See chapter 4 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>
"""
```

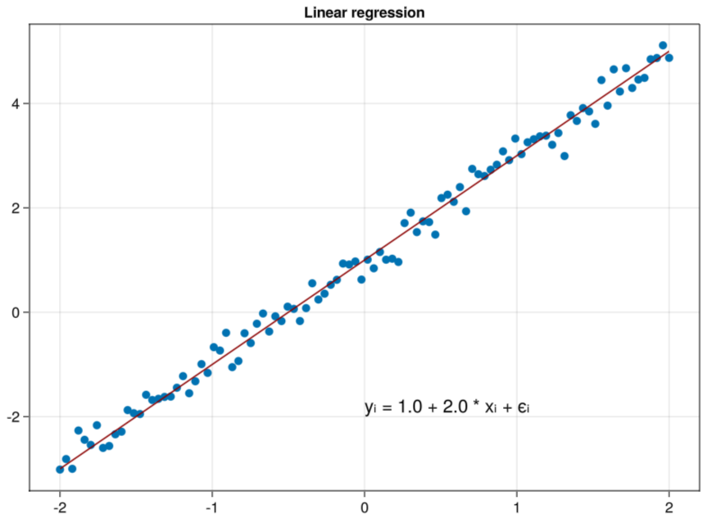
```
using Pkg ✓ , DrWatson ✓
```

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;
```

4.1 Sampling distributions and generative models.

```julia
begin
    Random.seed!(1)
    a = 1.0
    b = 2.0
    x = LinRange(-2, 2, 100)
    y = a .+ b .* x .+ rand(Normal(0.0,
    0.2), 100)
end;
```




```julia
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Linear
    regression")
    scatter!(x, y)
    lines!(x, a .+ b .* x; color=:darkred)
    annotations!("yᵢ = 1.0 + 2.0 * xᵢ +
    ϵᵢ", position=(0, -2), textsize=20)
    current_figure()
end
```

```
ppl4_1 (generic function with 2 methods)
```

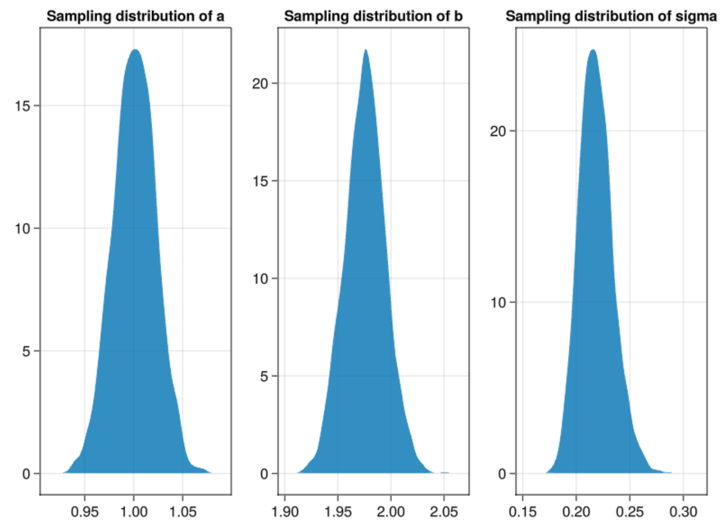
```julia
@model function ppl4_1(x, y)
    a ~ Normal(0, 1.5)
    b ~ Normal(1.0, 1.5)
    σ ~ Exponential(1)
    μ = a .+ b .* x
    for i in eachindex(y)
        y[i] ~ Normal(μ[i], σ)
    end
end
```

	parameters	mean	std	naive_
1	:a	1.00077	0.0218261	0.00034
2	:b	1.97562	0.0191309	0.00030
3	:σ	0.218557	0.0159491	0.00025

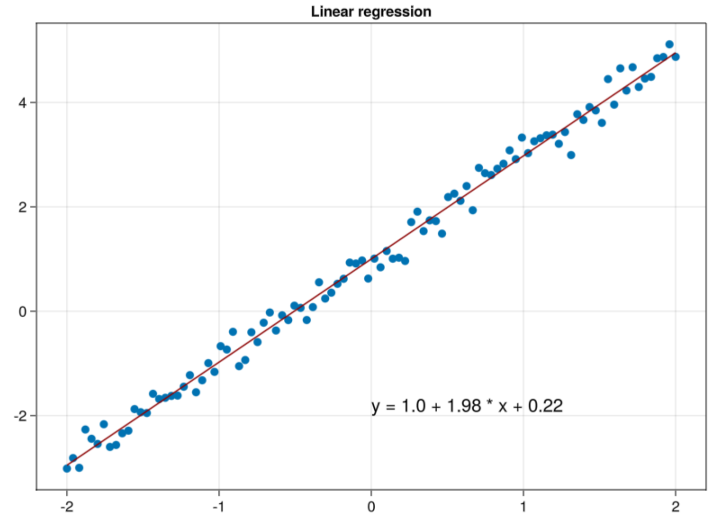
```
begin
    m4_1t = ppl4_1(x, y)
    chns4_1t = sample(m4_1t, NUTS(),
    MCMCThreads(), 1000, 4)
    describe(chns4_1t)
end
```

	parameters	median	mad_sd	mean	st
1	"a"	1.001	0.022	1.001	0.02
2	"b"	1.976	0.019	1.976	0.01
3	"σ"	0.218	0.016	0.219	0.01

```
begin
    post4_1t = DataFrame(chns4_1t)[:, 3:5]
    ms4_1t = model_summary(post4_1t,
    names(post4_1t))
end
```

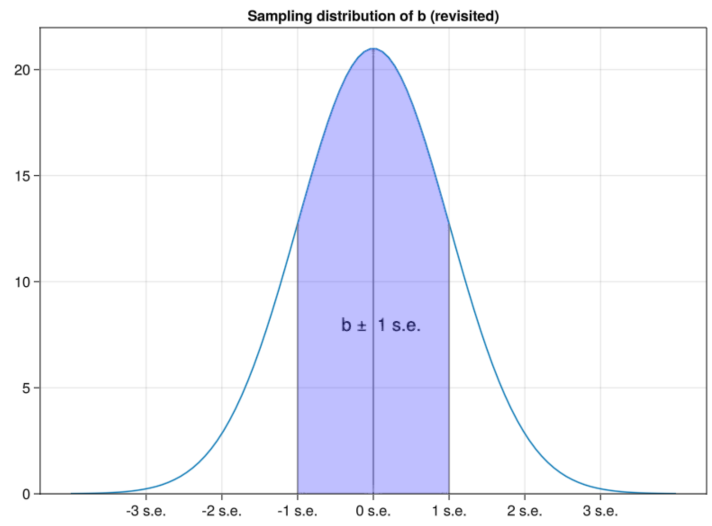



```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Sampling
    distribution of a")
    density!(post4_1t.a)
    ax = Axis(f[1, 2]; title="Sampling
    distribution of b")
    density!(post4_1t.b)
    ax = Axis(f[1, 3]; title="Sampling
    distribution of sigma")
    density!(post4_1t.σ)
    current_figure()
end
```

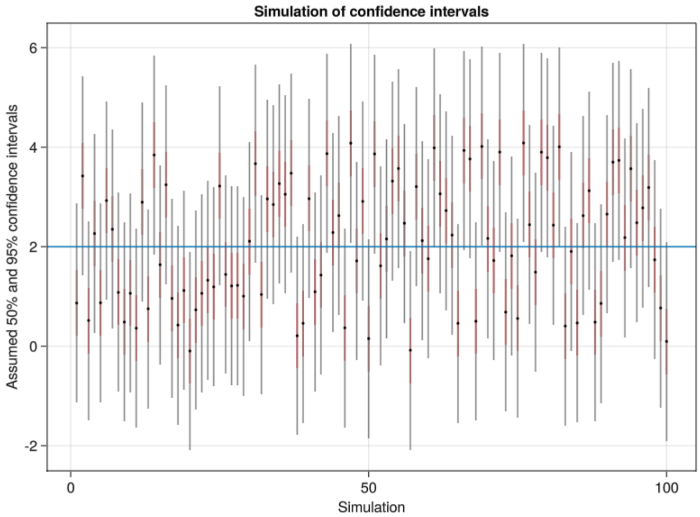



```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Linear
    regression")
    scatter!(x, y)
    lines!(x, ms4_1t["a", "median"] .+
    ms4_1t["b", "median"] .* x;
    color=:darkred)
    mean_a = round(ms4_1t["a", "mean"];
    digits=2)
    mean_b = round(ms4_1t["b", "mean"];
    digits=2)
    mean_σ = round(ms4_1t["σ", "mean"];
    digits=2)
    annotations!("y = $(mean_a) + $(mean_b)
    * x + $(mean_σ)", position=(0, -2),
    textsize=20)
    current_figure()
end
```

4.2 Estimates, standard errors, and confidence intervals.




```julia
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Sampling distribution of b (revisited)")
    b̂ = ms4_1t["b", "median"]
    σ̂ = ms4_1t["b", "std"]
    x = LinRange(b̂ - 4σ̂ , b̂ + 4σ̂, 100)
    y = pdf.(Normal(b̂, σ̂), x)
    ylims!(ax, [0, maximum(y) + 1.0])
    ax.xticks = b̂ - 3σ̂ : σ̂ : b̂ + 3σ̂
    ax.xtickformat = xs -> ["$(i) s.e." for i in -3:3]
    lines!(x, y)
    vlines!(ax, b̂;
    ymax=maximum(y)/(maximum(y) + 1.0), color=:grey)
    vlines!(ax, b̂-σ̂; ymax=pdf.(Normal(b̂, σ̂), b̂-σ̂)/(maximum(y) + 1.0), color=:grey)
    vlines!(ax, b̂+σ̂; ymax=pdf.(Normal(b̂, σ̂), b̂+σ̂)/(maximum(y) + 1.0), color=:grey)
    annotations!("b ±  1 s.e.", position=(b̂-0.008, 7.5), textsize=20)
    x1 = range(b̂ - σ̂ , b̂ + σ̂; length=60)
    band!(x1, fill(0, length(x1)), pdf.(Normal(b̂, σ̂), x1); color = (:blue, 0.25))
    f
end
```




```
let
    n = 100
    b = 2.0

    f = Figure()
    ax = Axis(f[1,1]; title="Simulation of confidence intervals",
    xlabel="Simulation",
        ylabel="Assumed 50% and 95% confidence intervals")

    x = 1:n
    y = [rand(Uniform(b - 2.1 , b + 2.1), 1)[1] for i in 1:n]

    # Assumed s.e. = 1.0
    lowerrors = fill(0.66, n)
    higherrors = fill(2, n)

    errorbars!(x, y, lowerrors, color = :red) # same low and high error
    errorbars!(x, y, higherrors, color = :grey) # same low and high error

    scatter!(x, y, markersize = 3, color = :black)
    hlines!(ax, [2])

    f
end
```

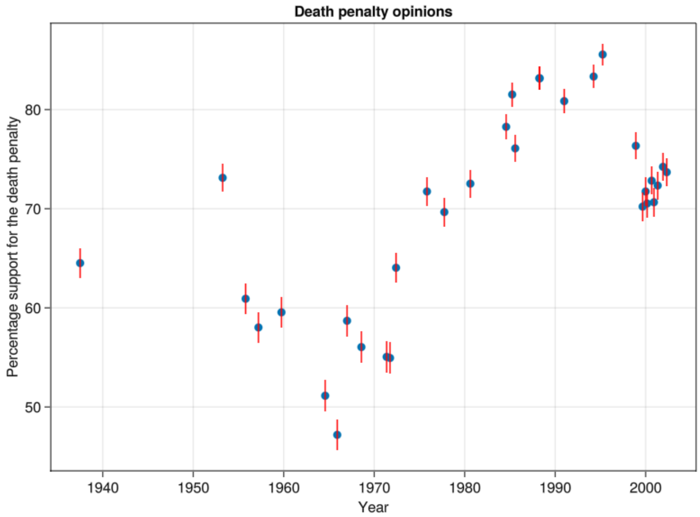
```
▸ (estimate = 0.7, se = 0.0144914, int_95 = [0.67
let
    n = 1000
    yes = 700
    no = n - yes
    est = yes/n
    se = sqrt(est * (1 - est)/n)

    (estimate = est, se = se, int_95 = est
    .+ quantile.(Normal(0, 1), [0.025,
    0.975]) * se)
end
```

```
▸ (estimate = 35.8, se = 0.734847, int_50 = [35.2
let
    y = [35, 34, 38, 35, 37]
    n = length(y)
    est = mean(y)
    se = std(y)/sqrt(n)
    int_50 = est .+ quantile.(TDist(n-1),
    [0.25, 0.75]) * se
    int_95 =  est .+ quantile.(TDist(n-1),
    [0.025, 0.975]) * se

    (estimate = est, se = se, int_50 =
    int_50, int_95 = int_95)
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)
    global 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
```

```
▸ [1.3925, 1.38313, 1.41453, 1.43996, 1.40676, 1.4
```

```
err_lims
```

4.3 Bias and unmodeled uncertainty.

4.4 Statistical significance, hypothesis testing, and statistical erros.

4.5 Problems with the concept of statistical significance.

4.6 Example of hypothesis testing: 55,000 residents need your help!

4.7 Moving beyond hypothesis testing.