

In Regression and Other Stories, mcmc is *just* a tool. Hence whether one uses Stan or Turing is not the main focus of the book.

This notebook uses

ElectionsEconomy: hibbs.csv to illustrate how Turing and other computations are used in the Julia *project* ROSTuringPluto.jl.

Over time I plan to expand below the list of topics:

1. Turing (see the Turing playground)
2. Using median and mad to summarize a posterior distribution.
3. ...
4. Model comparisons (TBD)
5. DAGs (TBD)
6. Graphs (TBD)
7. ...

Widen the cells.

```
• html"""
• <style>
•     main {
•         margin: 0 auto;
•         max-width: 2000px;
•         padding-left: max(160px, 10%);
•         padding-right: max(160px, 10%);
•     }
• </style>
• """
```

A typical set of Julia packages to include in notebooks.

```
• using Pkg ✓
```

```
• 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;
```

Note

All data files are available (as .csv files) in the data subdirectory of package RegressionAndOtherStories.jl.

```
"/Users/rob/.julia/packages/RegressionAndOtherStories.jl/data"
```

```
• ros_datadir()
```

Note

After evaluating above cell, use

```
ros_datadir("ElectionsEconomy", "hibbs.dat")
```

to obtain data.

```
hibbs =
```

	year	growth	vote	inc_party_candidate
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-30
growth	3.06053	0.696274	4.40	0.0004

```
• hibbs_lm = lm(@formula(vote ~ growth),  
  hibbs)
```

```
► [-8.99292, 2.66743, 1.0609, 2.20753, -5.89044, 4.27444]
```

```
• residuals(hibbs_lm)
```

```
2.2744434224582912
```

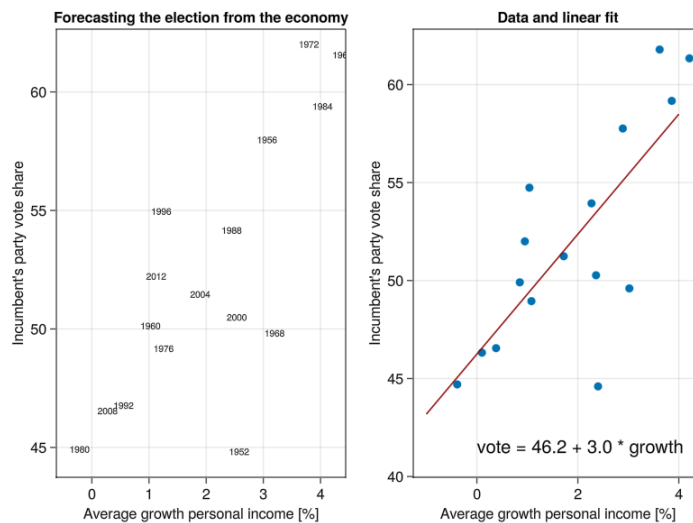
```
• mad(residuals(hibbs_lm))
```

```
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

```

Below some additional cells demonstrating the use of Turing.

ppl1_1 (generic function with 2 methods)

```
• @model function ppl1_1(growth, vote)
•   a ~ Normal(50, 20)
•   b ~ Normal(0, 5)
•   σ ~ Exponential(1)
•   μ = a .+ b .* growth
•   for i in eachindex(vote)
•       vote[i] ~ Normal(μ[i], σ)
•   end
• end
```

Note

The sequence of the statements matter in Turing models!

	parameters	mean	std	naive_se
1	:a	46.3462	1.56997	0.0248234
2	:b	3.00368	0.669857	0.0105914
3	:σ	3.59603	0.627185	0.00991667

```
• begin
•   m1_1t = ppl1_1(hibbs.growth, hibbs.vote)
•   chns1_1t = sample(m1_1t, NUTS(),
•   MCMCThreads(), 1000, 4)
•   describe(chns1_1t)
• end
```

Note

Mostly I disable logging early on in notebooks using Turing. But it is also possible to do this by `cell`. Click on the little circle with 3 dots at the top of the selected cell and select `Hide logs`.

post1_1t =

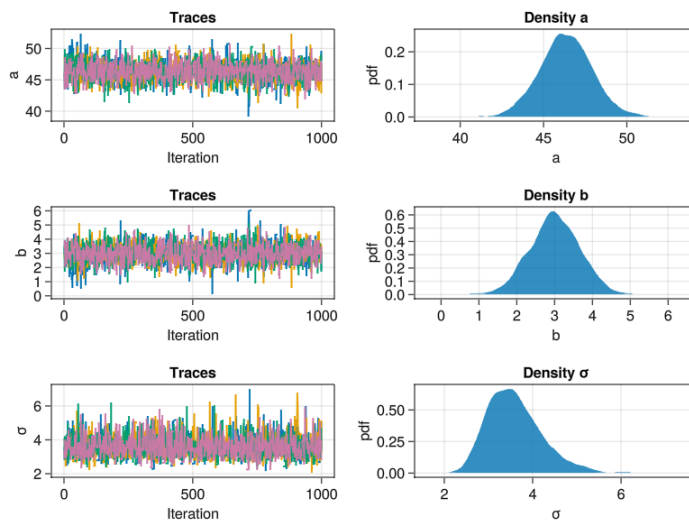
	a	b	σ
1	45.8543	2.97746	2.8181
2	44.6516	3.5725	4.1422
3	43.4512	3.90471	3.7756
4	49.171	2.07733	3.85881
5	48.2288	2.02114	3.61326
6	45.2312	3.51025	2.8635
7	44.071	3.05668	3.16067
8	46.8092	2.00857	3.99798
9	47.9348	3.09165	4.13494
10	47.0714	2.54041	3.46668
: more			
4000	47.6985	2.63067	2.81791

```
• post1_1t = DataFrame(chns1_1t)[: , [:a, :b, :σ]]
```

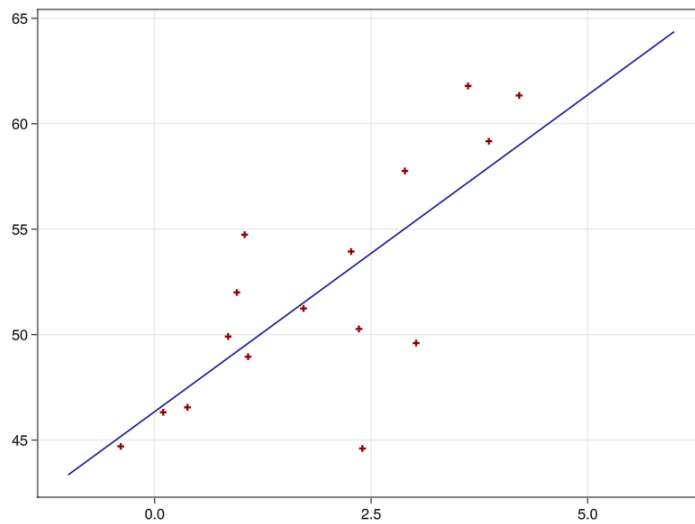
ms1_1t =

	parameters	median	mad_sd	mean	std
1	"a"	46.341	1.52	46.346	1.57
2	"b"	3.002	0.652	3.004	0.67
3	"σ"	3.52	0.6	3.596	0.62

```
• ms1_1t = model_summary(post1_1t, [:a, :b, :σ])
```



```
• plot_chains(post1_1t, [:a, :b, :sigma])
```



```
• let
•     x = -1.0:0.1:6.0
•     preds = mean(post1_1t.a) .+
•             mean(post1_1t.b) .* x
•     lines(x, preds, color=:darkblue,
•           label="Regression line")
•     scatter!(hibbs.growth, hibbs.vote,
•             marker=:cross, markersize=10,
•             color=:darkred,
•             label="Observations")
•     current_figure()
end
```

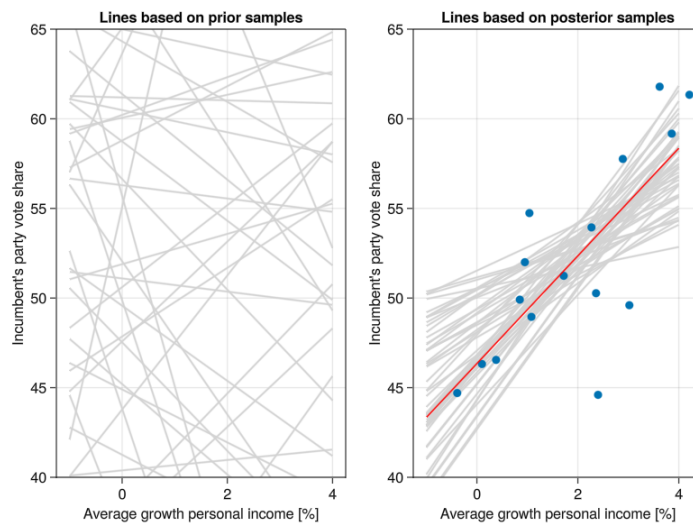
Priors of Turing models.

	parameters	mean	std	naive_se
1	:a	49.6665	21.0826	0.666692
2	:b	0.133146	4.94452	0.15636
3	: σ	1.0248	1.04171	0.0329417

```

• begin
•   prior_chns1_1t = sample(m1_1t, Prior(),
•     1000)
•   describe(prior_chns1_1t)
end

```



```

let
  N = 100
  x = LinRange(-1, 4, N)
  priors4_1t = DataFrame(prior_chns1_1t)

  mat1 = zeros(50, 100)
  for i in 1:50
    mat1[i, :] = priors4_1t.a[i] .+
      priors4_1t.b[i] .* x
  end
  ā = mean(post1_1t.a)
  b̄ = mean(post1_1t.b)

  # Maybe could use a 'link' function here
  mat2 = zeros(50, 100)
  for i in 1:50
    mat2[i, :] = post1_1t.a[i] .+
      post1_1t.b[i] .* x
  end

  fig = Figure()
  xlabel = "Average growth personal
income [%]"
ylabel="Incumbent's party vote share"
ax = Axis(fig[1, 1]; title="Lines based
on prior samples",
  xlabel, ylabel)
ylims!(ax, 40, 65)
series!(fig[1, 1], x, mat1,
  solid_color=:lightgrey)
ax = Axis(fig[1, 2]; title="Lines based
on posterior samples",
  xlabel, ylabel)
ylims!(ax, 40, 65)
series!(fig[1, 2], x, mat2,
  solid_color=:lightgrey)

```

```
scatter!(hibbs.growth, hibbs.vote)  
lines!(fig[1, 2], x,  $\bar{a} + \bar{b} * x$ , color  
= :red)  
  
fig  
end
```

Quadratic approximation.

```
map_estimate =  
ModeResult with maximized lp of -52.84  
3-element Named Vector{Float64}  
A  
├──  
└──  
:a 46.3459  
:b 3.01183  
:σ 3.213
```

- map_estimate = optimize(m1_1t, MAP())

- `map_estimate = optimize(m1_1t, MAP())`

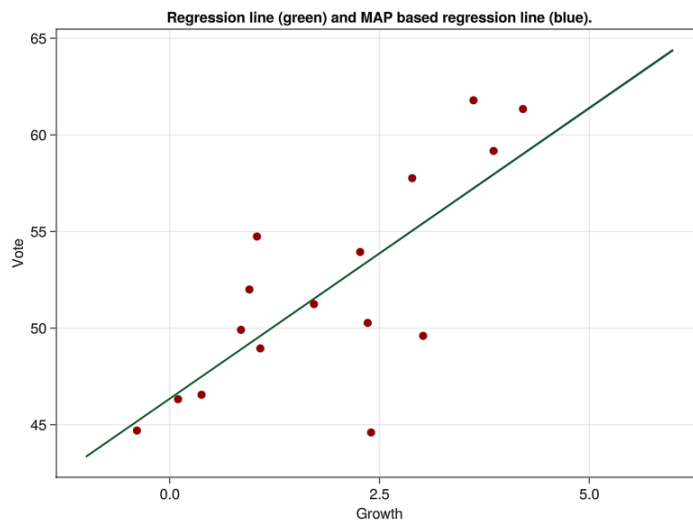
```
► ( $\hat{a}$  = 46.3459,  $\hat{b}$  = 3.01183,  $\hat{\sigma}$  = 3.213)
```

- begin
- \hat{a} , \hat{b} , $\hat{\sigma}$ = map_estimate.values
- (\hat{a} = \hat{a} , \hat{b} = \hat{b} , $\hat{\sigma}$ = $\hat{\sigma}$)
- end

```

• begin
•    $\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\boldsymbol{\sigma}} = \text{map\_estimate.values}$ 
•   ( $\hat{\mathbf{a}} = \hat{\mathbf{a}}, \hat{\mathbf{b}} = \hat{\mathbf{b}}, \hat{\boldsymbol{\sigma}} = \hat{\boldsymbol{\sigma}}$ )
• end

```



```

• let
•   x = -1:0.1:6
•   preds = mean(post1_1t.a) .+
•   mean(post1_1t.b) .* x
•   f = Figure()
•   ax = Axis(f[1, 1], title = "Regression
•   line (green) and MAP based regression
•   line (blue).",
•   xlabel = "Growth", ylabel = "Vote")
•
•   lines!(f[1, 1], x,  $\hat{a}$  .+  $\hat{b}$  .* x,
•   color=:darkblue)
•   lines!(x, preds, color=:darkgreen,
•   label="Regression line")
•   scatter!(hibbs.growth, hibbs.vote,
•   color=:darkred, leg=false)
•   current_figure()
• end

```

Prediction

	iteration	chain	vote[1]	vote[2]	vote[3]
1	1	1	45.8542	46.7915	47.1881
2	2	1	44.3899	47.6689	53.9884
3	3	1	46.4277	43.9124	44.7674
4	4	1	50.9526	46.8558	52.6576
5	5	1	50.1537	48.2343	47.8607
6	6	1	47.384	49.6843	49.374
7	7	1	43.7402	54.2323	45.0889
8	8	1	45.2788	49.6192	39.3787
9	9	1	46.6127	48.007	52.2737
10	10	1	47.6908	41.2327	50.7031
	: more				

```

• begin
•   x_test = [0, 1, 2, 3, 4, 5]
•   m_test = ppl1_1(x_test, fill(missing,
•   length(x_test)))
•   pred_chns1_1t = predict(m_test,
•   chns1_1t)
•   pred_chns1_1t
• end

```

	parameters	mean	std	naï
1	Symbol("vote[1]")	46.2443	3.97209	0.00
2	Symbol("vote[2]")	49.3039	3.8636	0.00
3	Symbol("vote[3]")	52.3164	3.75129	0.00
4	Symbol("vote[4]")	55.2959	3.87343	0.00
5	Symbol("vote[5]")	58.4459	3.99468	0.00
6	Symbol("vote[6]")	61.4424	4.30721	0.00

```

• describe(pred_chns1_1t)

```

	parameters	median	mad_sd	mean	std
1	"vote[1]"	46.34	3.861	46.244	3.97
2	"vote[2]"	49.367	3.762	49.304	3.86
3	"vote[3]"	52.299	3.606	52.316	3.75
4	"vote[4]"	55.277	3.704	55.296	3.87
5	"vote[5]"	58.395	3.815	58.446	3.99
6	"vote[6]"	61.491	4.171	61.442	4.30

```

• begin
•   pred1_1t = DataFrame(pred_chns1_1t)[: ,
•   3:end]
•   ps1_1t = model_summary(pred1_1t,
•   names(pred1_1t))
end

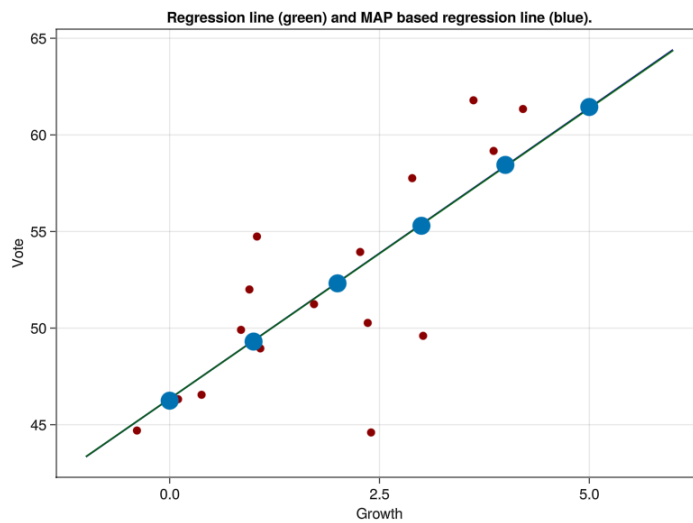
```

	vote[1]	vote[2]	vote[3]	vote[4]	vote[5]
1	45.8542	46.7915	47.1881	58.3118	56.1
2	44.3899	47.6689	53.9884	56.5369	63.1
3	46.4277	43.9124	44.7674	56.2496	58.1
4	50.9526	46.8558	52.6576	55.6043	56.1
5	50.1537	48.2343	47.8607	56.0421	57.1
6	47.384	49.6843	49.374	48.2467	60.1
7	43.7402	54.2323	45.0889	52.272	52.1
8	45.2788	49.6192	39.3787	56.9915	51.1
9	46.6127	48.007	52.2737	57.2739	58.1
10	47.6908	41.2327	50.7031	55.7591	57.1
⋮ more					
4000	52.632	52.0344	53.4976	61.7088	59.1

```

• pred1_1t

```



```

let
  x = -1:0.1:6
  preds = mean(post1_1t.a) .+
  mean(post1_1t.b) .* x
  f = Figure()
  ax = Axis(f[1, 1], title = "Regression
line (green) and MAP based regression
line (blue).",
  xlabel = "Growth", ylabel = "Vote")

  lines!(f[1, 1], x,  $\hat{a}$  .+  $\hat{b}$  .* x,
  color=:darkblue)
  lines!(x, preds, color=:darkgreen,
  label="Regression line")
  scatter!(hibbs.growth, hibbs.vote,
  color=:darkred, leg=false)
  scatter!(x_test,
  reshape(mean(Matrix(pred1_1t); dims=1),
  ncol(pred1_1t)), markersize=20)
  current_figure()
end

```

```
4000x6 Matrix{Float64}:
 45.8542  46.7915  47.1881  58.3118  56.8116  62
 44.3899  47.6689  53.9884  56.5369  63.4704  55
 46.4277  43.9124  44.7674  56.2496  58.8367  65
 50.9526  46.8558  52.6576  55.6043  56.4084  63
 50.1537  48.2343  47.8607  56.0421  57.0635  56
 47.384   49.6843  49.374   48.2467  60.3191  62
 43.7402  54.2323  45.0889  52.272   52.5568  62
  ⋮
 40.8571  52.016   47.0977  51.3776  60.1531  60
 46.7978  44.8882  49.5469  53.4672  55.0919  65
 48.2217  52.5758  51.6126  55.7905  54.4956  62
 51.8283  54.5989  48.9244  60.7026  58.49    60
 42.6466  48.4912  49.5714  56.0471  46.793   68
 52.632   52.0344  53.4976  61.7088  59.1616  64
```

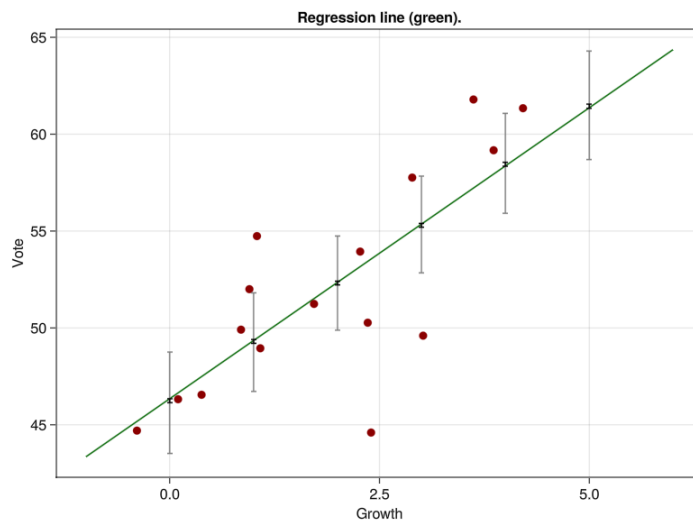
```
• Array(group(pred_chns1_1t, :vote))
```

	parameters	estimate	se	p
1	"vote[1]"	46.2443	0.0628042	▶ [0.055, (
2	"vote[2]"	49.3039	0.0610889	▶ [0.055, (
3	"vote[3]"	52.3164	0.059313	▶ [0.055, (
4	"vote[4]"	55.2959	0.0612443	▶ [0.055, (
5	"vote[5]"	58.4459	0.0631614	▶ [0.055, (
6	"vote[6]"	61.4424	0.068103	▶ [0.055, (

```
• errorbars_mean(pred1_1t)
```

	parameters	median	mad_sd	p
1	"vote[1]"	46.3402	3.86139	▶ [0.055, 0.9
2	"vote[2]"	49.3666	3.76239	▶ [0.055, 0.9
3	"vote[3]"	52.2987	3.60625	▶ [0.055, 0.9
4	"vote[4]"	55.2769	3.70398	▶ [0.055, 0.9
5	"vote[5]"	58.395	3.81504	▶ [0.055, 0.9
6	"vote[6]"	61.4909	4.17119	▶ [0.055, 0.9

```
• errorbars_draws(pred1_1t, [0.055, 0.945])
```

```

let
  x = -1:0.1:6
  preds = mean(post1_1t.a) .+
  mean(post1_1t.b) .* x
  pred_values =
  reshape(mean(Matrix(pred1_1t); dims=1),
  ncol(pred1_1t))

  f = Figure()
  ax = Axis(f[1, 1], title = "Regression
  line (green).",
  xlabel = "Growth", ylabel = "Vote")

  lines!(x, preds, color=:darkgreen,
  label="Regression line")
  scatter!(hibbs.growth, hibbs.vote,
  color=:darkred, leg=false)

  # 50% interval predictions
  errorBars =
  nested_column_to_array(errorbars_draws(p
  red1_1t, [0.25, 0.75]), "q")
  errorbars!(x_test, pred_values,
  errorBars[:, 1], errorBars[:, 2],
  whiskerwidth = 6, color=:grey)

  # 89% s.e. of the mean
  errorBars =
  nested_column_to_array(errorbars_mean(pr
  ed1_1t, [0.055, 0.945]), :q)
  errorbars!(x_test, pred_values,
  errorBars[:, 1], errorBars[:, 2],
  whiskerwidth = 6, color=:black)
  current_figure()
end

```

```
6×2 Matrix{Float64}:
 2.72267  2.51099
 2.58164  2.50705
 2.43288  2.42853
 2.45352  2.53978
 2.52928  2.62923
 2.75313  2.84876
```

- `nested_column_to_array(errorbars_draws(pred1_1t, [0.25, 0.75]), "q")`

A quick look at broadcasting and vectorization. See also [more dots](#)

f (generic function with 1 method)

- `f(x) = 3x^2 + 5x + 2`

nobcst (generic function with 1 method)

- `function nobcst(f, x)`
- `f.(2 .* x.^2 .+ 6 .* x.^3 .- sqrt.(x))`
- `end`

bcst (generic function with 1 method)

- `function bcst(f, x)`
- `@. f(2 * x^2 + 6 * x^3 - sqrt(x))`
- `end`

► [2.0, 1.99293, 1.99001, 1.98777, 1.98588, 1.9842]

- `let`
- `n = 10^6`
- `x = LinRange(0, 2, n)`
- `@time nobcst(f, x)`
- `end`

► [2.0, 1.99293, 1.99001, 1.98777, 1.98588, 1.9842]

- `let`
- `n = 10^6`
- `x = LinRange(0, 2, n)`
- `@time bcst(f, x)`
- `end`

Compute median and mad.

► [1.52, 0.652, 0.6]

- `[ms1_1t[v, "mad_sd"] for v in [:a, :b, :σ]]`

Alternative computation of mad().

```
► [1.51994, 0.652132, 0.599873]
```

```
• let  
• 1.483 .* [median(abs.(post1_1t.a .-  
• median(post1_1t.a))),  
• median(abs.(post1_1t.b .-  
• median(post1_1t.b))),  
• median(abs.(post1_1t.σ .-  
• median(post1_1t.σ)))]  
end
```

Quick simulation with median, mad, mean and std of Normal observations.

```
nt =
```

```
► (x = [0.897963, 2.87332, 4.87619, 6.41182, 5.49:
```

```
• nt = (x=rand(Normal(5, 2), 10000),)
```

```
► [5.02562, 2.02818, 5.00777, 2.02935]
```

```
• [median(nt.x), mad(nt.x), mean(nt.x),  
std(nt.x)]
```

```
sd_mean = 0.02
```

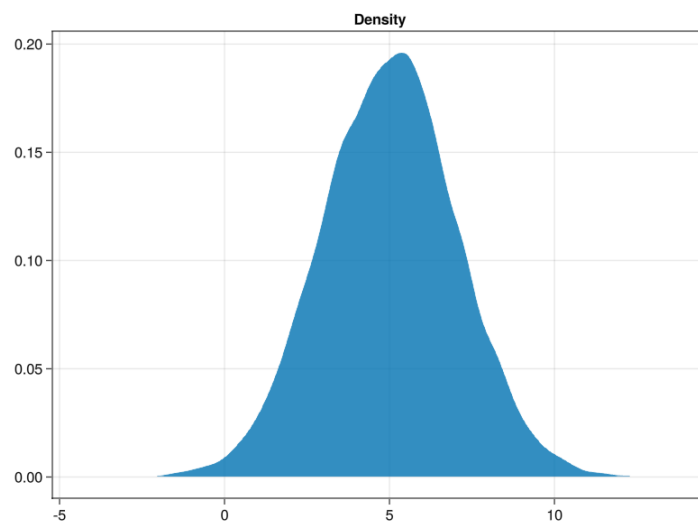
```
• sd_mean = round(mad(nt.x)/√10000; digits=2)
```

```
1.3679885818444575
```

```
• median(abs.(nt.x .- median(nt.x)))
```

```
2.0287270668753306
```

```
• 1.483 * median(abs.(nt.x .- median(nt.x)))
```



```
• let
•   fig = Figure()
•   ax = Axis(fig[1, 1]; title = "Density")
•   den = density!(nt.x)
•   fig
• end
```

► [1.04308, 8.98945]

```
• quantile(nt.x, [0.025, 0.975])
```

► [3.62681, 6.35746]

```
• quantile(nt.x, [0.25, 0.75])
```

Note

Click on "Live docs" and place cursor on link to see more help.

Click little down arrow to the right to remove live docs again.