### See chapter 8 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 ✓
    using Optim ✓

    # Specific to ROSStanPluto
    using StanSample ✓
    using StanOptimize ✓

    # Graphics related
    using GLMakie ✓

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

## 8.1 Least squares, maximum likelihood, and Bayesian inference.

| | x | y | ϵ | error |
|---|---|---|---|---|
| 1 | 0.0 | 45.9177 | -0.282333 | -0.282333 |
| 2 | 0.0251256 | 48.4013 | 2.12591 | 2.12591 |
| 3 | 0.0502513 | 43.1233 | -3.22741 | -3.22741 |
| 4 | 0.0753769 | 56.2541 | 9.82797 | 9.82797 |
| 5 | 0.100503 | 51.161 | 4.6595 | 4.6595 |
| 6 | 0.125628 | 47.6471 | 1.07026 | 1.07026 |
| 7 | 0.150754 | 53.652 | 6.99973 | 6.99973 |
| 8 | 0.175879 | 43.4236 | -3.30408 | -3.30408 |
| 9 | 0.201005 | 42.632 | -4.17101 | -4.17101 |
| 10 | 0.226131 | 45.5619 | -1.31654 | -1.31654 |
| ⋮ more | | | | |
| 200 | 5.0 | 57.0887 | -4.11126 | -4.11126 |

```
let
    Random.seed!(1)
    a = 46.2
    b = 3.0
    sigma = 4.0
    x = LinRange(0, 5, 200)
    ϵ = rand(Normal(0, sigma), length(x))
    y = a .+ b .* x .+ ϵ

    # DataFrame used to collect differen
    estimates, shown later on.

    global estimate_comparison = DataFrame()
    estimate_comparison.parameters = [:a,
    :b, :sigma]

    global sim = DataFrame(x = x, y = y, ϵ
    = ϵ, error = y .- (a .+ b .* x))
end
```

```
stan8_1 = "
data {
    int<lower=1> N;      // total number of
    observations
    vector[N] x;         // Independent
    variable: growth
    vector[N] y;         // 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(1, 5);
    b ~ normal(1, 5);
    sigma ~ exponential(1);

    mu = a + b * x;

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

| | parameters | mean | mcse | std |
|---|---|---|---|---|
| 1 | "b" | 3.2503 | 0.00490206 | 0.208701 |
| 2 | "a" | 45.6201 | 0.0141328 | 0.599706 |
| 3 | "sigma" | 4.37646 | 0.00496148 | 0.22432 |

```
let
    data = (N = nrow(sim), x = sim.x, y =
    sim.y)
    global m8_1s = SampleModel("m8_1s",
    stan8_1)
    global rc8_1s = stan_sample(m8_1s; data)
    success(rc8_1s) && describe(m8_1s)
end
```

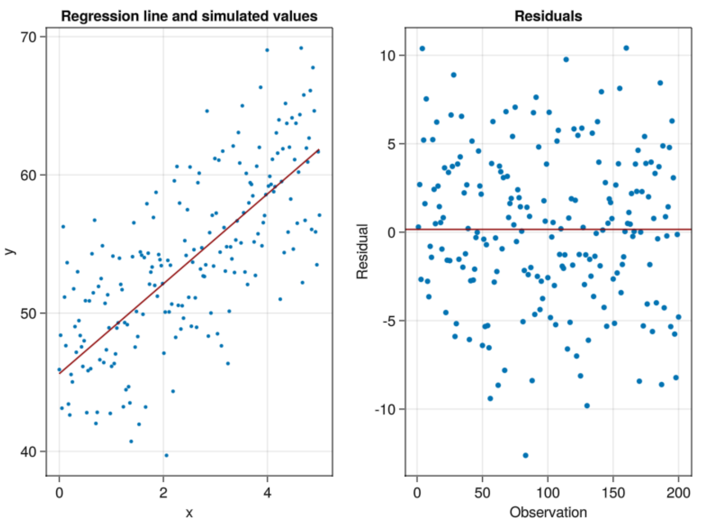
```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
d.
```

| | parameters | median | mad_sd | mean | st |
|---|---|---|---|---|---|
| 1 | "a" | 45.628 | 0.595 | 45.62 | 0.6 |
| 2 | "b" | 3.251 | 0.206 | 3.25 | 0.2( |
| 3 | "sigma" | 4.368 | 0.22 | 4.376 | 0.2: |

```
if success(rc8_1s)
    post8_1s = read_samples(m8_1s,
    :dataframe)
    ms8_1s = model_summary(post8_1s, [:a,
    :b, :sigma])
    estimate_comparison[!, :m8_1s] =
    [Vector(i) for i in eachrow(ms8_1s[:,
    [:median, :mad_sd]])]
    ms8_1s
end
```

| | x | y | ϵ | error |
|---|---|---|---|---|
| 1 | 0.0 | 45.9177 | -0.282333 | -0.282333 |
| 2 | 0.0251256 | 48.4013 | 2.12591 | 2.12591 |
| 3 | 0.0502513 | 43.1233 | -3.22741 | -3.22741 |
| 4 | 0.0753769 | 56.2541 | 9.82797 | 9.82797 |
| 5 | 0.100503 | 51.161 | 4.6595 | 4.6595 |
| 6 | 0.125628 | 47.6471 | 1.07026 | 1.07026 |
| 7 | 0.150754 | 53.652 | 6.99973 | 6.99973 |
| 8 | 0.175879 | 43.4236 | -3.30408 | -3.30408 |
| 9 | 0.201005 | 42.632 | -4.17101 | -4.17101 |
| 10 | 0.226131 | 45.5619 | -1.31654 | -1.31654 |
| ⋮ more | | | | |
| 200 | 5.0 | 57.0887 | -4.11126 | -4.11126 |

```
let
    â = ms8_1s[:a, :median]
    b̂ = ms8_1s[:b, :median]
    sim.residual = sim.y .- (â .+ b̂ .*
    sim.x)
    sim
end
```

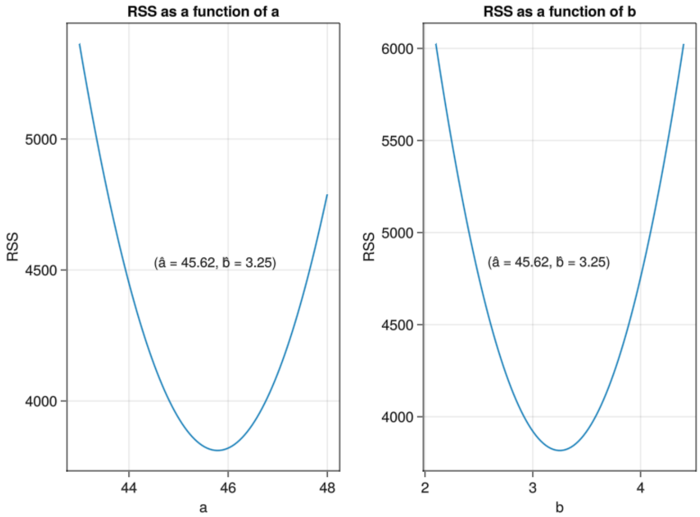



```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Regression line and simulated values", xlabel="x",
    ylabel="y")
    x_range = LinRange(minimum(sim.x),
    maximum(sim.x), 200)
    y_res = mean.(link(post8_1s, (r,x) ->
    r.a + x * r.b, x_range))
    scatter!(sim.x, sim.y; markersize=4)
    lines!(x_range, y_res; color=:darkred)

    ax = Axis(f[1, 2]; title="Residuals",
    xlabel="Observation", ylabel="Residual")
    scatter!(sim.residual; markersize=6)
    hlines!(ax, mean(sim.residual);
    color=:darkred)
    f
end
```

```
RSS = 3815.9793429532633
```

```
RSS = sum(sim.residual .^ 2)
```




```
let
    â = ms8_1s[:a, :mean]
    b̂ = ms8_1s[:b, :mean]

    f = Figure()
    ax = Axis(f[1, 1]; title="RSS as a
    function of a", xlabel="a",
    ylabel="RSS")
    a_range = LinRange(43, 48, 100)
    r = [sum((sim.y .- (k .+ b̂ .* sim.x))
    .^ 2) for k in a_range]
    lines!(a_range, r)
    annotations!("$((â = â, b̂ = b̂))",
    position=(44.5, 4500), textsize=15)
    ax = Axis(f[1, 2]; title="RSS as a
    function of b", xlabel="b",
    ylabel="RSS")
    b_range = LinRange(2.1, 4.4, 100)
    r = [sum((sim.y .- (â .+ k .* sim.x))
    .^ 2) for k in b_range]
    lines!(b_range, r)
    annotations!("$((â = â, b̂ = b̂))",
    position=(2.58, 4800), textsize=15)
    f
end
```

## Least squares

```
▸ (46.2831, 3.05172)
```

```
let
    global lsq = [0.0 missing; 0.0 missing;
    0.0 missing]
    df = DataFrame(ones = ones(nrow(sim)), x
     = sim.x)
    X = Array(df)
    Xt = transpose(X)
    â, b̂ = (Xt * X)^-1 * Xt * sim.y
    lsq[1, 1] = â
    lsq[2, 1] = b̂
    â, b̂
end
```

```
▸ (â = 46.2831, b̂ = 3.05172)
```

```
let
    b̂ = sum((sim.x .- mean(sim.x)) .*
    sim.y) / sum(((sim.x .- mean(sim.x)) .^
    2))
    â = mean(sim.y) - b̂ * mean(sim.x)
    (â = â, b̂ = b̂)
end
```

```
4.390059560439744
```

```
let
    σ̂ = sqrt(sum(sim.residual .^
    2)/(nrow(sim) - 2))
    lsq[3, 1] = σ̂
    estimate_comparison[!, :least_squares]
    = [Vector(i) for i in eachrow(lsq)]
    σ̂
end
```

### Maximum likelihood

```
loglik (generic function with 1 method)
```

```
function loglik(x)
    ll = 0.0
    ll += log(pdf(Normal(50, 20), x[1]))
    ll += log(pdf(Normal(2, 10), x[2]))
    ll += log(pdf(Exponential(1), x[3]))
    for i in 1:nrow(sim)
        ll += sum(logpdf.(Normal(x[1] .+
        x[2] .* sim.x[i], x[3]), sim.y[i]))
    end
    -ll
end
```

```
0.1353352832366127
```

```julia
pdf(Exponential(1), 2.0)
```

```
[170.0, 10.0, 2.0]
```

```julia
begin
    lower = [0.0, 0.0, 0.0]
    upper = [250.0, 50.0, 10.0]
    x0 = [170.0, 10.0, 2.0]
end
```

```
res =
 * Status: success (objective increased between :

 * Candidate solution
    Final objective value:     5.895739e+02

 * Found with
    Algorithm:     Fminbox with L-BFGS

 * Convergence measures
    |x - x'|               = 2.27e-08 ≰ 0.0e+00
    |x - x'|/|x'|          = 4.88e-10 ≰ 0.0e+00
    |f(x) - f(x')|         = 0.00e+00 ≤ 0.0e+00
    |f(x) - f(x')|/|f(x')| = 0.00e+00 ≤ 0.0e+00
    |g(x)|                 = 6.16e-09 ≤ 1.0e-08

 * Work counters
    Seconds run:   1  (vs limit Inf)
    Iterations:    5
    f(x) calls:    120
    ∇f(x) calls:   120
```

```julia
res = optimize(loglik, lower, upper, x0)
```

```
[46.2877, 3.05023, 4.30947]
```

```julia
let
    mle = Optim.minimizer(res)
    lsq[:, 1] = mle
    estimate_comparison[!, :mle] =
    [Vector(i) for i in eachrow(lsq)]
    mle
end
```

### MLE estimate (using StanOptimize and 4 chains)

| | a | b | sigma |
|---|---|---|---|
| 1 | 45.6309 | 3.24582 | 4.32164 |
| 2 | 45.6312 | 3.24574 | 4.32149 |
| 3 | 45.6313 | 3.24578 | 4.32148 |
| 4 | 45.6313 | 3.24577 | 4.32148 |

```
let
    data = (N=nrow(sim), y=sim.y, x=sim.x)
    o8_1s = OptimizeModel("m8_1s", stan8_1)
    rc8_1s = stan_optimize(o8_1s; data)
    result = success(rc8_1s) &&
    read_optimize(o8_1s)
    global o8_1_df = DataFrame()
    for p in ["a", "b", "sigma"]
        o8_1_df[!, p] = result[1][p]
    end
    o8_1_df
end
```

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

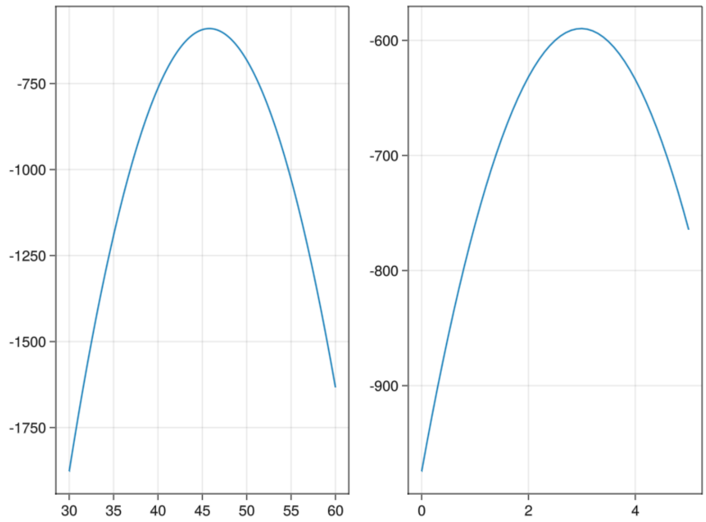
**Compare the four results.**

| | parameters | m8_1s | least_squa |
|---|---|---|---|
| 1 | :a | ▶ [45.628, 0.595] | ▶ [46.2831, m |
| 2 | :b | ▶ [3.251, 0.206] | ▶ [3.05172, m |
| 3 | :sigma | ▶ [4.368, 0.22] | ▶ [4.39006, m |

```
let
    lsq[:, 1] = mean(Array(o8_1_df); dims=1)
    estimate_comparison[!, :o8_1s] =
    [Vector(i) for i in eachrow(lsq)]
    estimate_comparison
end
```

```
590.2799283100538
```

```julia
loglik([45.6, 3.25, 4.4])
```




```julia
let
    f = Figure()
    ax = Axis(f[1, 1])
    lines!(30:0.1:60, [-loglik([a, 3.25,
    4.4]) for a in 30:0.1:60])
    ax = Axis(f[1, 2])
    lines!(0:0.1:5, [-loglik([46.5, b,
    4.4]) for b in 0:0.1:5])
    f
end
```

```
600.0086334504888
```

```julia
loglik([45, 3, 4.4])
```

```
2×200 Matrix{Float64}:
  1.0       1.0201   1.0402   1.0603   1.0804  …
 46.6171   43.4073  42.8262  48.5102  51.4102
```

```julia
let
    using StatsAPI
    Random.seed!(123)
    a = 46.2
    b = 3.0
    sigma = 4.0
    x = LinRange(1, 5, 200)
    ϵ = rand(Normal(0, sigma), length(x))
    y = a .+ b .* x .+ ϵ
    global obs = Matrix(hcat(x, y)')
end
```

```
distr8_1 =
FullNormal(
dim: 2
μ: [2.9999999999999996, 55.07524018977074]
Σ: [1.3467336683417086 3.9331584800491735; 3.933
)
```

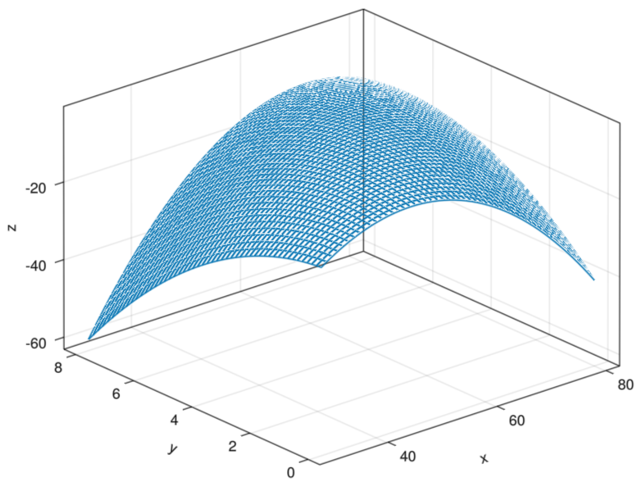
```julia
distr8_1 = fit_mle(MvNormal, obs)
```

```
2×1 Matrix{Float64}:
  3.000724768186341
 55.02344421495838
```

```julia
mean(rand(distr8_1, 1000); dims=2)
```

```
-3.389758334022121
```

```julia
loglikelihood(distr8_1, [3, 55])
```




```julia
let
    a = collect(LinRange(30, 80, 50))
    b = collect(LinRange(0, 8, 50))
    global z = [loglikelihood(distr8_1, [b,
    a]) for a in a, b in b]
    m, i = findmax(z)
    maxz = [a[i[1]], b[i[1]], z[i]]
    println(maxz)
    wireframe(a, b, z, axis=(type=Axis3,))
end
```

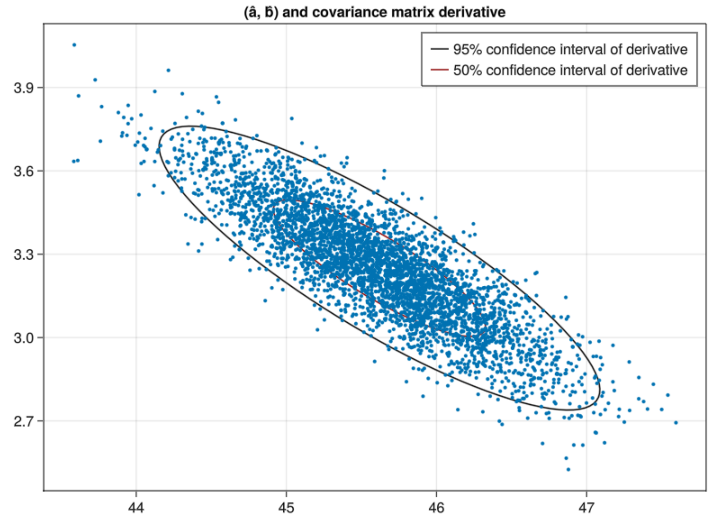
```
[55.51020408163265, 4.081632653061225, -3.39401
```

```
my_μ = ▸[45.62, 3.25]
```

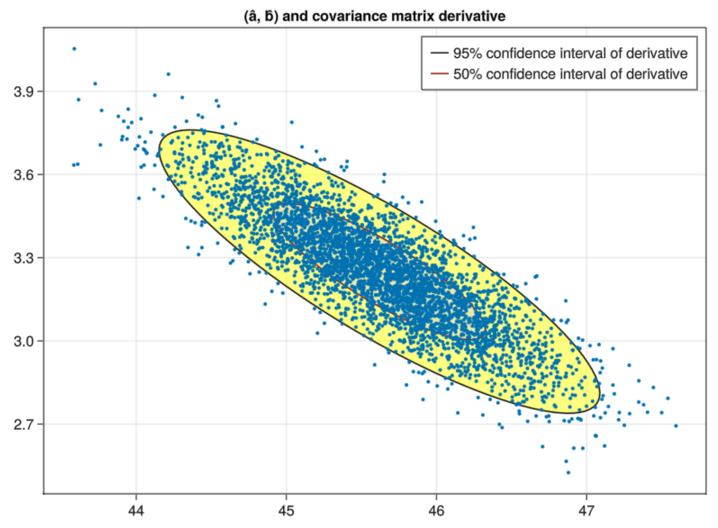
```julia
my_μ = [ms8_1s["a", "mean"], ms8_1s["b",
"mean"]]
```

```
my_Σ = 2×2 Matrix{Float64}:
  0.359647  -0.107326
 -0.107326   0.043556
```

```
my_Σ = cov([post8_1s.a post8_1s.b])
```

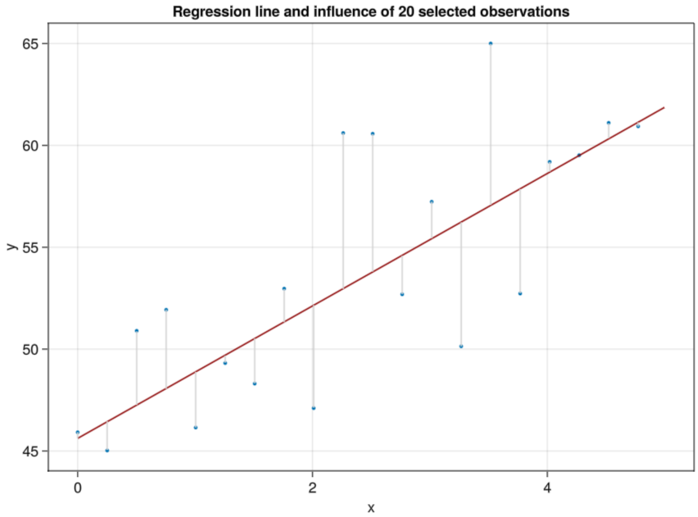



```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="(â, b̂) and
    covariance matrix derivative")
    lines!(getellipsepoints(my_μ, my_Σ)...,
    label="95% confidence interval of
    derivative", color=:black)
    lines!(getellipsepoints(my_μ, my_Σ,
    0.5)..., label="50% confidence interval
    of derivative", color=:darkred)
    scatter!(post8_1s.a, post8_1s.b;
    markersize=4)
    axislegend(position=:rt)
    f
end
```




```julia
let
    f = Figure()
    ax = Axis(f[1, 1]; title="(â, b̂) and covariance matrix derivative")
    poly!(Point2f.(zip(getellipsepoints(my_μ, my_Σ)...));
    color=(:yellow, 0.5))
    poly!(Point2f.(zip(getellipsepoints(my_μ, my_Σ, 0.50)...)); color=(:lightgrey, 0.5))
    lines!(getellipsepoints(my_μ, my_Σ)...,
    label="95% confidence interval of derivative", color=:black)
    lines!(getellipsepoints(my_μ, my_Σ, 0.5)..., label="50% confidence interval of derivative", color=:darkred)
    scatter!(post8_1s.a, post8_1s.b; markersize=4)
    axislegend(position=:rt)
    f
end
```

## 8.2 Influence of individual points in a fitted regression.




```
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Regression
    line and influence of 20 selected
    observations", xlabel="x", ylabel="y")
    x_range = LinRange(minimum(sim.x),
    maximum(sim.x), 200)
    y_res = mean.(link(post8_1s, (r,x) ->
    r.a + x * r.b, x_range))
    select_obs = 1:10:200
    scatter!(sim.x[select_obs],
    sim.y[select_obs]; markersize=4)
    lines!(x_range, y_res; color=:darkred)
    for ind in select_obs
        ymin = min(sim.y[ind], y_res[ind])
        ymax = max(sim.y[ind], y_res[ind])
        lines!([sim.x[ind], sim.x[ind]],
        [ymin, ymax]; color=:lightgrey)
    end
    f
end
```

## 8.3 Least squares slope as a weighted average of slopes of pairs.

```
(weighted_slopes = 3.05172, least_squares = [3.
```

```
let
    s1 = sum([(sim.x[i]-sim.x[j]) *
    (sim.y[i]-sim.y[j]) for i in
    1:length(sim.x), j in 1:length(sim.y)])
    s2 = sum([(sim.x[i]-sim.x[j])^2 for i in
     1:length(sim.x), j in 1:length(sim.y)])
    (weighted_slopes = round(s1/s2;
    digits=5),
    least_squares=estimate_comparison[2,
    :least_squares])
end
```

## 8.4 Comparing two fitting functions: `glm` and `stan_sample`.

```
stan8_2 = "
data {
    int<lower=1> N;      // total number of
    observations
    vector[N] x;         // Independent
    variable: growth
    vector[N] y;         // 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(0, 50);
    b ~ normal(0, 50);
    sigma ~ uniform(0, 50);

    mu = a + b * x;

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

| | parameters | mean | mcse | std |
|---|---|---|---|---|
| 1 | "b" | 5.05481 | 0.035551 | 1.26818 |
| 2 | "a" | -13.4538 | 0.209502 | 7.64578 |
| 3 | "sigma" | 11.1433 | 0.091969 | 3.43567 |

```
let
    x = LinRange(1, 10, 10)
    y = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
    global fake = DataFrame(x = x, y = y)
    data = (N = nrow(fake), x = fake.x, y =
    fake.y)
    global m8_2s = SampleModel("m8_2s",
    stan8_2)
    global rc8_2s = stan_sample(m8_2s; data)
    success(rc8_2s) && describe(m8_2s)
end
```

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

| | parameters | median | mad_sd | mean | st |
|---|---|---|---|---|---|
| 1 | "a" | -13.63 | 7.192 | -13.454 | 7.64 |
| 2 | "b" | 5.097 | 1.176 | 5.055 | 1.26 |
| 3 | "sigma" | 10.44 | 2.645 | 11.143 | 3.43 |

```
if success(rc8_2s)
    post8_2s = read_samples(m8_2s,
    :dataframe)
    ms8_2s = model_summary(post8_2s, [:a,
    :b, :sigma])
end
```

[2.45738, 7.50822]

```
quantile(post8_2s.b, [0.025, 0.975])
```

[2.96928, 7.04544]

```
quantile(post8_2s.b, [0.05, 0.95])
```

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

y ~ 1 + x

Coefficients:
──────────────────────────────────────────────────
                  Coef.  Std. Error      t  Pr(>|
──────────────────────────────────────────────────
(Intercept)  -13.8667      6.32766   -2.19    0.0
x              5.12121     1.01979    5.02    0.0
──────────────────────────────────────────────────
```

```
fake_lm = lm(@formula(y ~ x), fake)
```