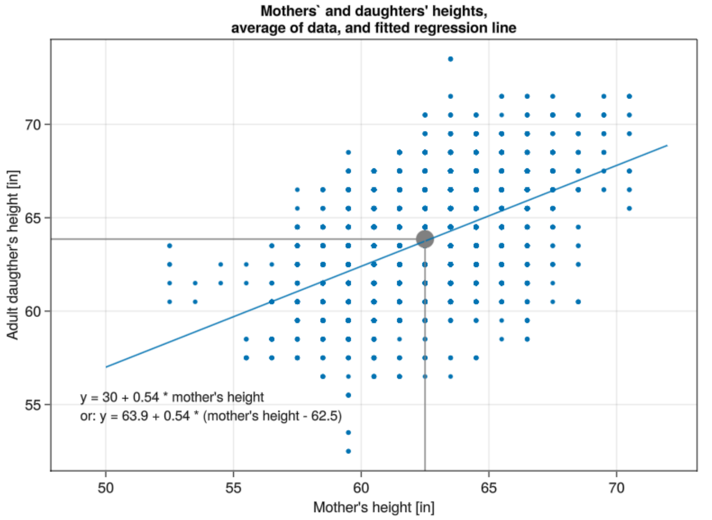
```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
d.
```

	parameters	median	mad_sd	mean	st
1	"a"	29.77	0.791	29.791	0.75
2	"b"	0.545	0.013	0.545	0.01
3	"sigma"	2.263	0.022	2.263	0.02

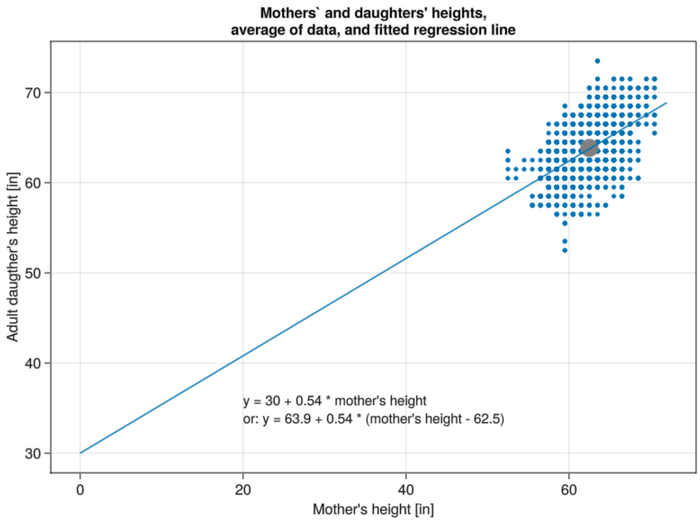
```
if success(rc6_3s)
    post6_3s = read_samples(m6_3s,
    :dataframe)
    ms6_3s = model_summary(post6_3s, [:a,
    :b, :sigma])
end
```




```
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_3s,
    (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
```




```julia
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
```

```
stan6_4 = "
data {
    int N;
    vector[N] m;
    vector[N] d;
}
parameters {
    real a;
    real b;
    real<lower=0> sigma;
}
model {
    vector[N] mu;
    a ~ normal(25, 3);
    b ~ normal(0, 0.5);
    sigma ~ exponential(1);
    mu = a + b * m;
    d ~ normal(mu, sigma);
}";
```

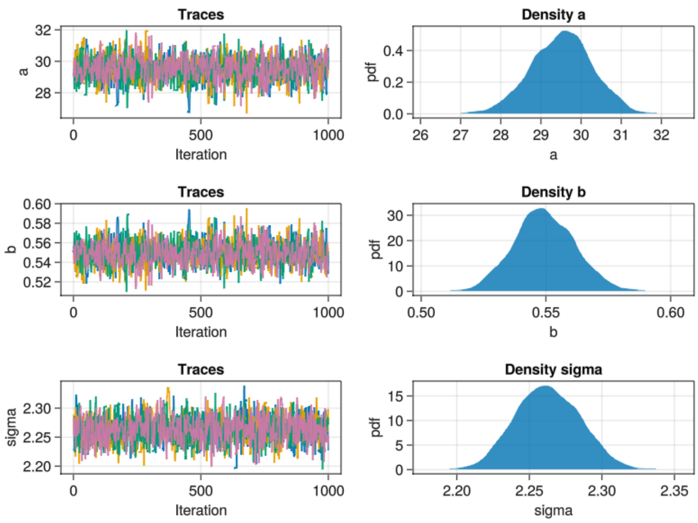
	parameters	mean	mcse	std
1	"a"	29.5157	0.0235803	0.769221
2	"b"	0.54945	0.000376458	0.0122975
3	"sigma"	2.26274	0.000620423	0.0223053

```
let
    data = (N = nrow(heights), m =
    heights.mother_height, d =
    heights.daughter_height)
    global m6_4s = SampleModel("m6_4s",
    stan6_4)
    global rc6_4s = stan_sample(m6_4s; data)
    success(rc6_4s) && describe(m6_4s)
end
```

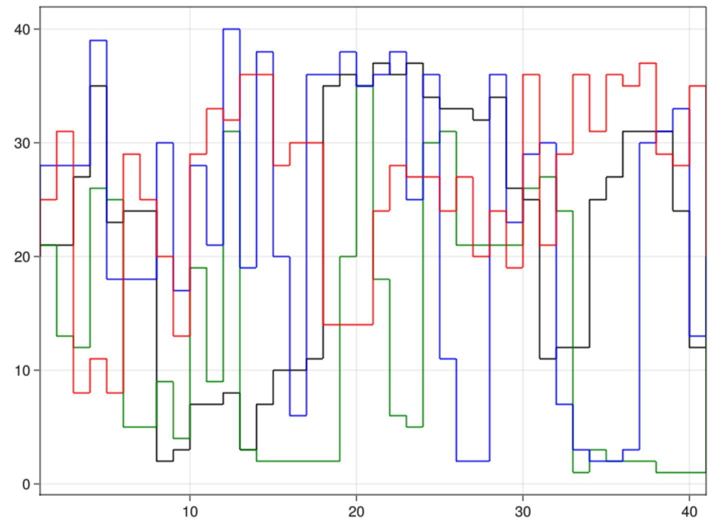
```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
d.
```

	parameters	median	mad_sd	mean	st
1	"a"	29.541	0.769	29.516	0.76
2	"b"	0.549	0.012	0.549	0.01
3	"sigma"	2.262	0.023	2.263	0.02

```
if success(rc6_4s)
    post6_4s = read_samples(m6_4s,
    :dataframe)
    ms6_4s = model_summary(post6_4s, [:a,
    :b, :sigma])
end
```




```
plot_chains(post6_4s, [:a, :b, :sigma])
```




```
trankplot(post6_4s, "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	39.0331	31.4929
2	58.12	51.6225
3	58.9264	47.0342
4	47.4465	35.3108
5	62.3286	38.493
6	50.4649	66.393
7	52.7304	44.1828
8	43.209	37.7982
9	23.9576	36.4188
10	55.5456	77.6372
⋮ more		
1000	40.0598	29.367

```julia
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
```

```
stan6_5 = "
data {
    int N;
    vector[N] midterm;
    vector[N] final;
}
parameters {
    real a;
    real b;
    real<lower=0> sigma;
}
model {
    vector[N] mu;
    sigma ~ exponential(1);
    mu = a + b * midterm;
    final ~ normal(mu, sigma);
}";
```

	parameters	mean	mcse	std
1	"a"	24.0196	0.0393881	1.40601
2	"b"	0.525487	0.000759792	0.02697
3	"sigma"	12.0779	0.00617686	0.261575

```
let
    data = (N=nrow(exams),
    midterm=exams.midterm,
    final=exams.final)
    global m6_5s = SampleModel("m6_5s",
    stan6_5)
    global rc6_5s = stan_sample(m6_5s; data)
    success(rc6_5s) && describe(m6_5s)
end
```

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

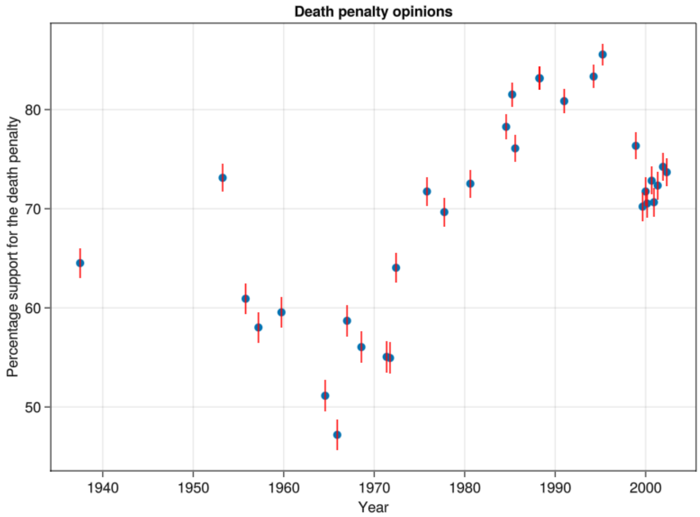
	parameters	median	mad_sd	mean	st
1	"a"	24.05	1.377	24.02	1.40
2	"b"	0.525	0.026	0.525	0.02
3	"sigma"	12.072	0.252	12.078	0.26

```
if success(rc6_5s)
    post6_5s = read_samples(m6_5s,
    :dataframe)
    ms6_5s = model_summary(post6_5s, [:a,
    :b, :sigma])
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)
```




```
begin
    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;
```