Chap 7 Ulysses' Compass

```
#    margin-left: 1%;
#    margin-right: 5%;
html"""<style>
main {
    margin: 0 auto;
    max-width: 90%;
    padding-left: max(50px, 1%);
    padding-right: max(253px, 10%);
    # 253px to accomodate TableOfContents(aside=true)
}
"""
```

```
versioninfo()
```

```
Julia Version 1.10.2
Commit bd47eca2c8a (2024-03-01 10:14 UTC)
Build Info:
  Official https://julialang.org/ release
Platform Info:
  OS: Linux (x86_64-linux-gnu)
  CPU: 32 × Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-15.0.7 (ORCJIT, haswell)
Threads: 16 default, 0 interactive, 8 GC (on 32 virtual cores)
Environment:
  JULIA_PKG_SERVER = https://mirrors.tuna.tsinghua.edu.cn/julia
  JULIA_REVISE_WORKER_ONLY = 1
```

Selection deleted

```
using Pkg, DrWatson
```

Table of Contents


```
begin
  using PlutoUI
  TableOfContents()
```

Selection deleted

```
begin
    using Optim
    using GLM
    using CSV
    using Random
    using StatsBase
    using DataFrames
    using Dagitty
    using Turing
    using StatsPlots
    using StatisticalRethinking
    using StatisticalRethinkingPlots
    using ParetoSmoothedImportanceSampling
    import ParetoSmoothedImportanceSampling as psis
    import ParetoSmooth
    using Logging
    # DocStringExtensions provides $(SIGNATURES)
    using DocStringExtensions
    #using ProgressBars
    # have to import the student T distribution explicitly,
    as it is not exported
    import Distributions: IsoTDist
end
```

Selection deleted

link function from StatisticalRethinking.

- The variant that takes a function as input.

`link`

link

Generalized link function to evaluate callable for all parameters in dataframe over range of x values.

```
link(dfa, rx_to_val, xrange)
```

Required arguments

- `dfa::DataFrame`: data frame with parameters
- `rx_to_val::Function`: function of two arguments: row object and x
- `xrange`: sequence of x values to be evaluated on

Return values

Selection deleted

Is the vector, where each entry was calculated on each value from xrange. Every such entry is a list corresponding each row in the data frame.

Examples

```
julia> using StatisticalRethinking, DataFrames

julia> d = DataFrame(:a => [1,2], :b=>[1,1])
2×2 DataFrame
 Row │ a      b
     │ Int64  Int64
─────┼──────────────
   1 │     1      1
   2 │     2      1

julia> link(d, (r,x) -> r.a+x*r.b, 1:2)
2-element Vector{Vector{Int64}}:
 [2, 3]
 [3, 4]
```

```
"""
# link

Generalized link function to evaluate callable for all
parameters in dataframe over range of x values.

$(SIGNATURES)

## Required arguments
* `dfa::DataFrame`: data frame with parameters
* `rx_to_val::Function`: function of two arguments: row
object and x
* `xrange`: sequence of x values to be evaluated on

## Return values
Is the vector, where each entry was calculated on each
value from xrange.
Every such entry is a list corresponding each row in the
data frame.

## Examples
```jldoctest
```
```

```
julia> using StatisticalRethinking, DataFrames

julia> d = DataFrame(:a => [1,2], :b=>[1,1])
2×2 DataFrame
 Row │ a      b
     │ Int64  Int64
─────┼──────────────
   1 │     1      1
   2 │     2      1

julia> link(d, (r,x) -> r.a+x*r.b, 1:2)
2-element Vector{Vector{Int64}}:
 [2, 3]
 [3, 4]

```
"""
function link(dfa::DataFrame, rx_to_val::Function, xrange)
  [
    rx_to_val.(eachrow(dfa), (x,))
    for x ∈ xrange
  ]
```

7.0 How to display stdout/terminal output in Pluto.jl

Selection deleted

```
begin
    Plots.default(labels=false)
    # Disable the following line first.
    #Logging.disable_logging(Logging.Warn);
end;
```

- The following for-loop displays nothing because the loop returns a nothing object.
- Pluto.jl only displays returned object.

```
for i in 1:20
    R = i+100
    T = R^2
    md"hello $R and $T"
end
```

- the following for-loop will return and output the last iteration. but you can run it differently via map(...) do ... end

hello 120 and 14400

```
let
    x =  nothing
    for i in 1:20
        R = i+100
        T = R^2
        # something
        x = md"hello $R and $T"
    end
    x
end
```

- Output of every iteration of the for-loop is possible due to map(1:20)
```

```
[hello 101 and 10201, hello 102 and 10404, hello 103 and 10609,
```

```julia
map(1:20) do i
    R = i+100
    T = R^2
    md"hello $R and $T"
end
```

```
@with_stdout (macro with 1 method)
```

```julia
macro with_stdout(expr)
        escaped_expr = esc(expr)
    return quote
        stdout_bk = stdout
        rd, wr = redirect_stdout()
        result = ($escaped_expr)
        redirect_stdout(stdout_bk)
        close(wr)
        print_result = read(rd, String) |> Text
        (result, print_result)
    end
end
```

```julia
@with_stdout println("I am");
```

Selection deleted

After some tinkering (enable Logging.Warn), terminal output is now displayed in Pluto.jl

```julia
@time println("I am");
```

```
I am
  0.000103 seconds (18 allocations: 200.500 KiB)
```

```
@seeprints (macro with 1 method)
```

```julia
macro seeprints(expr)
    quote
        stdout_bk = stdout
        rd, wr = redirect_stdout()
        $expr
        redirect_stdout(stdout_bk)
        close(wr)
        read(rd, String) |> Text
    end
end
```

```julia
@seeprints println("I am");
```

7.1 The problem with parameters.

7.1 Load data & 7.2 Rescale data

```julia
begin
    sppnames = ["afarensis", "africanus", "habilis",
    "boisei",
        "rudolfensis", "ergaster", "sapiens"]
    brainvolcc = [438, 452, 612, 521, 752, 871, 1350]
    masskg = [37.0, 35.5, 34.5, 41.5, 55.5, 61.0, 53.5]
    d = DataFrame(:species => sppnames, :brain =>
    brainvolcc, :mass => masskg)
    d[!,:mass_std] = (d.mass .- mean(d.mass))./std(d.mass)
    d[!,:brain_std] = d.brain ./ maximum(d.brain)
    d
end;
```

Code 7.3 Regress brain on mass, optimized via MAP

To match results from the book, the model was optimized using MAP estimation, not the MCMC I used before. The reason for that is the MCMC producing different estimation for log_σ value, which makes all the values very different.

If you want, you can experiment with NUTS sampler for models in 7.1 and 7.2.

In addition, you can also check this discussion

```
model_m7_1 (generic function with 2 methods)
```

```
@model function model_m7_1(mass_std, brain_std)
    a ~ Normal(0.5, 1)
    b ~ Normal(0, 10)
    μ = @. a + b*mass_std
    log_σ ~ Normal()
    brain_std ~ MvNormal(μ, exp(log_σ))
end
```

Selection deleted

	variable	mean	min	median	max	nmissi
1	:a	0.53284	-0.000688679	0.534284	1.07771	0
2	:b	0.170031	-0.174648	0.16508	0.607495	0
3	:log_σ	-1.37747	-2.37392	-1.40897	0.42064	0

```
begin
    @time m7_1_ch = sample(model_m7_1(d.mass_std,
    d.brain_std), NUTS(), 1000)
    m7_1 = DataFrame(m7_1_ch)
    @time describe(m7_1)
end
```

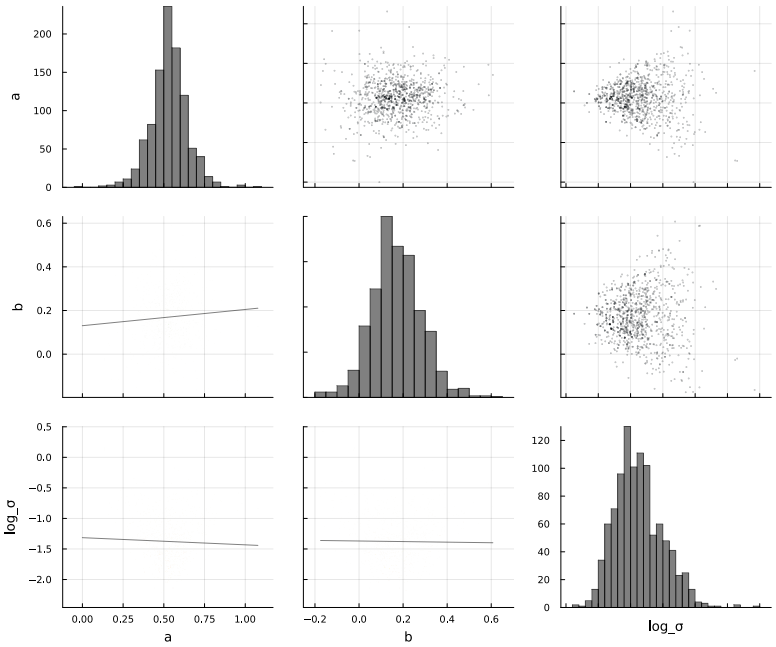
```
100%
```

```
Found initial step size
ϵ: 0.4
```

```
  4.247740 seconds (3.32 M allocations: 249.010 Mi
B, 2.40% gc time, 68.20% compilation time)
  0.000213 seconds (118 allocations: 30.766 KiB)
```

```
(1000, 3)
```

```
size(m7_1)
```




```
@df DataFrame(m7_1) corrplot(cols(1:3),
seriestype=:scatter, ms=0.2, alpha=0.5, size=(950, 800),
bins=30, grid=true)
```

Selection deleted

Checking the m7_1 Chains

```
md"### Checking the m7_1 Chains"
```

	iteration	chain	a	b	log_σ	lp	
1	501	1	0.509843	0.0890367	-2.04102	-5.61041	1
2	502	1	0.509843	0.0890367	-2.04102	-5.61041	2
3	503	1	0.555917	0.145738	-1.76824	-3.80884	2
4	504	1	0.555917	0.145738	-1.76824	-3.80884	2
5	505	1	0.530114	0.146504	-1.61682	-3.72554	2
6	506	1	0.775115	0.0608116	-1.60618	-9.86268	2
7	507	1	0.738181	0.189116	-1.27703	-6.51151	2
8	508	1	0.651913	0.117101	-0.36206	-9.33962	1
9	509	1	0.686841	0.36336	-1.38415	-7.40292	1
10	510	1	0.686841	0.36336	-1.38415	-7.40292	2

⋮ more

```
m7_1_ch
```

```
Chains{Float64, AxisArrays.AxisArray{Float64, 3, Array{Float64, 3
```

```
typeof(m7_1_ch)
```

```
▸ (varname_to_symbol = OrderedDict(a ⇒ :a, b ⇒ :b, log_σ ⇒ :log
```

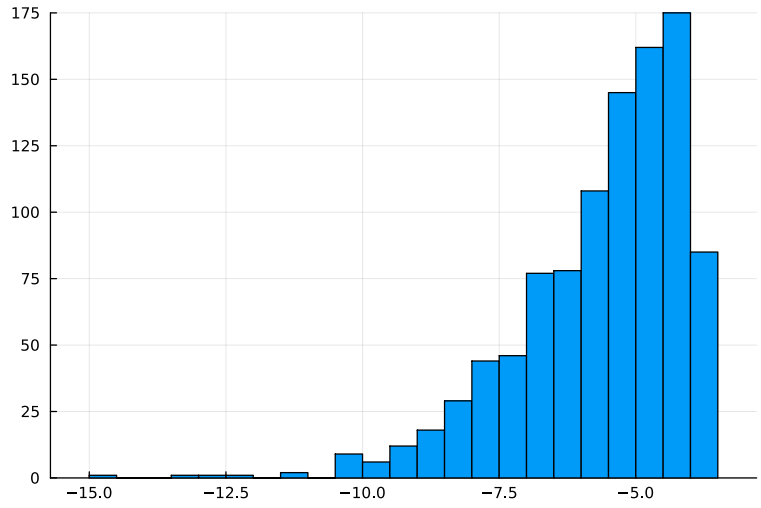
```
m7_1_ch.info
```

```
3-dimensional AxisArray{Float64,3,...} with axes:
    :iter, 501:1:1500
    :var, [:a, :b, :log_σ, :lp, :n_steps, :is_accept, :acceptance
    :chain, 1:1
And data, a 1000×15×1 Array{Float64, 3}:
[:, :, 1] =
 0.509843   0.0890367   -2.04102  …   8.92967   2.0  0.0  0.64834
 0.509843   0.0890367   -2.04102      4.27759   2.0  0.0  0.64834
 0.555917   0.145738    -1.76824      2.75461   2.0  0.0  0.64834
 0.555917   0.145738    -1.76824      0.677042  1.0  0.0  0.64834
 0.530114   0.146504    -1.61682      0.53324   2.0  0.0  0.64834
 0.775115   0.0608116   -1.60618  …   4.8479    2.0  0.0  0.64834
 0.738181   0.189116    -1.27703     -2.59115   2.0  0.0  0.64834
 ⋮                                   ⋮
 0.466506   0.0977929   -1.84985     -0.19042   2.0  0.0  0.64834
 0.524854   0.205494    -2.05463  …  -0.685438  2.0  0.0  0.64834
 0.524854   0.205494    -2.05463      4.72687   2.0  0.0  0.64834
 0.471906   0.225656    -1.67074      6.68196   2.0  0.0  0.64834
 0.518771   0.0432336   -1.29095      0.387717  2.0  0.0  0.64834
 0.491565  -0.00268598  -1.40619      0.211685  1.0  0.0  0.64834
```

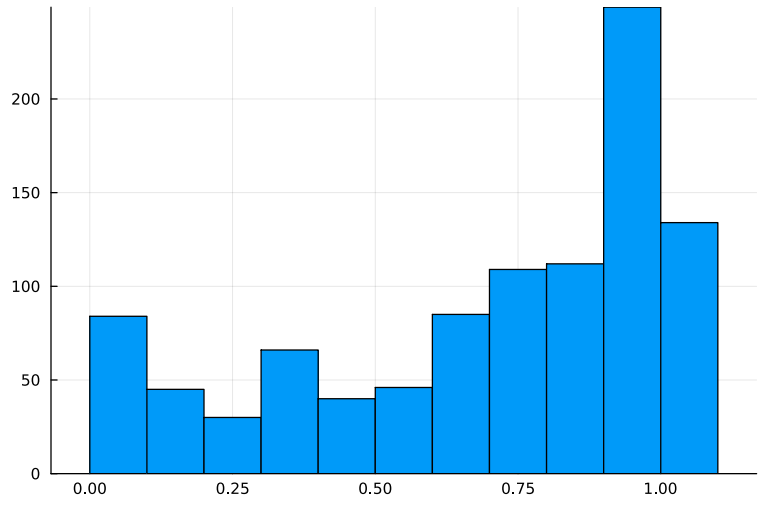
```
1 begin
2     @show m7_1_ch.name_map
3     m7_1_ch.value
4 end
```

```
m7_1_ch.name_map = (parameters = [:a, :b, :log_σ],
internals = [:lp, :n_steps, :is_accept, :acceptance_ra
te, :log_density, :hamiltonian_energy, :hamiltonian_en
ergy_error, :max_hamiltonian_energy_error, :tree_dept
h, :numerical_error, :step_size, :nom_step_size])
```

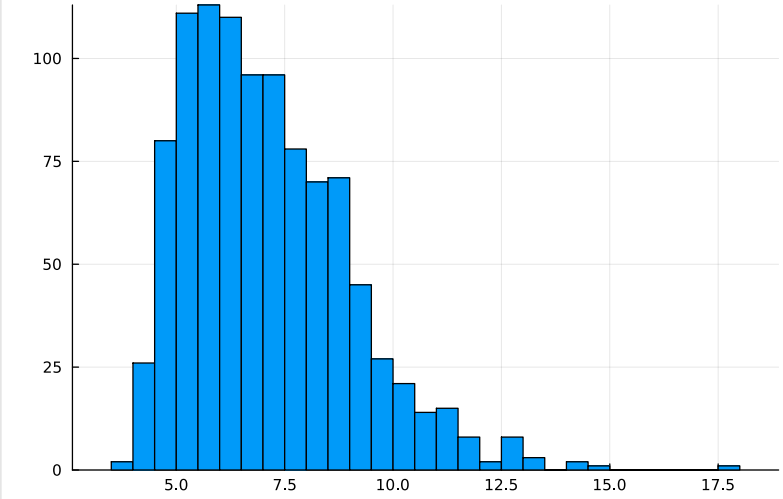
Selection (slider)



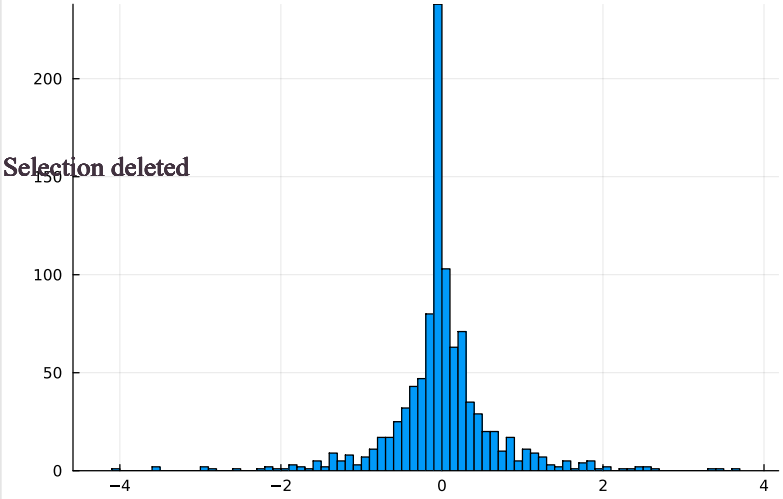

```
1 histogram(m7_1_ch[:log_density])
```



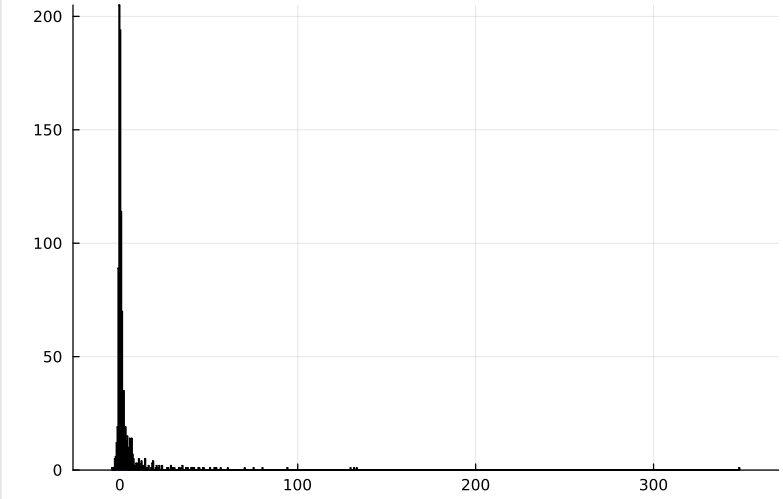

```
1 histogram(m7_1_ch[:acceptance_rate])
```




```
histogram(m7_1_ch[:hamiltonian_energy])
```




```
histogram(m7_1_ch[:hamiltonian_energy_error])
```




```
histogram(m7_1_ch[:max_hamiltonian_energy_error])
```

```
2-dimensional AxisArray{Float64,2,...} with axes:
    :iter, 501:1:1500
    :chain, 1:1
And data, a 1000×1 Matrix{Float64}:
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
 ⋮
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
 0.6483448249395375
```

```
m7_1_ch[:nom_step_size]
```

Code 7.4 OLS to obtain posterior distribution of a & b, but not σ

```
X = 7×2 Matrix{Float64}:
     1.0  -0.779467
     1.0  -0.91702
     1.0  -1.00872
     1.0  -0.866808
     1.0   0.91702
     1.0   1.42138
     1.0   0.733616
```

Selection deleted

```
X = hcat(ones(length(d.mass_std)), d.mass_std)
```

```
m =
GLM.LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Fl

Coefficients:
──────────────────────────────────────────────────────────────────
       Coef.  Std. Error      t  Pr(>|t|)   Lower 95%  Upper 95%
──────────────────────────────────────────────────────────────────
x1  0.528677   0.0705692  7.49     0.0007   0.347273    0.710081
x2  0.167118   0.0762235  2.19     0.0798  -0.0288204   0.363057
──────────────────────────────────────────────────────────────────
```

```
m = lm(X, d.brain_std)
```

Code 7.5 calculate R^2

```
Random.seed!(12);
```

Do explicit simulation due to log_σ.

- Why?

0.48838799760488294

```
@time begin
    # d.mass_std is a 7-element vector.
    s = [
        rand(MvNormal((@. r.a + r.b * d.mass_std),
        exp(r.log_σ)))
        for r ∈ eachrow(m7_1)
    ]
    # s is a vector of length 1000. each element is a 7-
    element vertical vector.
    @show size(s)
    # transpose each element of s to horizontal vector and
    then vertically concatenate them.
    s = vcat(s'...);
    @show size(s)

    r = mean.(eachcol(s)) .- d.brain_std;
    @show size(r)
    @show resid_var = var(r, corrected=false)
    outcome_var = var(d.brain_std, corrected=false)
    @show outcome_var
    1 - resid_var/outcome_var
end
```

```
size(s) = (1000,)
size(s) = (1000, 7)
size(μ) = (7,)
resid_var = var(r, corrected = false) = 0.024986533810
43236
outcome_var = 0.048838834299151763
  0.004239 seconds (18.69 k allocations: 2.092 MiB)
```

simulate(df::**DataFrames.DataFrame**, rx_to_dist::**Function**, xrange; *return_dist*, *seed*) in StatisticalRethinking at /y/home/huangyu/.julia/packages/StatisticalRethinking/RYYWV/src/simula

```
@which simulate(m7_1, (r, x) -> Normal(r.a + r.b * x,
exp(r.log_σ)), d.mass_std)
```

```
IsoNormal(
dim: 3
μ: [1.0, 2.0, 3.0]
Σ: [0.010000000000000002 0.0 0.0; 0.0 0.010000000000000002 0.0; 0
)
```

```
MvNormal([1,2,3], 0.1)
```

Code 7.6 Why R^2 is bad?

Function is implemented in a generic way to support any amount of b[x] coefficients.

```
R2_is_bad (generic function with 1 method)
```

```
function R2_is_bad(df; sigma=missing)
    degree = ncol(df[!,r"b"])
    # build mass_std*degree matrix, with each col
exponentiated to col's index
    t = repeat(d.mass_std, 1, degree)
    t = hcat(map(.^, eachcol(t), 1:degree)...)
    s = [
        begin
            # calculate product on coefficient's vector
            b = collect(r[r"b"])
            μ = r.a .+ t * b
            s = ismissing(sigma) ? exp(r.log_σ) : sigma
            rand(MvNormal(μ, s))
        end
        for r ∈ eachrow(df)
    ]
    s = vcat(s'...);

Selection deleted
    r = mean.(eachcol(s)) .- d.brain_std;
    v1 = var(r, corrected=false)
    v2 = var(d.brain_std, corrected=false)
    1 - v1 / v2
end
```

Code 7.7 2nd-degree polynomial

```
model_m7_2 (generic function with 2 methods)
```

```
@model function model_m7_2(mass_std, brain_std)
    a ~ Normal(0.5, 1)
    b ~ MvNormal([0, 0], 10)
    μ = @. a + b[1]*mass_std + b[2]*mass_std^2
    log_σ ~ Normal()
    brain_std ~ MvNormal(μ, exp(log_σ))
end
```

	variable	mean	min	median	max
1	:a	0.607718	-1.40189	0.606191	2.37658
2	Symbol("b[1]")	0.194592	-2.13629	0.195842	1.44602
3	Symbol("b[2]")	-0.0968602	-2.14742	-0.0949097	1.66151
4	:log_σ	-1.27554	-2.24776	-1.31453	1.32956

```
begin
    @time m7_2_ch = sample(model_m7_2(d.mass_std,
    d.brain_std), NUTS(), 10000)
    @time m7_2 = DataFrame(m7_2_ch)
    @time describe(m7_2)
end
```

```
100%
```

```
Found initial step size
ϵ: 0.4
```

```
  8.853600 seconds (16.16 M allocations: 1.366 GiB,
4.16% gc time, 36.34% compilation time)
  0.000926 seconds (113 allocations: 631.438 KiB)
  0.000739 seconds (137 allocations: 320.500 KiB)
```

Selection

Code 7.8 3-, 4-, 5-degree polynomial fitting: `brain_std ~ mass_std`

Implemented the sample in a general way.

```
model_m7_n (generic function with 2 methods)
```

```
@model function model_m7_n(mass_std, brain_std; degree::Int)
    a ~ Normal(0.5, 1)
    b ~ MvNormal(zeros(degree), 10)
    # build matrix n*degree
    t = repeat(mass_std, 1, degree)
    # exponent its columns
    t = hcat(map(.^, eachcol(t), 1:degree)...)
    # calculate product on coefficient's vector
    μ = a .+ t * b

    log_σ ~ Normal()
    brain_std ~ MvNormal(μ, exp(log_σ))
end
```

	variable	mean	min	median	max	n
1	:a	0.545904	-1.25153	0.532392	1.72525	0
2	Symbol("b[1]")	0.35406	-1.29638	0.363896	1.9635	0
3	Symbol("b[2]")	0.0248361	-2.06992	0.0343043	1.93808	0
4	Symbol("b[3]")	-0.169126	-1.73209	-0.170413	1.42126	0
5	:log_σ	-1.26632	-2.14229	-1.29211	0.937954	0

```
begin
    @time m7_3_ch = sample(model_m7_n(d.mass_std,
    d.brain_std, degree=3), NUTS(), 1000)
    @time m7_3 = DataFrame(m7_3_ch)
    describe(m7_3)
end
```

100%

Found initial step size
ϵ: 0.2

```
  5.452340 seconds (5.31 M allocations: 495.050 Mi
B, 3.64% gc time, 58.82% compilation time)
  0.000095 seconds (122 allocations: 86.250 KiB)
```

Selection

	variable	mean	min	median	max	n
1	:a	0.753135	-1.29212	0.758126	2.73901	0
2	Symbol("b[1]")	0.700627	-2.67157	0.633939	3.60469	0
3	Symbol("b[2]")	-0.684327	-4.93303	-0.586031	4.73013	0
4	Symbol("b[3]")	-0.589297	-3.85836	-0.538768	3.62783	0
5	Symbol("b[4]")	0.479884	-3.22847	0.430074	3.56131	0
6	:log_σ	-1.14958	-1.94352	-1.23606	0.812233	0

```
begin
    @time m7_4_ch = sample(model_m7_n(d.mass_std,
    d.brain_std, degree=4), NUTS(), 1000)
    @time m7_4 = DataFrame(m7_4_ch)
    describe(m7_4)
end
```

100%

Found initial step size
ϵ: 0.4

```
  5.442058 seconds (7.28 M allocations: 865.526 Mi
B, 4.55% gc time, 33.67% compilation time)
  0.000106 seconds (139 allocations: 102.828 KiB)
```

	variable	mean	min	median	max
1	:a	0.807506	-1.39643	0.813605	3.64389
2	Symbol("b[1]")	1.36051	-5.80807	1.42124	7.8659
3	Symbol("b[2]")	-0.448937	-9.47283	-0.336753	7.13566
4	Symbol("b[3]")	-2.11072	-16.9476	-2.28918	14.9686
5	Symbol("b[4]")	0.0860457	-7.38378	-0.0594394	7.71821
6	Symbol("b[5]")	0.779426	-8.71103	0.887674	6.77843
7	:log_σ	-1.02331	-1.89414	-1.07645	0.936759

```
begin
    m7_5_ch = sample(model_m7_n(d.mass_std, d.brain_std,
    degree=5), NUTS(), 1000)
    m7_5 = DataFrame(m7_5_ch)
    describe(m7_5)
end
```

```
100%
```

```
Found initial step size
ϵ: 0.025
```

Selection deleted

Code 7.9: 6-degree polynomial

```
model_m7_6 (generic function with 2 methods)
```

```
@model function model_m7_6(mass_std, brain_std)
    a ~ Normal(0.5, 1)
    b ~ MvNormal(zeros(6), 10)
    μ = @. a + b[1]*mass_std + b[2]*mass_std^2 +
b[3]*mass_std^3 +
               b[4]*mass_std^4 + b[5]*mass_std^5 +
b[6]*mass_std^6
    brain_std ~ MvNormal(μ, 0.001)
end
```

	variable	mean	min	median	max
1	:a	0.445913	-0.749119	0.505061	0.525469
2	Symbol("b[1]")	0.773959	-1.34346	0.878036	0.915289
3	Symbol("b[2]")	2.00922	1.59882	1.71208	8.04338
4	Symbol("b[3]")	-0.324458	-0.705828	-0.602745	5.33837
5	Symbol("b[4]")	-3.9359	-12.9266	-3.49367	-3.3188
6	Symbol("b[5]")	-0.532412	-4.17428	-0.353911	-0.283259
7	Symbol("b[6]")	1.83423	1.55421	1.63434	5.90625

```
begin
    m7_6_ch = sample(model_m7_6(d.mass_std, d.brain_std),
    NUTS(), 1000)
    m7_6 = DataFrame(m7_6_ch)
    describe(m7_6)
end
```

```
100%
```

```
Found initial step size
ϵ: 9.765625e-5
```

Selection deleted

Code 7.10 Fig 7.3

```
mass_seq =
-1.0087216024247314:0.02454648527479833:1.4213804397803034
```

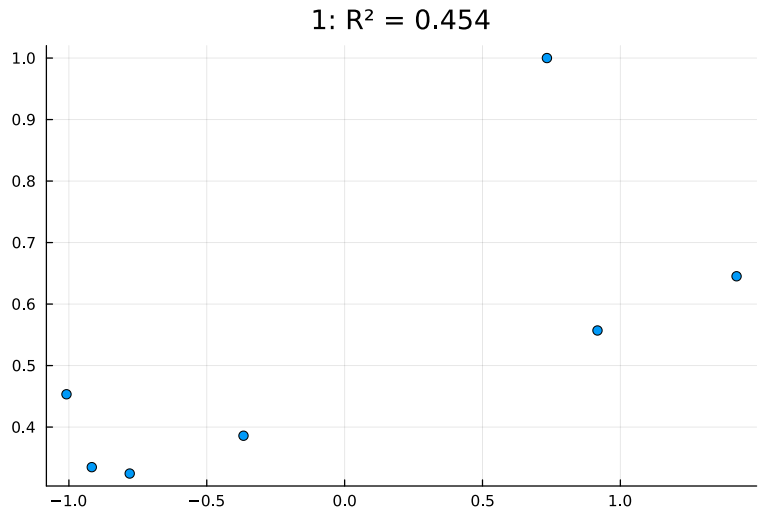
```
mass_seq = range(extrema(d.mass_std)...; length=100)
```

```
▸ [0.361326, 0.3655, 0.369673, 0.373847, 0.378021, 0.382194, 0.386
```

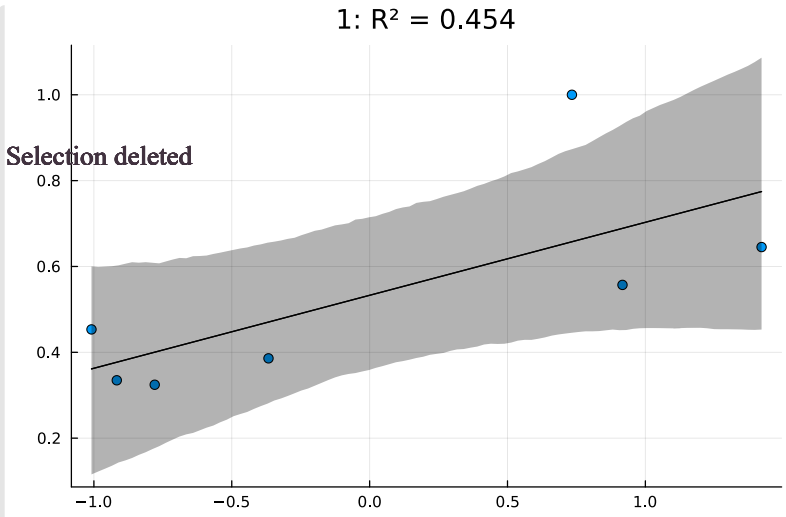
```
begin
    l = [
        @. r.a + r.b * mass_seq
        for r ∈ eachrow(m7_1)
    ]
    l = vcat(l'...)
    μ = mean.(eachcol(l))
end
```

```
100×2 Matrix{Float64}:
 0.115481  0.600084
 0.122292  0.59868
 0.128817  0.599644
 0.135562  0.600578
 0.143339  0.602665
 0.148106  0.606391
 0.153797  0.609951
 ⋮
 0.453709  1.04696
 0.453461  1.05329
 0.453229  1.0602
 0.45253   1.06744
 0.452237  1.07638
 0.452994  1.08648
```

```
begin
    ci = PI.(eachcol(l))
    ci = vcat(ci'...)
end
```



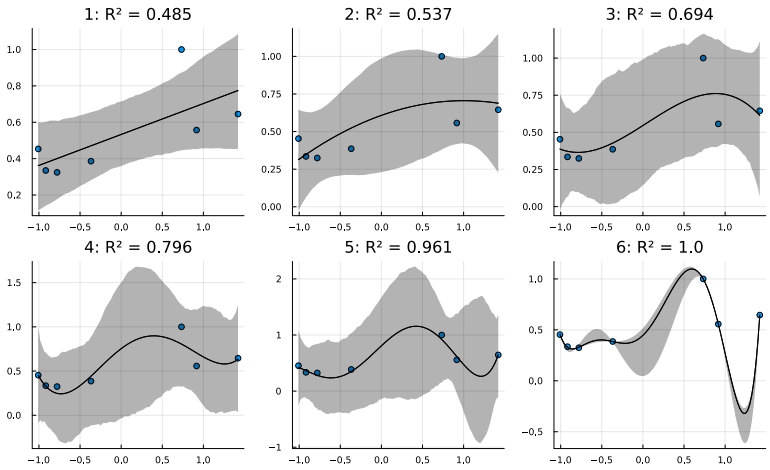

```
scatter(d.mass_std, d.brain_std; title="1: R² =
$(round(R2_is_bad(m7_1); digits=3))")
```




```
plot!(mass_seq, [μ μ]; fillrange=ci, c=:black,
fillalpha=0.3)
```

Reimplemented the brain_plot function to check my results.

```
function brain_plot(df; sigma=missing)
    degree = ncol(df[!,r"b"])
    # build mass_seq*degree matrix, with each col
exponentiated to col's index
    t = repeat(mass_seq, 1, degree)
    t = hcat(map(.^, eachcol(t), 1:degree)...)
    l = [
        r.a .+ t * collect(r[r"b"])
        for r ∈ eachrow(df)
    ]
    l = vcat(l'...)
    μ = mean.(eachcol(l))
    ci = PI.(eachcol(l))
    ci = vcat(ci'...)

    r2 = round(R2_is_bad(df, sigma=sigma); digits=3)
    scatter(d.mass_std, d.brain_std; title="$degree: R² =
$r2")
    plot!(mass_seq, [μ μ]; fillrange=ci, c=:black,
fillalpha=0.3)
end;
```




```
plot(
    brain_plot(m7_1),
    brain_plot(m7_2),
    brain_plot(m7_3),
    brain_plot(m7_4),
    brain_plot(m7_5),
    brain_plot(m7_6, sigma=0.001);
    size=(1000, 600)
)
```

Selection deleted

Code 7.11

```
i = 3;
```

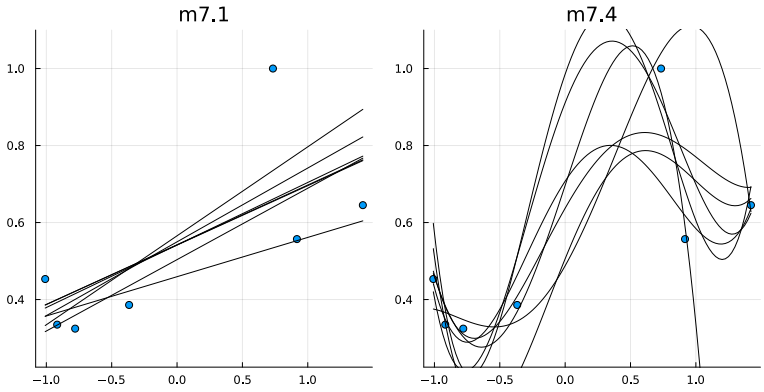
```
d_minus_i = d[setdiff(1:end,i),:];
```

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

```
function brain_loo_plot(model, data; title::String)
    (a, b) = extrema(data.brain_std)
    p = scatter(data.mass_std, data.brain_std; title=title,
ylim=(a-0.1, b+0.1))
    mass_seq = range(extrema(data.mass_std)...; length=100)

    for i ∈ 1:nrow(data)
        d_minus_i = data[setdiff(1:end,i),:]
        df = DataFrame(sample(
                model(d_minus_i.mass_std,
                d_minus_i.brain_std),
                NUTS(),
                1000))

        degree = ncol(df[!,r"b"])

        # build mass_seq*degree matrix, with each col
exponentiated to col's index

        t = repeat(mass_seq, 1, degree)
        t = hcat(map(.^, eachcol(t), 1:degree)...)
        l = [
            r.a .+ t * collect(r[r"b"])
            for r ∈ eachrow(df)
        ]
        l = vcat(l'...)
        μ = mean.(eachcol(l))
        plot!(mass_seq, μ; c=:black)
    end
    p
end
```

```
model_m7_4 = #15 (generic function with 1 method)
```

```
model_m7_4 = (mass, brain) -> model_m7_n(mass, brain,
degree=4)
```




```
plot(
    brain_loo_plot(model_m7_1, d, title="m7.1"),
    brain_loo_plot(model_m7_4, d, title="m7.4");
    size=(800, 400)
)
```

```
100%
Found initial step size
ϵ: 0.05
100%
Found initial step size
ϵ: 0.0125
100%
Found initial step size
ϵ: 0.8
100%
Found initial step size
ϵ: 0.025
100%
Found initial step size
ϵ: 1.6
100%
Found initial step size
ϵ: 0.2
100%
Found initial step size
ϵ: 0.2
100%
Found initial step size
ϵ: 0.05
100%
```

Selection deleted

7.2 Entropy and accuracy.

Code 7.12 Information and KL Divergence

```
p = [0.3, 0.7];
```

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

```
function calc_entropy_given_p(p)
    -sum(p .* log.(p))
end
```

```
0.6108643020548935
```

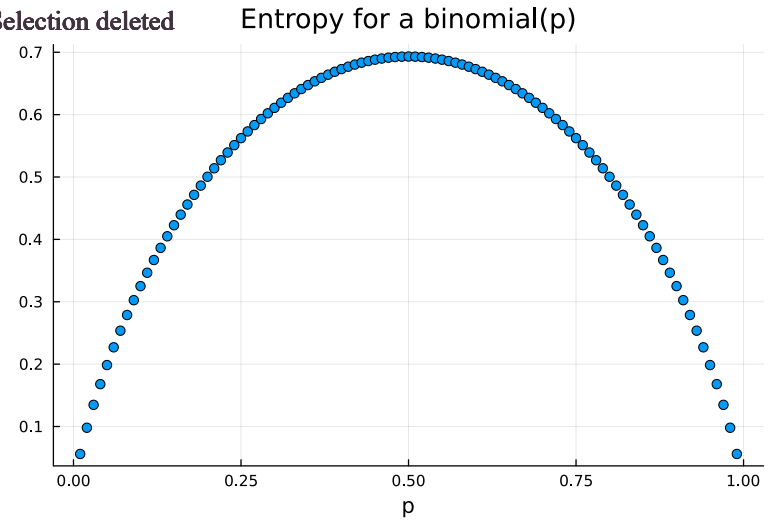
```
@time calc_entropy_given_p(p)
```

```
  0.000002 seconds (1 allocation: 80 bytes)
```

```
plot_entropy_for_binomial_varying_p (generic function with 1 meth
```

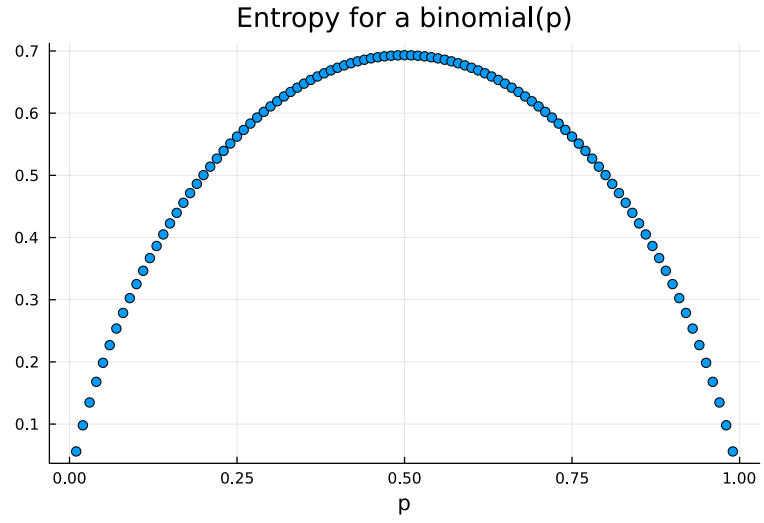
```
function plot_entropy_for_binomial_varying_p()
    let p0_vec = 0:0.01:1;
        entropy_vec = similar(p0_vec);
        for i ∈ 1:length(p0_vec)
            entropy_vec[i] =
            calc_entropy_given_p([p0_vec[i], 1-p0_vec[i]])
        end
        scatter(p0_vec, entropy_vec; title="Entropy for a
        binomial(p)", xlabel="p");
    end
end
```

Selection deleted



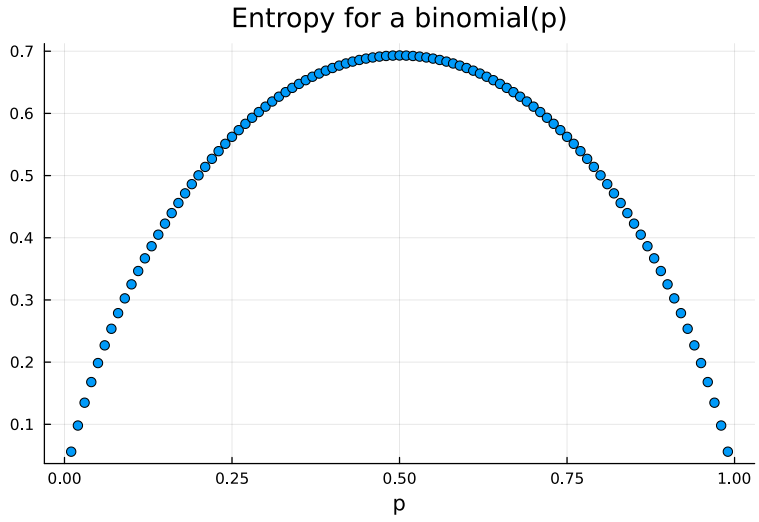

```
@time plot_entropy_for_binomial_varying_p()
```

```
  0.000806 seconds (701 allocations: 64.781 KiB)
```




```
@time plot_entropy_for_binomial_varying_p()
```

```
  0.000706 seconds (701 allocations: 64.781 KiB)
```




```
plot_entropy_for_binomial_varying_p()
```

7.13 lppd: Log-Pointwise-Predictive-Density

▸ [0.360622, 0.386067, 0.314472, 0.37288, 0.227718, 0.180141, -0.5

Selection deleted

```
lppd(m7_1, (r,x)->Normal(r.a + r.b*x, exp(r.log_σ)),
d.mass_std, d.brain_std)
```

7.14 Run lppd manually

▸ [0.360622, 0.386067, 0.314472, 0.37288, 0.227718, 0.180141, -0.5

```
[
    begin
        s = [
            logpdf(Normal(r.a + r.b * x, exp(r.log_σ)), y)
            for r ∈ eachrow(m7_1)
        ]
        # average log likelihood: log(Σᵢ expᵢ^s/n)
        # average all n samplings. i is sample i.
        logsumexp(s) - log(length(s))
    end
    for (x, y) ∈ zip(d.mass_std, d.brain_std)
]
```

7.15 lppd function

```
# it could be implemented in a generic way, but I'm too lazy
df_funcs = [
    (m7_1, (r, x) -> Normal(r.a + r.b*x, exp(r.log_σ))),
    (m7_2, (r, x) -> Normal(r.a + r."b[1]" * x +
r."b[2]"*x^2, exp(r.log_σ))),
    (m7_3, (r, x) -> Normal(r.a + r."b[1]" * x +
r."b[2]"*x^2 + r."b[3]"*x^3, exp(r.log_σ))),
    (m7_4, (r, x) -> Normal(r.a + r."b[1]" * x +
r."b[2]"*x^2 + r."b[3]"*x^3 +
                                    r."b[4]"*x^4,
exp(r.log_σ))),
    (m7_5, (r, x) -> Normal(r.a + r."b[1]" * x +
r."b[2]"*x^2 + r."b[3]"*x^3 +
                                    r."b[4]"*x^4 +
r."b[5]"*x^5, exp(r.log_σ))),
    (m7_6, (r, x) -> Normal(r.a + r."b[1]" * x +
r."b[2]"*x^2 + r."b[3]"*x^3 +
                                    r."b[4]"*x^4 +
r."b[5]"*x^5 + r."b[6]"*x^6, 0.001)),
];
```

▸ [1.31136, 0.735447, 0.855293, 0.305587, -0.299332, 38.785]

Selection deleted

```
[
    sum(lppd(df, f, d.mass_std, d.brain_std))
    for (df, f) ∈ df_funcs
]
```

7.16 Define multiple functions

```
m7_sim (generic function with 2 methods)
```

```
@model function m7_sim(x, y; degree::Int=2)
    beta ~ MvNormal(zeros(degree), 1)
    μ = x * beta
    y ~ MvNormal(μ, 1)
end
```

get_lppd

- Calculate sum(-2*lppd) from sampled params (b), an x matrix and target y values.
- call SR.lppd() function.
- Arguments:
  - m_df: data frame of model coefficients.
  - xseq: a matrix of x, each row is one sample.
- yseq: a vector of response variable y.
- Returned values
  - lppd of all samples.

```
"""
- Calculate sum(-2*lppd) from sampled params (b), an x
matrix and target y values.

$(SIGNATURES)
- call SR.lppd() function.
```

Selection deleted

```

- Arguments:
  - m_df: data frame of model coefficients.
  - xseq: a matrix of x, each row is one sample.
 - yseq: a vector of response variable y.

- Returned values
  - lppd of all samples.

"""
function get_lppd(m_df, xseq, yseq)
    t = DataFrame(:b => collect(eachrow(Matrix(m_df))))
    -2*sum(StatisticalRethinking.lppd(t, (r, x) ->
Normal(r.b'*x, 1), eachrow(xseq), yseq))
end
```

calc_train_test

```
"""
$(SIGNATURES)
"""
function calc_train_test(N, k; count=100)
    trn_v, tst_v = [], []
    for _ in 1:count
        # method sim_train_test from StatisticalRethinking
just simulates the data to be fitted by the model
        y, x_train, x_test = sim_train_test(N=N, K=k)

        estim = optimize(m7_sim(x_train, y,
degree=max(2,k)), MAP())
        m7_2 = DataFrame(sample(estim, 1000))
        # commented out is the MCMC way of estimation
instead of MAP
#         m_chain = sample(m7_sim(x_train, y,
degree=max(2,k)), NUTS(), 1000)
#         m7_2 = DataFrame(m_chain)
        t1 = get_lppd(m7_2, x_train, y)
        t2 = get_lppd(m7_2, x_test, y)
        push!(trn_v, t1)
        push!(tst_v, t2)
    end
    (mean_and_std(trn_v), mean_and_std(tst_v))
end
```

7.17 Multi-threading

TaskFailedException

```
nested task error: MethodError: no method matching
sample(::TuringOptimExt.ModeResult{NamedArrays.NamedVec
tor{Float64, Vector{Float64},
Tuple{OrderedCollections.OrderedDict{Symbol, Int64}}},
Optim.MultivariateOptimizationResults{Optim.LBFGS{Nothi
ng, LineSearches.InitialStatic{Float64},
LineSearches.HagerZhang{Float64, Base.RefValue{Bool}},
Optim.var"#19#21"}, Vector{Float64}, Float64, Float64,
Vector{Optim.OptimizationState{Float64,
Optim.LBFGS{Nothing,
LineSearches.InitialStatic{Float64},
LineSearches.HagerZhang{Float64, Base.RefValue{Bool}},
Optim.var"#19#21"}}}, Bool,
@NamedTuple{f_limit_reached::Bool,
g_limit_reached::Bool, h_limit_reached::Bool,
time_limit::Bool, callback::Bool, f_increased::Bool}},
DynamicPPL.LogDensityFunction{DynamicPPL.TypedVarInfo{@
NamedTuple{beta::DynamicPPL.Metadata{Dict{AbstractPPL.V
arName{:beta, Setfield.IdentityLens}, Int64},
Vector{Distributions.IsoNormal},
Vector{AbstractPPL.VarName{:beta,
Setfield.IdentityLens}}, Vector{Float64},
Vector{Set{DynamicPPL.Selector}}}}, Float64},
DynamicPPL.Model{typeof(Main.var"workspace#9".m7_sim),
(:x, :y), (:degree,), (), Tuple{Matrix{Float64},
Vector{Float64}}, Tuple{Int64},
DynamicPPL.DefaultContext},
Turing.Optimisation.OptimizationContext{DynamicPPL.Defa
ultContext}}}, ::Int64)
Closest candidates are:
sample(!Matched::DataFrames.DataFrame, ::Any; replace,
ordered)
@ StatisticalRethinking
~/.julia/packages/StatisticalRethinking/RYYWV/src/sampl
e_dataframe.jl:33
sample(!Matched::MCMCChains.Chains, ::Integer; replace,
ordered)
@ MCMCChains
~/.julia/packages/MCMCChains/zFCJy/src/sampling.jl:19
sample(!Matched::Random.AbstractRNG, ::Any,
!Matched::AbstractMCMC.AbstractSampler, !Matched::Any;
kwargs...)
@ AbstractMCMC
~/.julia/packages/AbstractMCMC/YrmkI/src/logdensityprob
lems.jl:43
...
Stacktrace:
[1] #calc_train_test#16
@
~/src/SR2TuringPluto.jl/notebooks/Chapter_07.jl#==#1e50
2519-f1ed-4c83-ba8d-d96e66269d4e:11 [inlined]
[2] macro expansion
@
~/src/SR2TuringPluto.jl/notebooks/Chapter_07.jl#==#823b
783e-f114-427e-b4b4-f135eec09d86:10 [inlined]
[3]
(::Main.var"workspace#9".var"#400#threadsfor_fun#18"
{Main.var"workspace#9".var"#400#threadsfor_fun#17#19"
{UnitRange{Int64}}})(tid::Int64; onethread::Bool)
@ Main.var"workspace#9"
./threadingconstructs.jl#@#==#823b783e-f114-427e-b4b4-
f135eec09d86:215
```

Selection deleted

```
[4] #400#threadsfor_fun
@ ./threadingconstructs.jl#@#==#823b783e-f114-427e-
b4b4-f135eec09d86:182 [inlined]
[5] (::Base.Threads.var"#1#2"
{Main.var"workspace#9".var"#400#threadsfor_fun#18"
{Main.var"workspace#9".var"#400#threadsfor_fun#17#19"
{UnitRange{Int64}}}, Int64})()
@ Base.Threads ./threadingconstructs.jl:154
...and 4 more exceptions.
```

Stack trace

Here is what happened, the most recent locations are first:

1. **threading_run**(fun::Main.var"workspace#9".var"#400#threadsfor_fun#18" {Main.var"workspace#9".var"#400#threadsfor_fun#17#19" {UnitRange{Int64}}}, static::Bool)
   @ *threadingconstructs.jl:172*
2. **macro expansion** @ *This cell: line 220*
3. Show more...

Selection deleted

```
begin
    k_count = 5
    k_seq = 1:k_count
    count = 100
    trn_20, tst_20 = [], []
    trn_100, tst_100 = [], []

    Threads.@threads for k in k_seq
        println("Processing $k with N=20...")
        t1, t2 = calc_train_test(20, k, count=count)
        push!(trn_20, t1)
        push!(tst_20, t2)
        println("Processing $k with N=100...")
        t1, t2 = calc_train_test(100, k, count=count)
        push!(trn_100, t1)
        push!(tst_100, t2)
    end
end
```

```
Processing 1 with N=20...
Processing 2 with N=20...
Processing 5 with N=20...
Processing 4 with N=20...
Processing 3 with N=20...
```

7.18 Plot the training and testing deviance

> **Failed to show value:**
> DivideError: integer division error

Stack trace

Here is what happened, the most recent locations are first:

1. **div** @ *int.jl:295*
2. **div** @ *div.jl:308*
3. **div** @ *div.jl:353*
4. **fld** @ *div.jl:319*
5. **mod** @ *int.jl:287*
6. **mod1** @ *operators.jl:830*
7. **mod** @ *range.jl:1508*
8. **_cycle**(v::Vector{Float64}, idx::Int64) @ *utils.jl:196*
9. **gr_draw_markers**(series::Plots.Series, x::UnitRange{Int64}, y::Vector{Float64}, z::Nothing, clims::Tuple{Float64, Float64}, msize::Int64, strokewidth::Int64) @ *gr.jl:1850*
10. **gr_draw_markers**(series::Plots.Series, x::UnitRange{Int64}, y::Vector{Float64}, z::Nothing, clims::Tuple{Float64, Float64}) @ *gr.jl:1839*
11. **gr_add_series**(sp::Plots.Subplot{Plots.GRBackend}, series::Plots.Series) @ *gr.jl:1744*
12. **gr_display**(sp::Plots.Subplot{Plots.GRBackend}, w::Measures.AbsoluteLength, h::Measures.AbsoluteLength, vp_canvas::Plots.GRViewport{Float64}) @ *gr.jl:964*
13. (::Plots.var"#509#510"{Int64, Int64, Plots.GRViewport{Float64}})(sp::Plots.Subplot{Plots.GRBackend}) @ *gr.jl:688*
14. **foreach**(f::Plots.var"#509#510"{Int64, Int64, Plots.GRViewport{Float64}}, itr::Vector{Plots.Subplot}) @ *abstractarray.jl:3097*
15. **gr_display**(plt::Plots.Plot{Plots.GRBackend}, dpi_factor::Int64) @ *gr.jl:688*
16. **#554** @ *gr.jl:2062*
17. **withenv**(::Plots.var"#554#555"{Plots.Plot{Plots.GRBackend}, Int64}, ::Pair{String, String}, ::Vararg{Pair{String, String}}) @ *env.jl:257*
18. **_show**(io::IOContext{IOBuffer}, ::MIME{Symbol("image/svg+xml")}, plt::Plots.Plot{Plots.GRBackend}) @ *gr.jl:2057*
19. **#invokelatest#2** @ *essentials.jl:892*
20. **invokelatest** @ *essentials.jl:889*
21. **show**(io::IOContext{IOBuffer}, m::MIME{Symbol("image/svg+xml")}, plt::Plots.Plot{Plots.GRBackend}) @ *output.jl:232*

Selection deleted

```
begin
    scatter(k_seq, first.(trn_20); yerr=last.(trn_20),
    label="train", title="N=20")
```

```
    scatter!(k_seq .+ .1, first.(tst_20); yerr=last.
    (tst_20), label="test")
```

Selection deleted

Failed to show value:
DivideError: integer division error

Stack trace

Here is what happened, the most recent locations are first:

1. **div** @ *int.jl:295*
2. **div** @ *div.jl:308*
3. **div** @ *div.jl:353*
4. **fld** @ *div.jl:319*
5. **mod** @ *int.jl:287*
6. **mod1** @ *operators.jl:830*
7. **mod** @ *range.jl:1508*
8. **_cycle**(v::Vector{Float64}, idx::Int64) @ *utils.jl:196*
9. **gr_draw_markers**(series::Plots.Series, x::UnitRange{Int64}, y::Vector{Float64}, z::Nothing, clims::Tuple{Float64, Float64}, msize::Int64, strokewidth::Int64) @ *gr.jl:1850*
10. **gr_draw_markers**(series::Plots.Series, x::UnitRange{Int64}, y::Vector{Float64}, z::Nothing, clims::Tuple{Float64, Float64}) @ *gr.jl:1839*
11. **gr_add_series**(sp::Plots.Subplot{Plots.GRBackend}, series::Plots.Series) @ *gr.jl:1744*
12. **gr_display**(sp::Plots.Subplot{Plots.GRBackend}, w::Measures.AbsoluteLength, h::Measures.AbsoluteLength, vp_canvas::Plots.GRViewport{Float64}) @ *gr.jl:964*
13. (::Plots.var"#509#510"{Int64, Int64, Plots.GRViewport{Float64}})(sp::Plots.Subplot{Plots.GRBackend}) @ *gr.jl:688*
14. **foreach**(f::Plots.var"#509#510"{Int64, Int64, Plots.GRViewport{Float64}}, itr::Vector{Plots.Subplot}) @ *abstractarray.jl:3097*
15. **gr_display**(plt::Plots.Plot{Plots.GRBackend}, dpi_factor::Int64) @ *gr.jl:688*
16. **#554** @ *gr.jl:2062*
17. **withenv**(::Plots.var"#554#555"{Plots.Plot{Plots.GRBackend}, Int64}, ::Pair{String, String}, ::Vararg{Pair{String, String}}) @ *env.jl:257*
18. **_show**(io::IOContext{IOBuffer}, ::MIME{Symbol("image/svg+xml")}, plt::Plots.Plot{Plots.GRBackend}) @ *gr.jl:2057*
19. **#invokelatest#2** @ *essentials.jl:892*
20. **invokelatest** @ *essentials.jl:889*
21. **show**(io::IOContext{IOBuffer}, m::MIME{Symbol("image/svg+xml")}, plt::Plots.Plot{Plots.GRBackend}) @ *output.jl:232*

Selection deleted

```
begin
    scatter(k_seq, first.(trn_100); yerr=last.(trn_100),
    label="train", title="N=100")
    scatter!(k_seq .+ .1, first.(tst_100); yerr=last.
    (tst_100), label="test")
end
```

7.3 Golem taming: regularization

No code pieces in this section

7.4 Predicting predictive accuracy

7.19 Monte Carlo of a Bayesian Linear Model

d_cars =

Selection deleted

	speed	dist
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10
7	10	18
8	10	26
9	10	34
10	11	17
⋮	more	
50	25	85

```
d_cars = DataFrame(CSV.File("data/cars.csv", drop=["Column1"]))
```

```
(50, 2)
```

```
size(d_cars)
```

	variable	mean	min	median	max	nmissing	eltype
1	:speed	15.4	4	15.0	25	0	Int64
2	:dist	42.98	2	36.0	120	0	Int64

```
describe(d_cars)
```

```
[5.28764, 25.7694]
```

```
std.(eachcol(d_cars))
```

	a	b	σ
1	-13.5798	3.81361	13.1482
2	-22.8556	4.48706	12.5707
3	-22.8556	4.48706	12.5707
4	-22.0897	3.9545	14.1456
5	-19.7891	4.11428	14.4395
6	-23.3199	4.35198	15.4593
7	-16.7861	3.86095	12.0726
8	-18.3343	4.0153	14.9624
9	-19.8632	4.1205	14.6075
10	-20.4602	4.07331	14.5578
⋮ more			
1000	-14.6604	3.71711	13.564

```
begin
    @model function model_m(speed, dist)
        a ~ Normal(0, 100)
        b ~ Normal(0, 10)
        μ = @. a + b * speed
        σ ~ Exponential(1)
        dist ~ MvNormal(μ, σ)
    end

    Random.seed!(17)
    @time m_cars_ch = sample(model_m(d_cars.speed,
    d_cars.dist), NUTS(), 1000)
    m_cars_df = DataFrame(m_cars_ch);
end
```

Selection deleted

100%

Found initial step size
ϵ: 0.003125

```
  4.633270 seconds (4.28 M allocations: 394.747 Mi
B, 3.92% gc time, 64.14% compilation time)
```

	variable	mean	min	median	max	nmissing
1	:a	-17.3265	-35.5831	-17.2642	0.299773	0
2	:b	3.91677	2.84743	3.92755	5.00309	0
3	:σ	13.7938	10.0242	13.7182	18.9374	0

```
describe(m_cars_df)
```

▸ [5.80003, 0.357458, 1.17729]

```
std.(eachcol(m_cars_df))
```

7.20 logprob

```
1000×50 Matrix{Float64}:
 -3.49553  -3.69569  -3.73555  -3.72352  …  -4.06641  -4.15059
 -3.60128  -4.15347  -3.51592  -4.02238     -3.6128   -3.66131
 -3.60128  -4.15347  -3.51592  -4.02238     -3.6128   -3.66131
 -3.73931  -4.22994  -3.57467  -4.24109     -4.48774  -4.5861
 -3.65709  -4.01515  -3.64912  -3.99351     -3.99709  -4.06206
 -3.78812  -4.18686  -3.67783  -4.11889  …  -3.90446  -3.95205
 -3.4482   -3.85121  -3.54348  -3.88427     -4.30168  -4.41574  -
  ⋮                                    ⋱
 -3.56572  -3.87583  -3.65045  -3.88701     -4.09757  -4.17747
 -3.69113  -3.74813  -3.9188   -3.78145  …  -4.22037  -4.28792
 -3.72779  -3.8742   -3.84009  -3.9114      -4.32572  -4.39368
 -3.50471  -3.72501  -3.68585  -3.78668     -4.47811  -4.58639
 -3.59089  -3.9929   -3.55534  -4.08819     -5.0411   -5.17345  -
 -3.53509  -3.78693  -3.67355  -3.83406     -4.35388  -4.45144
```

```
begin
    fun = (r, (x, y)) -> normlogpdf(r.a + r.b * x, r.σ, y)
    @time lp = StatisticalRethinking.link(m_cars_df, fun,
    zip(d_cars.speed, d_cars.dist))
    lp = hcat(lp...);
end
```

```
  0.022135 seconds (523.86 k allocations: 8.390 Mi
B)
```

Selection deleted

7.21 lppd

```
▸ [-3.6192, -3.91929, -3.66225, -3.92798, -3.57279, -3.72152, -3.5
```

```
begin
    n_samples, n_cases = size(lp)
    lppd_vals = [
        logsumexp(c) - log(n_samples)
        for c in eachcol(lp)
    ];

    ## if only lppd were needed, we can calculate it with
    # lppd_vals = lppd(m_df, (r, x) -> Normal(r.a + r.b *
    x, r.σ), d.speed, d.dist)
end
```

7.22 penalty of WAIC

```
pWAIC = Float64[
            1:  0.0181654
            2:  0.081476
            3:  0.0197703
            4:  0.0520469
            5:  0.00931132
            6:  0.0193994
            7:  0.00966674
            8:  0.0100506
            9:  0.0310315
            10:  0.0154818
            11:  0.00811483
            12:  0.0346919
            13:  0.0147627
            14:  0.00953882
            15:  0.0076483
            16:  0.0102063
            17:  0.00746253
            18:  0.00746253
            19:  0.018783
            20:  0.0142461
             ⋮  more
            41:  0.0167242
            42:  0.0100626
            43:  0.00867662
            44:  0.00947696
            45:  0.117229
            46:  0.0219985
            47:  0.088327
            48:  0.100576
            49:  1.26813
            50:  0.0173583
        ]
```

Selection deleted

```
pWAIC = [
    StatisticalRethinking.var2(c)
    for c in eachcol(lp)
]
```

7.23 WAIC

```
420.926630596472
```

```
-2*(sum(lppd_vals) - sum(pWAIC))
```

7.24 stddev of WAIC

```
16.19714181056546
```

```
begin
    waic_vec = -2 * (lppd_vals .- pWAIC)
    sqrt(n_cases * StatisticalRethinking.var2(waic_vec))
end
```

7.5 Model comparison

- Data and models from chapter 6

	a	bt	σ
1	1.32535	0.149351	1.75975
2	1.32833	0.133065	1.95495
3	1.31844	0.129862	1.88123
4	1.31068	0.198652	1.63951
5	1.31332	0.199151	1.69585
6	1.35276	0.136793	1.84516
7	1.30178	0.201123	2.01347
8	1.37402	0.098118	1.79777
9	1.36072	0.113754	1.79714
10	1.34972	0.140805	1.67981
⋮ more			
1000	1.34246	0.147348	1.90526

Selection deleted

```
begin
    begin
        Random.seed!(70)
        # number of plants
        N = 100
        h0 = rand(Normal(10, 2), N)
        treatment = repeat(0:1, inner=div(N, 2))
        fungus = [rand(Binomial(1, 0.5 - treat*0.4)) for
        treat in treatment]
        h1 = h0 .+ rand(MvNormal(5 .- 3 .* fungus, 1))

        d_fungus = DataFrame(:h0 => h0, :h1 => h1,
        :treatment => treatment, :fungus => fungus)

        @model function model_m6_6(h0, h1)
            p ~ LogNormal(0, 0.25)
            σ ~ Exponential(1)
            μ = h0 .* p
            h1 ~ MvNormal(μ, σ)
        end

        @time m6_6 = sample(model_m6_6(d_fungus.h0,
        d_fungus.h1), NUTS(), 1000)
        m6_6_df = DataFrame(m6_6)

        @model function model_m6_7(h0, treatment, fungus,
        h1)
            a ~ LogNormal(0, 0.2)
            bt ~ Normal(0, 0.5)
            bf ~ Normal(0, 0.5)
            σ ~ Exponential(1)
            p = @. a + bt*treatment + bf*fungus
            μ = h0 .* p
            h1 ~ MvNormal(μ, σ)
        end

        @time m6_7 = sample(model_m6_7(d_fungus.h0,
        d_fungus.treatment, d_fungus.fungus, d_fungus.h1),
        NUTS(), 1000)
        m6_7_df = DataFrame(m6_7)

        @model function model_m6_8(h0, treatment, h1)
            a ~ LogNormal(0, 0.2)
            bt ~ Normal(0, 0.5)
            σ ~ Exponential(1)
            p = @. a + bt*treatment
            μ = h0 .* p
            h1 ~ MvNormal(μ, σ)
```

```
        end

        @time m6_8 = sample(model_m6_8(d_fungus.h0,
        d_fungus.treatment, d_fungus.h1), NUTS(), 1000)
        m6_8_df = DataFrame(m6_8);
    end
end
```

```
100%
```

```
Found initial step size
ϵ: 0.025
```

```
100%
```

```
Found initial step size
ϵ: 0.0125
```

```
100%
```

```
Found initial step size
ϵ: 0.003125
```

```
  3.593926 seconds (3.03 M allocations: 245.770 MiB, 72.44% compilation time)
  5.777817 seconds (4.12 M allocations: 488.726 MiB, 3.72% gc time, 66.40% compilation time)
  4.867473 seconds (3.62 M allocations: 366.757 MiB, 2.19% gc time, 68.24% compilation time)
```

Selection deleted

7.25 WAIC for m6_7 (include both treatment and fungus)

```
(WAIC = 334.209, lppd = -163.395, penalty = 3.70913, std_err = 1
```

```
begin
    @time calc_normlogpdf = (r, (x,bt,bf,y)) ->
    normlogpdf(x*(r.a + r.bt*bt + r.bf*bf), r.σ, y)

    # log likelihood calculation
    @time ll = StatisticalRethinking.link(m6_7_df,
    calc_normlogpdf,
        zip(d_fungus.h0, d_fungus.treatment,
        d_fungus.fungus, d_fungus.h1));
    ll = hcat(ll...);
    @show size(ll)
    @time psis.waic(ll)
end
```

```
  0.000021 seconds
  0.054654 seconds (1.70 M allocations: 26.684 MiB)
size(ll) = (1000, 100)
  0.001115 seconds (119 allocations: 1.544 MiB)
```

7.26 Compare WAIC of m6.6 - m6.8

	models	WAIC	lppd	SE	dWAIC	dSE	pWAI
1	"m7"	334.2	326.79	13.03	0.0	0.0	3.71
2	"m8"	401.7	394.89	17.42	67.5	14.47	3.42
3	"m6"	418.7	415.09	13.23	84.5	12.27	1.82

```
begin
    calc_normlogpdf_1 = (r, (x,y)) -> normlogpdf(x*r.p, r.σ, y)
    m6_ll = StatisticalRethinking.link(m6_6_df, calc_normlogpdf_1, zip(d_fungus.h0, d_fungus.h1));
    m6_ll = hcat(m6_ll...);

    calc_normlogpdf_3 = (r, (x,bt,bf,y)) -> normlogpdf(x*(r.a + r.bt*bt + r.bf*bf), r.σ, y)
    m7_ll = StatisticalRethinking.link(m6_7_df, calc_normlogpdf_3, zip(d_fungus.h0, d_fungus.treatment, d_fungus.fungus, d_fungus.h1));
    m7_ll = hcat(m7_ll...);

    calc_normlogpdf_2 = (r, (x,bt,y)) -> normlogpdf(x*(r.a + r.bt*bt), r.σ, y)
    m8_ll = StatisticalRethinking.link(m6_8_df, calc_normlogpdf_2, zip(d_fungus.h0, d_fungus.treatment, d_fungus.h1));
    m8_ll = hcat(m8_ll...);

    compare([m6_ll, m7_ll, m8_ll], :waic, mnames=["m6", "m7", "m8"])
end
```

Selection deleted

7.27 stddev of WAIC delta between m6.7 and m6.8

```
14.397840569160383
```

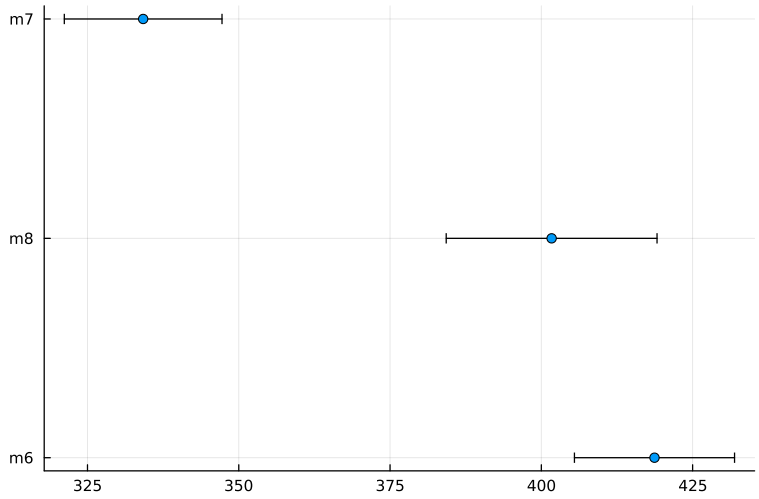
```
begin
    waic_m6_7 = waic(m7_ll, pointwise=true).WAIC
    waic_m6_8 = waic(m8_ll, pointwise=true).WAIC
    n = length(waic_m6_7)
    diff_m6_78 = waic_m6_7 - waic_m6_8
    sqrt(n*StatisticalRethinking.var2(diff_m6_78))
end
```

7.28 99% confidence interval of dWAIC

```
▸ [12.96, 67.04]
```

```
 40.0 .+ [-1, 1]*10.4*2.6
```

7.29 plot WAIC/deviance of 3 different models




```
begin
    dw = compare([m6_ll, m7_ll, m8_ll], :waic, mnames=
    ["m6", "m7", "m8"])
    scatter(reverse(dw.WAIC), reverse(dw.models);
    xerror=reverse(dw.SE))
end
```

Selection deleted

7.30 stddev of WAIC differences between m6.6 and m6.8

```
9.896685041881293
```

```
begin
    waic_m6_6 = waic(m6_ll, pointwise=true).WAIC
    waic_m6_8_2 = waic(m8_ll, pointwise=true).WAIC
    diff_m6_68 = waic_m6_6 - waic_m6_8_2
    sqrt(n*StatisticalRethinking.var2(diff_m6_68))
end
```

7.31 stddev of WAIC differences among all models

Current version of `StatisticalRethinking.compare` doesn't calculate pairwise error. You should use above logic to get values not returned in `compare` result.

7.32 Fit 3 models of the divorce dataset (Divorce rate ~ Marriage rate, Age at Marriage) and get the corresponding pointwise log-likelihood.

- The divorce datasets contain 50 data points/states.

	a	bA	bM	σ
1	-0.0558592	-0.564404	-0.227424	0.881001
2	0.0269661	-0.586021	0.115961	0.932721
3	0.0941538	-0.600504	-0.0344909	0.864356
4	-0.00379184	-0.637799	-0.128804	0.960596
5	-0.00379184	-0.637799	-0.128804	0.960596
6	0.0439258	-0.404662	-0.104283	0.672458
7	0.0439258	-0.404662	-0.104283	0.672458
8	-0.0231142	-0.478822	-0.0779828	0.795121
9	-0.113931	-0.675914	-0.106931	0.732142
10	-0.115146	-0.408913	0.0772604	0.678882
⋮ more				
1000	0.0493391	-0.774777	-0.0907425	0.834947

Selection deleted

```
begin
    Random.seed!(1)
    d_divorce =
    DataFrame(CSV.File("data/WaffleDivorce.csv"))
    d_divorce[!,:D] = standardize(ZScoreTransform,
    d_divorce.Divorce)
    d_divorce[!,:M] = standardize(ZScoreTransform,
    d_divorce.Marriage)
    d_divorce[!,:A] = standardize(ZScoreTransform,
    d_divorce.MedianAgeMarriage)
    @show size(d_divorce)
    @model function model_m5_1(A, D)
        σ ~ Exponential(1)
        a ~ Normal(0, 0.2)
        bA ~ Normal(0, 0.5)
        μ = @. a + bA * A
        D ~ MvNormal(μ, σ)
    end

    @time m5_1 = sample(model_m5_1(d_divorce.A,
    d_divorce.D), NUTS(), 1000)
    m5_1_df = DataFrame(m5_1)

    @model function model_m5_2(M, D)
        σ ~ Exponential(1)
        a ~ Normal(0, 0.2)
        bM ~ Normal(0, 0.5)
        μ = @. a + bM * M
        D ~ MvNormal(μ, σ)
    end

    @time m5_2 = sample(model_m5_2(d_divorce.M,
    d_divorce.D), NUTS(), 1000)
    m5_2_df = DataFrame(m5_2);

    @model function model_m5_3(A, M, D)
        σ ~ Exponential(1)
        a ~ Normal(0, 0.2)
        bA ~ Normal(0, 0.5)
        bM ~ Normal(0, 0.5)
        μ = @. a + bA * A + bM * M
        D ~ MvNormal(μ, σ)
    end

    @time m5_3 = sample(model_m5_3(d_divorce.A,
    d_divorce.M, d_divorce.D), NUTS(), 1000)
    m5_3_df = DataFrame(m5_3);
end
```

```
100%
```

```
Found initial step size
ϵ: 0.05
```

```
100%
```

```
Found initial step size
ϵ: 0.025
```

```
100%
```

```
Found initial step size
ϵ: 0.025
```

```
size(d_divorce) = (50, 16)
  4.255711 seconds (3.26 M allocations: 263.999 MiB,
2.36% gc time, 68.55% compilation time)
  4.245426 seconds (3.25 M allocations: 261.452 MiB,
1.43% gc time, 69.63% compilation time)
  5.417593 seconds (3.74 M allocations: 333.725 MiB,
3.13% gc time, 64.58% compilation time)
```

```
1000×50 Matrix{Float64}:
 -2.00667  -2.37024  -0.996025  -2.71984  …  -0.879512  -1.17348
 -1.77528  -1.35202  -0.970049  -1.48071     -0.852814  -1.04555
 -1.73167  -1.57019  -0.877906  -1.75229     -0.777612  -1.16394
 -1.75853  -1.80834  -1.00913   -1.95755     -0.900331  -1.16842
 -1.75853  -1.80834  -1.00913   -1.95755     -0.900331  -1.16842
 -2.58925  -2.64059  -0.786789  -3.53794  …  -0.646975  -1.24077
 -2.58925  -2.64059  -0.786789  -3.53794     -0.646975  -1.24077
  ⋮                                      ⋱
 -1.89483  -1.73666  -0.837993  -1.91391     -0.690428  -1.09316
 -2.32464  -2.15832  -0.957138  -2.51128  …  -0.74571   -0.961021
 -2.30974  -2.21973  -0.948318  -2.43778     -0.701305  -0.934621
 -1.81103  -1.94061  -0.689592  -2.05472     -0.562244  -1.30165
 -2.05383  -2.93154  -0.735413  -2.9586      -0.531055  -1.34278
 -1.66605  -1.61211  -0.857697  -1.60905     -0.738616  -1.11867
```

Selection deleted

```
begin
    @time m5_1_ll = StatisticalRethinking.link(m5_1_df,
        (r, (x,y)) -> StatsFuns.normlogpdf(r.a + r.bA * x,
        r.σ, y),
        zip(d_divorce.A, d_divorce.D));
    m5_1_ll = hcat(m5_1_ll...)
    @show size(m5_1_ll)

    @time m5_2_ll = StatisticalRethinking.link(m5_2_df,
        (r, (x,y)) -> StatsFuns.normlogpdf(r.a + r.bM * x,
        r.σ, y),
        zip(d_divorce.M, d_divorce.D));
    m5_2_ll = hcat(m5_2_ll...)
    @show size(m5_2_ll)

    @time m5_3_ll = StatisticalRethinking.link(m5_3_df,
        (r, (a,m,y)) -> StatsFuns.normlogpdf(r.a + r.bA * a
        + r.bM * m, r.σ, y),
        zip(d_divorce.A, d_divorce.M, d_divorce.D));
    m5_3_ll = hcat(m5_3_ll...);
end
```

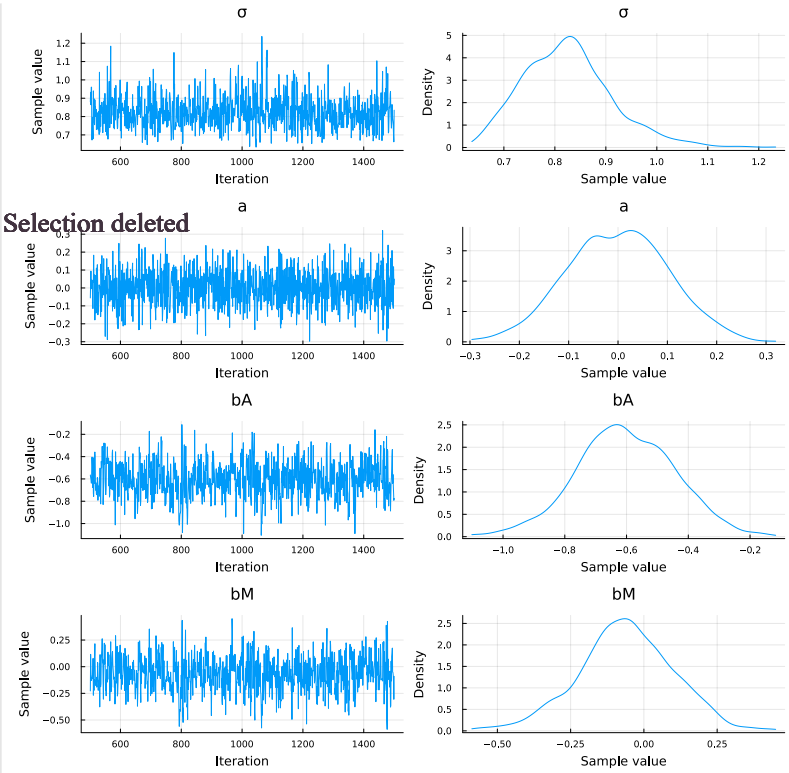
```
  0.026356 seconds (623.86 k allocations: 9.916 Mi
B)
size(m5_1_ll) = (1000, 50)
  0.021203 seconds (623.86 k allocations: 9.916 MiB)
size(m5_2_ll) = (1000, 50)
  0.025481 seconds (848.31 k allocations: 13.342 MiB)
```

	variable	mean	min	median	max	nr
1	:a	-0.000837454	-0.2971	0.00141241	0.320335	0
2	:bA	-0.603134	-1.10509	-0.607978	-0.113036	0
3	:bM	-0.0623091	-0.587797	-0.0600914	0.450345	0
4	:σ	0.825118	0.633363	0.820318	1.23687	0

```
describe(m5_3_df)
```

▸ [0.101411, 0.159148, 0.162949, 0.0882977]

```
std.(eachcol(m5_3_df))
```



Selection deleted

```
plot(m5_3)
```

7.33 Compare 3 models via PSIS: m5_1 is best.

	models	PSIS	lppd	SE	dPSIS	dSE	pPSI
1	"m5.1"	124.6	118.5	12.26	0.0	0.0	3.28
2	"m5.3"	126.7	118.19	12.28	2.1	0.77	4.6
3	"m5.2"	139.1	133.48	9.61	14.5	8.82	3.03

```
compare([m5_1_ll, m5_2_ll, m5_3_ll], :psis, mnames=["m5.1",
"m5.2", "m5.3"])
```

7.34 m5_3: Compare pointwise PSIS Pareto k and WAIC penalty

▸ [0.101411, 0.159148, 0.162949, 0.0882977]

```
begin
    @show describe(m5_3_df)
    std.(eachcol(m5_3_df))
end
```

```
describe(m5_3_df) = 4×7 DataFrame
 Row │ variable  mean          min        median      max         nmissing  eltype
     │ Symbol    Float64       Float64    Float64     Float64     Int64     DataType
─────┼──────────────────────────────────────────────────────────────────────────────
   1 │ a         -0.000837454  -0.2971     0.00141241  0.320335          0  Float64
   2 │ bA        -0.603134     -1.10509   -0.607978   -0.113036          0  Float64
   3 │ bM        -0.0623091    -0.587797  -0.0600914   0.450345          0  Float64
   4 │ σ          0.825118      0.633363   0.820318    1.23687           0  Float64
```

Selection deleted

ll_to_psis

- reshape data to format of ParetoSmooth.psis_loo function
- Arguments
  - ll: a [number-of-samplings X number-of-data-points] matrix of log-likelihoods.
- Returned values:
  - a 3D array: [number-of-data-points X number-of-samples X 1].

```
"""
- reshape data to format of ParetoSmooth.psis_loo function

$(SIGNATURES)

- Arguments
  - ll: a [number-of-samplings X number-of-data-points] matrix of log-likelihoods.

- Returned values:
  - a 3D array: [number-of-data-points X number-of-samples X 1].
"""
function ll_to_psis(ll::Matrix{<:Real})
    @show size(ll)
    t = ll'
    @show size(t)
    #Make the dataset 3 dimensional
    collect(reshape(t, size(t)..., 1))
end
```

▸ (WAIC = [4.10301, 4.01367, 1.83596, 4.51732, 1.99058, 2.86319, 2

```
begin
    m5_3_t = ll_to_psis(m5_3_ll)
    @show size(m5_3_t)
    PSIS_m5_3 = ParetoSmooth.psis_loo(m5_3_t)
    WAIC_m5_3 = psis.waic(m5_3_ll, pointwise=true)
end
```

ⓘ No source provided for samples; variables are assumed t
o be from a Markov Chain. If the samples are independen
t, specify this with keyword argument `source=:other`.

```
size(ll) = (1000, 50)
size(t) = (50, 1000)
size(m5_3_t) = (50, 1000, 1)
```

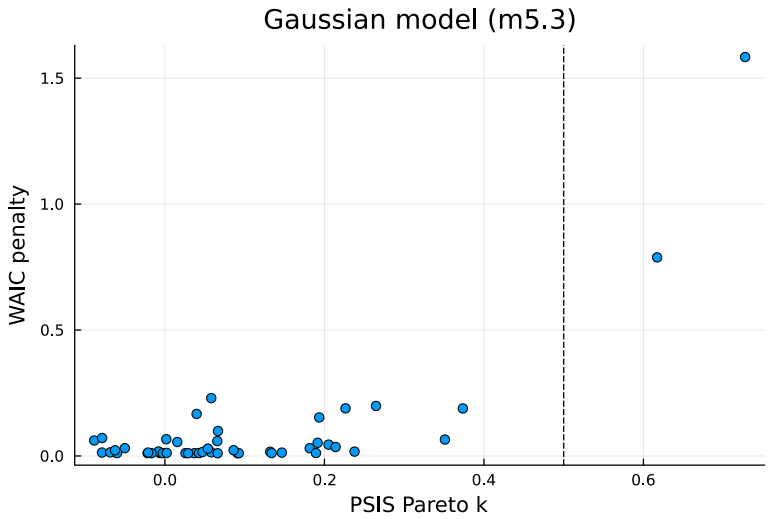
```
PSIS_m5_3
```

⚠ Some Pareto k values are high (>.7), indicating PSIS ha
s failed to approximate the true distribution.

Results of PSIS-LOO-CV with 1000 Monte Carlo sample
s and 50 data points. Total Monte Carlo SE of 0.14.

	total	se_total	mean	se_mean
cv_elpd	-63.83	6.37	-1.28	0.13
naive_lpd	-59.09	4.67	-1.18	0.09
p_eff	4.74	1.82	0.09	0.04

Selection deleted




```
begin
    scatter(PSIS_m5_3.pointwise(:pareto_k),
    WAIC_m5_3.penalty,
            xlab="PSIS Pareto k", ylab="WAIC penalty",
        title="Gaussian model (m5.3)")
    vline!([0.5], c=:black, s=:dash)
end
```

find_outliers

Find outliers with either Pareto K too large or WAIC penalty too large.

Returned arguments

- a new dataframe given selected rows

```
"""
Find outliers with either Pareto K too large or WAIC
penalty too large.
$(SIGNATURES)

Returned arguments
- a new dataframe given selected rows
"""
find_outliers(df::DataFrame, pareto_k_vec,
waic_penalty_vec::Vector{Float64}; min_pareto_k::Float64=0,
min_waic_penalty::Float64=0) = begin
    state_ind_vec = [i for i in 1:length(pareto_k_vec) if
    pareto_k_vec[i] >= min_pareto_k ||
    waic_penalty_vec[i]>=min_waic_penalty]
    #newdf = DataFrame(d_divorce[state_ind_vec,:],
    "ParetoK"=>pareto_k_vec[state_ind_vec],
    "WAIC_Penalty"=>waic_penalty_vec[state_ind_vec])

    # Combine with hcat
    new_df = DataFrames.hcat(df[state_ind_vec,:],
    pareto_k_vec[state_ind_vec],
    waic_penalty_vec[state_ind_vec]; makeunique=true)
    # Set column names (optional)
    DataFrames.rename!(new_df, append!(names(df),
    ["ParetoK", "WAIC_Penalty"]))
    new_df
end
```

Selection deleted

	Location	Loc	Population	MedianAgeMarriage	Marriage
1	"Idaho"	"ID"	1.57	23.2	25.8
2	"Maine"	"ME"	1.33	26.4	13.5

```
find_outliers(d_divorce, PSIS_m5_3.pointwise(:pareto_k),
WAIC_m5_3.penalty; min_pareto_k=0.4, min_waic_penalty=0.5)
```

7.34.1 `m5_3_log_σ_normal`: Compare pointwise PSIS Pareto k and WAIC penalty

```
model_m5_3_log_σ (generic function with 2 methods)
```

```
@model function model_m5_3_log_σ(A, M, D)
    log_σ ~ Normal(0, 0.2)
    a ~ Normal(0, 0.2)
    bA ~ Normal(0, 0.5)
    bM ~ Normal(0, 0.5)
    μ = @. a + bA * A + bM * M
    D ~ MvNormal(μ, exp(log_σ))
end
```

	a	bA	bM	log_σ
1	-0.0597823	-0.687137	-0.208746	-0.0589428
2	-0.121128	-0.660067	-0.248804	-0.0354727
3	-0.139087	-0.724421	0.053917	-0.129499
4	0.122024	-0.794938	-0.0262842	-0.194692
5	-0.111765	-0.522158	-0.149208	-0.105211
6	-0.0946616	-0.660986	-0.066957	-0.183706
7	-0.0880742	-0.604822	-0.0846655	-0.231954
8	0.131327	-1.04851	-0.346722	-0.0505851
9	-0.119918	-0.425962	0.361554	-0.294779
10	0.0601405	-0.929196	-0.264513	-0.208219
⋮ more				
1000	-0.0411175	-0.540578	-0.206741	-0.321188

```
begin
    @time m5_3_log_σ = sample(model_m5_3_log_σ(d_divorce.A,
    d_divorce.M, d_divorce.D), NUTS(), 1_000)
    m5_3_log_σ_df = DataFrame(m5_3_log_σ);
end
```

Selection deleted

100%

Found initial step size
ϵ: 0.2

```
  5.387357 seconds (3.78 M allocations: 340.967 Mi
B, 2.72% gc time, 63.30% compilation time)
```

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

```
est_summary(mcmc_df; σ_ind=4) = begin
    mcmc_df_sum = describe(mcmc_df)
    println("exp(median(log_σ estimate) ", exp.
    (mcmc_df_sum[σ_ind, :median]))
    println("σ median est is ", median(exp.(mcmc_df[!,
    :log_σ])))
    mcmc_df_sum[!, :σ] = std.(eachcol(mcmc_df))
    @show mcmc_df_sum
    "stddev of σ is $(std(exp.(mcmc_df[!,:log_σ])))"
end
```

```
"stddev of σ is 0.08817422802326008"
```

```
est_summary(m5_3_log_σ_df)
```

```
exp(median(log_σ estimate) 0.857319310631472
σ median est is 0.8573193134942809
mcmc_df_sum = 4×8 DataFrame
 Row │ variable  mean         min        median
max        nmissing  eltype    σ
     │ Symbol    Float64      Float64    Float64
Float64    Int64     DataType  Float64
─────┼──────────────────────────────────────────
   1 │ a         -0.00208058  -0.335236   0.000526995
0.375616          0  Float64   0.102158
   2 │ bA        -0.599445    -1.09072   -0.604161
0.0760033         0  Float64   0.153298
   3 │ bM        -0.0553365   -0.540503  -0.0593282
0.733242          0  Float64   0.155058
   4 │ log_σ     -0.149939    -0.419509  -0.153945
0.248442          0  Float64   0.100376
```

▸ (WAIC = [3.94292, 3.81473, 1.89714, 4.31483, 2.03468, 2.8069, 2.

```
begin
    @time m5_3_log_σ_ll = link(m5_3_log_σ_df,
        (r, (a,m,d)) -> StatsFuns.normlogpdf(r.a + r.bA * a
        + r.bM * m, exp(r.log_σ), d),
        zip(d_divorce.A, d_divorce.M, d_divorce.D));
    m5_3_log_σ_ll = hcat(m5_3_log_σ_ll...);
    @time PSIS_m5_3_log_σ =
    ParetoSmooth.psis_loo(ll_to_psis(m5_3_log_σ_ll))
    @time WAIC_m5_3_log_σ = psis.waic(m5_3_log_σ_ll,
    pointwise=true)
end
```

ⓘ No source provided for samples; variables are assumed t
o be from a Markov Chain. If the samples are independen
t, specify this with keyword argument `source=:other`.

```
  0.184650 seconds (1.08 M allocations: 26.523 MiB,
85.84% compilation time)
size(ll) = (1000, 50)
size(t) = (50, 1000)
  0.006581 seconds (1.52 k allocations: 4.180 MiB)
  0.000539 seconds (63 allocations: 790.562 KiB)
```

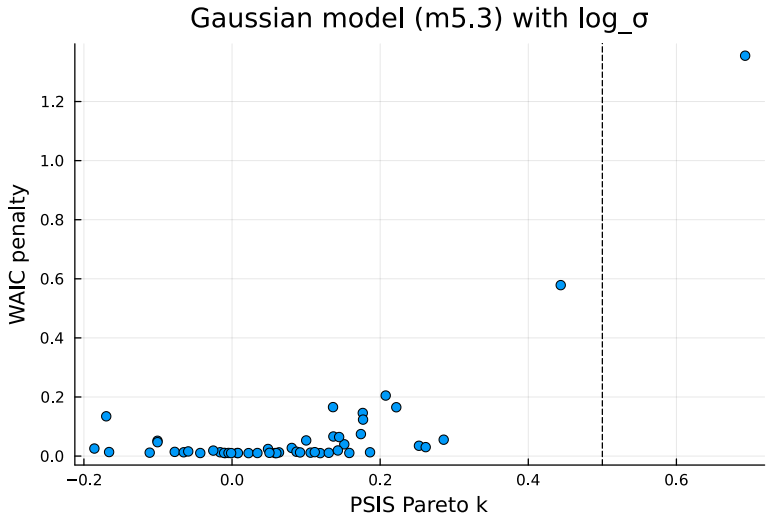
```
PSIS_m5_3_log_σ
```

Selection deleted

ⓘ Some Pareto k values are slightly high (>.5); some poin
twise estimates may be slow to converge or have high va
riance.

Results of PSIS-LOO-CV with 1000 Monte Carlo sample
s and 50 data points. Total Monte Carlo SE of 0.1.

	total	se_total	mean	se_mean
cv_elpd	-63.36	5.67	-1.27	0.11
naive_lpd	-59.50	4.30	-1.19	0.09
p_eff	3.86	1.47	0.08	0.03




```
begin
    scatter(PSIS_m5_3_log_σ.pointwise(:pareto_k),
    WAIC_m5_3_log_σ.penalty,
            xlab="PSIS Pareto k", ylab="WAIC penalty",
        title="Gaussian model (m5.3) with log_σ")
    vline!([0.5], c=:black, s=:dash)
end
```

	Location	Loc	Population	MedianAgeMarriage	Marriage
1	"Idaho"	"ID"	1.57	23.2	25.8
2	"Maine"	"ME"	1.33	26.4	13.5

```
find_outliers(d_divorce,
PSIS_m5_3_log_σ.pointwise(:pareto_k),
WAIC_m5_3_log_σ.penalty; min_pareto_k=0.4,
min_waic_penalty=0.3)
```

```
Base
```

```
@which(names)
```

7.35 m5_3t: MV T-dist df=2, Compare pointwise PSIS Pareto k and WAIC penalty.

Selection deleted

- logNormal for σ is way better than Exponential, manifested by pareto_k.

	a	bA	bM	log_σ
1	0.0431919	-0.413591	-0.0924977	0.164298
2	-0.038226	-0.461917	-0.0751188	-0.287217
3	-0.189934	-0.491807	0.271032	-0.100144
4	-0.0223252	-0.476613	0.1827	-0.166115
5	-0.101952	-0.702216	-0.284762	-0.141744
6	-0.0716474	-0.401141	0.115855	-0.116104
7	0.0323814	-0.604936	-0.125901	-0.151199
8	-0.0525956	-0.902033	-0.311722	0.146498
9	-0.0109573	-0.842162	-0.207502	0.236001
10	0.111475	-0.423923	0.024449	-0.14612
⋮ more				
1000	-0.0964225	-0.588292	-0.035434	0.0609817

Selection deleted

```
begin
    @model function model_m5_3t(A, M, D)
        #σ ~ Exponential(1)
        # logNormal is way better than Exponential,
        manifested by pareto_k.
        log_σ ~ Normal(0, 0.3)
        a ~ Normal(0, 0.2)
        bA ~ Normal(0, 0.5)
        bM ~ Normal(0, 0.5)
        μ = @. a + bA * A + bM * M
        #z = (D.-μ)/σ
        # The vector . below is optional for NUTS sample.
        But psis_loo() requires it.
        #z ~ TDist(2)
        #D ~ Distributions.IsoTDist(2, μ, σ)
        D ~ Distributions.mvtdist(2, μ, exp(log_σ))
    end

    @time m5_3t = sample(model_m5_3t(d_divorce.A,
    d_divorce.M, d_divorce.D), NUTS(), 1000)
    m5_3t_df = DataFrame(m5_3t);
end
```

100%

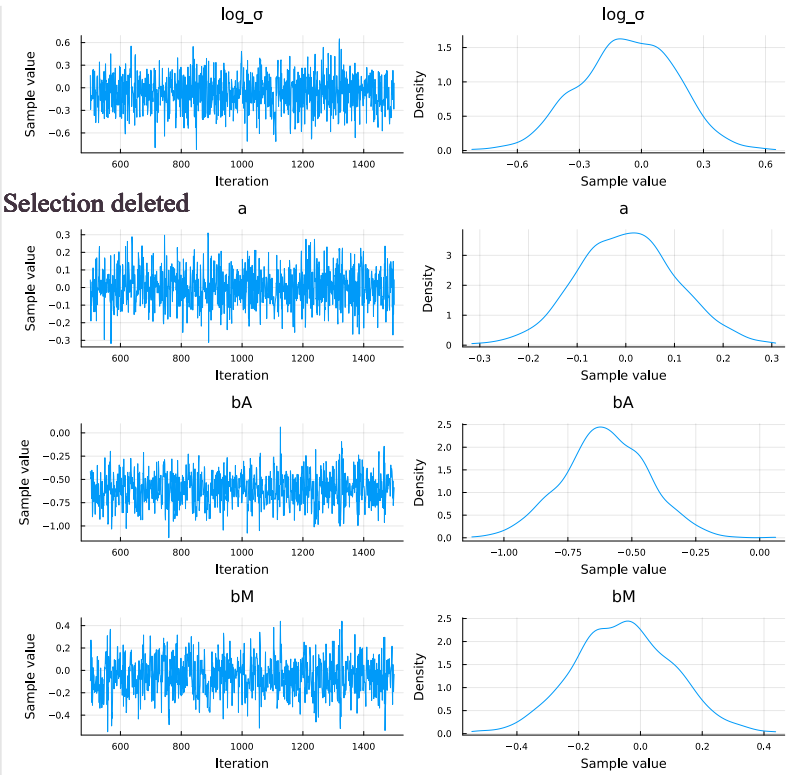
Found initial step size
ϵ: 0.4

```
  5.208111 seconds (3.79 M allocations: 335.960 Mi
B, 2.22% gc time, 67.53% compilation time)
```

"stddev of σ is 0.22022987430308238"

```
1 est_summary(m5_3t_df)
```

```
exp(median(log_σ estimate) 0.941349073343889
σ median est is 0.9413490951127395
mcmc_df_sum = 4×8 DataFrame
 Row │ variable  mean         min        median      max        nmissing  eltype
     │ Symbol    Float64      Float64    Float64     Float64    Int64     DataType
─────┼──────────────────────────────────────────────────────────────────────────────
   1 │ a          0.00266562  -0.318649   0.0021977  0.310176          0  Float64
   2 │ bA        -0.603098    -1.12604   -0.600879   0.0628147         0  Float64
   3 │ bM        -0.0551153   -0.548575  -0.0561427  0.440388          0  Float64
   4 │ log_σ     -0.0670473   -0.820551  -0.0604412  0.649521          0  Float64
```




```
1 plot(m5_3t)
```

```
1 ParetoSmooth.psis_loo(model_m5_3t(d_divorce.A, d_divorce.M,
  d_divorce.D), m5_3t)
```

```
No source provided for samples; variables are assumed t
o be from a Markov Chain. If the samples are independen
t, specify this with keyword argument `source=:other`.
```

```
⚠ Some Pareto k values are extremely high (>1). PSIS will
not produce consistent estimates.
```

```
Results of PSIS-LOO-CV with 1000 Monte Carlo sample
s and 1 data points. Total Monte Carlo SE of 0.45.
```

	total	se_total	mean	se_mean
cv_elpd	-64.19	NaN	-64.19	NaN
naive_lpd	-62.04	NaN	-62.04	NaN
p_eff	2.15	NaN	2.15	NaN

▸ (WAIC = [4.06068, 3.93203, 2.40157, 4.20458, 2.56256, 3.2509, 3.

```
begin

    m5_3t_ll = StatisticalRethinking.link(m5_3t_df,
        (r, (a,m,y)) ->
        Distributions.logpdf(Distributions.mvtdist(2, [r.a
        + r.bA * a + r.bM * m], exp(r.log_σ)), [y]),
        zip(d_divorce.A, d_divorce.M, d_divorce.D));
    m5_3t_ll = hcat(m5_3t_ll...);

    m5_3t_t = ll_to_psis(m5_3t_ll)
    @show size(m5_3t_t)
    PSIS_m5_3t = ParetoSmooth.psis_loo(m5_3t_t)
    WAIC_m5_3t = psis.waic(m5_3t_ll, pointwise=true)
end
```

ⓘ No source provided for samples; variables are assumed t
o be from a Markov Chain. If the samples are independen
t, specify this with keyword argument `source=:other`.

```
size(ll) = (1000, 50)
size(t) = (50, 1000)
size(m5_3t_t) = (50, 1000, 1)
```

Selection deleted

Results of PSIS-LOO-CV with 1000 Monte Carlo sample
s and 50 data points. Total Monte Carlo SE of 0.057.

	total	se_total	mean	se_mean
cv_elpd	-70.67	3.24	-1.41	0.06
naive_lpd	-68.12	3.11	-1.36	0.06
p_eff	2.55	0.20	0.05	0.00

```
PSIS_m5_3t.psis_object
```

```
Results of PSIS with 1000 posterior samples and 50 cases.
```

pareto_k	ess	sup_ess
0.12	746.61	331.96
0.05	799.58	307.70
0.13	935.96	568.16
0.17	987.88	377.35
0.19	876.83	439.71
0.05	786.54	413.28
0.17	989.91	342.25
0.04	778.72	383.53
0.35	630.51	135.21
0.08	650.16	329.76
0.00	823.99	476.27
0.27	627.98	202.37
0.35	899.14	149.75
0.12	874.61	479.67
0.02	901.77	560.95
0.06	938.39	487.33
0.10	935.95	489.79
0.16	927.72	426.33
-0.03	917.22	623.09
0.08	630.02	164.95
0.05	930.02	478.47
0.02	889.95	449.06
0.09	905.45	480.76
0.20	663.79	199.66
-0.04	859.18	568.05
0.10	791.92	425.28
0.09	645.79	367.49
0.26	657.33	303.14
-0.02	989.06	631.64
0.14	1120.81	348.26
0.06	950.78	503.74
0.26	874.41	260.23
0.05	938.52	486.03
0.18	757.13	252.13
0.05	955.93	501.29
0.11	1174.78	517.91
0.06	949.16	535.75

Selection deletion

		∅	
		(This table has no columns)	
1	1	:cv_elpd	-2.03076
2	2	:cv_elpd	-1.96683
3	3	:cv_elpd	-1.20084
4	4	:cv_elpd	-2.10265
5	5	:cv_elpd	-1.28145
6	6	:cv_elpd	-1.6257
7	7	:cv_elpd	-1.64614
8	8	:cv_elpd	-1.05047
9	9	:cv_elpd	-1.17854
10	10	:cv_elpd	-1.24652
⋮	more		

```
begin
    @show propertynames(PSIS_m5_3t)
    @show size(PSIS_m5_3t.pointwise)
    @show typeof(PSIS_m5_3t.pointwise)
    @show propertynames(PSIS_m5_3t.pointwise)
    PSIS_m5_3t.pointwise
end
```

Selection deleted

```
propertynames(PSIS_m5_3t) = (:estimates, :pointwise, :psis_object, :gmpd, :mcse)
size(PSIS_m5_3t.pointwise) = (50, 5)
typeof(PSIS_m5_3t.pointwise) = AxisKeys.KeyedArray{Float64, 2, NamedDims.NamedDimsArray{(:data, :statistic), Float64, 2, Matrix{Float64}}, Tuple{UnitRange{Int64}, Vector{Symbol}}}
propertynames(PSIS_m5_3t.pointwise) = (:data, :statistic)
```

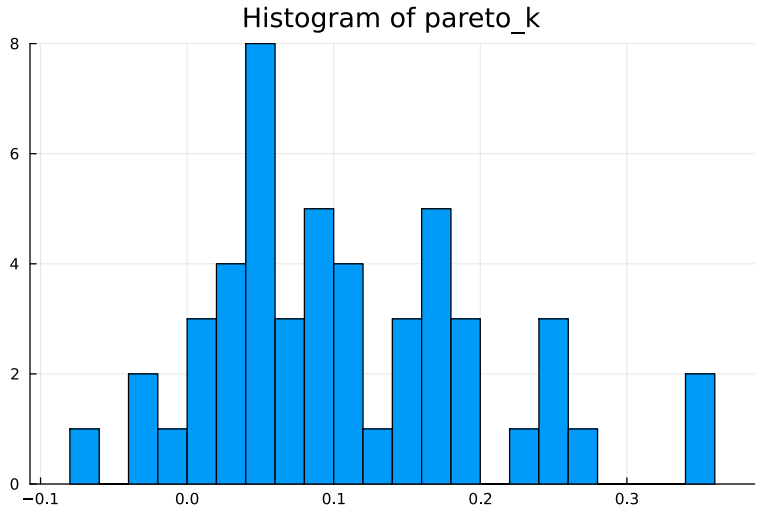
▸ [:cv_elpd, :naive_lpd, :p_eff, :mcse, :pareto_k]

```
begin
    @show PSIS_m5_3t.pointwise.data
    @show PSIS_m5_3t.pointwise.statistic
end
```

```
PSIS_m5_3t.pointwise.data = 1:50
PSIS_m5_3t.pointwise.statistic = [:cv_elpd, :naive_lpd, :p_eff, :mcse, :pareto_k]
```

		∅	
		(This table has no columns)	
1	:cv_elpd		-1.96683
2	:naive_lpd		-1.89794
3	:p_eff		0.0688969
4	:mcse		0.0100315
5	:pareto_k		0.0489091

```
#2nd data point (divorce dataset)
PSIS_m5_3t.pointwise[data=2]
```




```
histogram(collect(PSIS_m5_3t.pointwise[statistic=5]),
bins=20, title="Histogram of pareto_k")
```

ArgumentError: invalid index: ":pareto_k" of type String

Selection deleted

Stack trace

Here is what happened, the most recent locations are first:

1. **to_index**(i::String) @ *indices.jl:300*
2. **to_index**(A::AxisKeys.KeyedArray{Float64, 2, NamedDims.NamedDimsArray{(:data, :statistic), Float64, 2, Matrix{Float64}}, Tuple{UnitRange{Int64}, Vector{Symbol}}}, i::String) @ *indices.jl:277*
3. **_to_indices1**(A::AxisKeys.KeyedArray{Float64, 2, NamedDims.NamedDimsArray{(:data, :statistic), Float64, 2, Matrix{Float64}}, Tuple{UnitRange{Int64}, Vector{Symbol}}}, inds::Tuple{Base.OneTo{Int64}}, I1::String) @ *indices.jl:359*
4. **to_indices** @ *indices.jl:354*
5. **to_indices** @ *selectors.jl:212*
6. **to_indices** @ *indices.jl:344*
7. **getindex**(::AxisKeys.KeyedArray{Float64, 2, NamedDims.NamedDimsArray{(:data, :statistic), Float64, 2, Matrix{Float64}}, Tuple{UnitRange{Int64}, Vector{Symbol}}}, ::Function, ::String) @ *struct.jl:75*
8. **#getindex#26** @ *names.jl:85*
9. **getindex** @ *names.jl:82*
10. *This cell: line 2*

```
# this approach does not work.
PSIS_m5_3t.pointwise[statistic=":pareto_k"]
```

```
# this approach does not work.
PSIS_m5_3t.pointwise[statistic=":pareto_k"]
```

	∅	
	(This table has no columns)	
1	1	0.117485
2	2	0.0489091
3	3	0.128208
4	4	0.166018
5	5	0.18981
6	6	0.049939
7	7	0.166855
8	8	0.0443371
9	9	0.351656
10	10	0.0808025
⋮	more	

```
PSIS_m5_3t.pointwise(Symbol("pareto_k"))
```

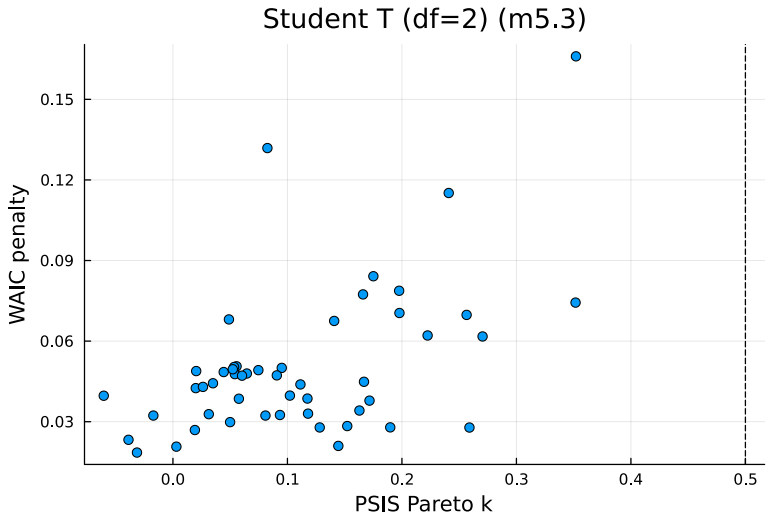
▸ (50)

Selection deleted

```
begin
    @show typeof(PSIS_m5_3t.pointwise(:cv_elpd))
    size(PSIS_m5_3t.pointwise(:cv_elpd))
end
```

```
typeof(PSIS_m5_3t.pointwise(:cv_elpd)) = AxisKeys.KeyedArray{Float64, 1, NamedDims.NamedDimsArray{(:data,), Float64, 1, SubArray{Float64, 1, Matrix{Float64}, Tuple{Base.Slice{Base.OneTo{Int64}}, Int64}, true}}, Base.RefValue{UnitRange{Int64}}}
```

- No outliers in PSIS K after replacing Normal() with student T




```
begin
        scatter(PSIS_m5_3t.pointwise(:pareto_k),
        WAIC_m5_3t.penalty,
            xlab="PSIS Pareto k", ylab="WAIC penalty",
        title="Student T (df=2) (m5.3)")
        vline!([0.5], c=:black, s=:dash)
end
```

	Location	Loc	Population	MedianAgeMarri
1	"District of Columbia"	"DC"	0.6	29.7
2	"Idaho"	"ID"	1.57	23.2
3	"Maine"	"ME"	1.33	26.4
4	"Utah"	"UT"	2.76	23.3

```
@time find_outliers(d_divorce,
PSIS_m5_3t.pointwise(:pareto_k), WAIC_m5_3t.penalty;
min_pareto_k=0.3, min_waic_penalty=0.1)
```

```
  0.000195 seconds (162 allocations: 17.859 KiB)
```

7.36 `m5_3t`: 1D T-dist (failed in ForwardDiff but succeeded via using arraydist)

Selection deleted

```
md"### 7.36 `m5_3t`: 1D T-dist (failed in ForwardDiff but succeeded via using arraydist)"
```

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

```
TDist(μ, σ, ν) = μ + Distributions.TDist(ν)*σ
```

```
MethodError: no method matching
Float64(::ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.D
ynamicPPLTag, Float64}, Float64, 4})
Closest candidates are:
(::Type{T})(::Real, !Matched::RoundingMode) where
T<:AbstractFloat
@ Base rounding.jl:207
(::Type{T})(::T) where T<:Number
@ Core boot.jl:792
Float64(!Matched::Float32)
@ Base float.jl:261
...
```

Stack trace

Here is what happened, the most recent locations are first:

1. **convert**(::Type{Float64}, x::ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 4}) @ *number.jl:7*
2. **setindex!**(A::Vector{Float64}, x::ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 4}, i1::Int64) @ *array.jl:1021*
3. **macro expansion** @ *This cell: line 10*

```
9       for i in 1:no_of_D
10          μ[i] = a + bA * A[i] + bM * M[i]
11          D[i] ~ TDist(μ[i], σ, 2::Int64)
```

4. Show more...

Selection deleted

```
begin
    @model function model_m5_3_singleT(A, M, D)
        no_of_D = length(D)
        σ ~ Exponential(1)
        a ~ Normal(0, 0.2)
        bA ~ Normal(0, 0.5)
        bM ~ Normal(0, 0.5)
        μ = similar(A)
        for i in 1:no_of_D
            μ[i] = a + bA * A[i] + bM * M[i]
            D[i] ~ TDist(μ[i], σ, 2::Int64)
        end
    end

    @time m5_3_singleT =
    sample(model_m5_3_singleT(d_divorce.A, d_divorce.M,
    d_divorce.D), NUTS(), 1_000);
    m5_3_singleT_df = DataFrame(m5_3_singleT);
end
```

100%

	a	bA	bM	log_σ
1	0.00756524	-0.596769	-0.0244071	-0.405985
2	-0.016395	-0.734817	-0.0244059	-0.389975
3	0.0942605	-0.559865	0.0817123	-0.360903
4	-0.0454193	-0.894398	0.0496404	-0.36093
5	-0.0202127	-0.667483	-0.243533	-0.488368
6	-0.143304	-0.623603	0.36428	-0.459321
7	0.0517743	-0.744883	-0.170589	-0.384995
8	0.0122438	-0.751377	0.0110436	-0.62418
9	0.0583168	-0.550769	0.262727	-0.275354
10	-0.0616891	-0.739931	-0.169161	-0.352399
⋮ more				
1000	-0.0218696	-0.894011	-0.041088	-0.525688

```
begin
    #tdist_custom(μ) = TDist(μ, σ, 2::Int64)
    #lazyarray(f, x) = LazyArray(Base.broadcasted(f, x))
    @model function model_m5_3_array_singleT(A, M, D)
        #σ ~ Exponential(1)
        log_σ ~ Normal(0, 0.3)
        a ~ Normal(0, 0.2)
        bA ~ Normal(0, 0.5)
        bM ~ Normal(0, 0.5)
        μ = @. a + bA * A + bM * M
        D ~ arraydist(map(x -> TDist(x, exp(log_σ),
    2::Int64), μ))
    end

    @time m5_3_array_singleT =
    sample(model_m5_3_array_singleT(d_divorce.A,
    d_divorce.M, d_divorce.D), NUTS(), 1_000);
    m5_3_array_singleT_df = DataFrame(m5_3_array_singleT);
end
```

Selection deleted

```
100%
```

```
Found initial step size
ϵ: 0.4
```

```
  6.361126 seconds (5.50 M allocations: 476.216 MiB, 2.85% gc time, 58.44% compilation time)
```

```
"stddev of σ is 0.08716211151314428"
```

```
est_summary(m5_3_array_singleT_df)
```

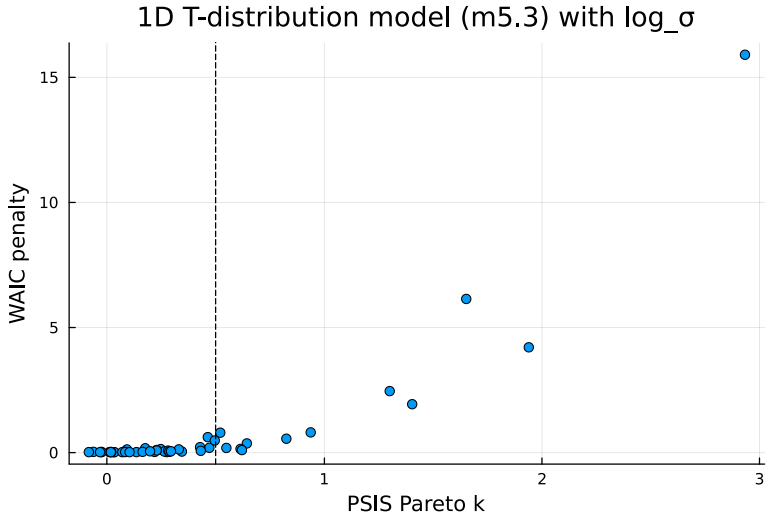
```
exp(median(log_σ estimate) 0.6382447917093729
σ median est is 0.6382447935724113
mcmc_df_sum = 4×8 DataFrame
 Row │ variable  mean        min        median      max        nmissing  eltype
     │ Symbol    Float64     Float64    Float64     Float64    Int64     DataType
─────┼──────────────────────────────────────────────────────────────────────────
   1 │ a          0.0281614  -0.347638   0.027957   0.349162          0  Float64
   2 │ bA        -0.682299   -1.20396   -0.682835  -0.210936          0  Float64
   3 │ bM         0.0479895  -0.551996   0.0432765  0.797033          0  Float64
   4 │ log_σ     -0.450273   -0.879415  -0.449033   0.0270338         0  Float64
```

▸ (WAIC = [5.16654, 4.30301, 1.48751, 4.81252, 1.7165, 2.7939, 2.7

```
begin
    @time m5_3_array_singleT_ll =
    StatisticalRethinking.link(m5_3_array_singleT_df,
        (r, (a,m,d)) -> StatsFuns.normlogpdf(r.a + r.bA * a
        + r.bM * m, exp(r.log_σ), d),
        zip(d_divorce.A, d_divorce.M, d_divorce.D));
    m5_3_array_singleT_ll = hcat(m5_3_array_singleT_ll...);
    @time PSIS_m5_3_array_singleT =
    ParetoSmooth.psis_loo(ll_to_psis(m5_3_array_singleT_ll))
    @time WAIC_m5_3_array_singleT =
    psis.waic(m5_3_array_singleT_ll, pointwise=true)
end
```

ⓘ No source provided for samples; variables are assumed t
o be from a Markov Chain. If the samples are independen
t, specify this with keyword argument `source=:other`.

```
  0.166821 seconds (1.06 M allocations: 25.030 MiB,
84.24% compilation time)
size(ll) = (1000, 50)
size(t) = (50, 1000)
  0.010295 seconds (1.52 k allocations: 4.181 MiB)
  0.000604 seconds (63 allocations: 790.562 KiB)
```




```
begin
    scatter(PSIS_m5_3_array_singleT.pointwise(:pareto_k),
    WAIC_m5_3_array_singleT.penalty,
            xlab="PSIS Pareto k", ylab="WAIC penalty",
        title="1D T-distribution model (m5.3) with log_σ")
    vline!([0.5], c=:black, s=:dash)
end
```

	Location	Loc	Population	MedianAgeMarriage	Ma
1	"Idaho"	"ID"	1.57	23.2	25
2	"Maine"	"ME"	1.33	26.4	13
3	"North Dakota"	"ND"	0.67	25.3	26
4	"Utah"	"UT"	2.76	23.3	29
5	"Wyoming"	"WY"	0.56	24.2	30

```
find_outliers(d_divorce,
PSIS_m5_3_array_singleT.pointwise(:pareto_k),
WAIC_m5_3_array_singleT.penalty; min_pareto_k=1.0,
min_waic_penalty=1.0)
```

- Many more outliers using 1D T-dist vs 2D T-dist.
- Primary cause: log_σ estimates by 1D T-dist (-.0.45) is much smaller than 2D T-dist (-0.15). But not sure why so different.

```
md"- Many more outliers using 1D T-dist vs 2D T-dist.
- Primary cause: log_σ estimates by 1D T-dist (-.0.45) is
much smaller than 2D T-dist (-0.15). But not sure why so
different."
```

Try Truncated cauchy as σ (represented as log*σ so that est*summary can be invoked).

```
md"### Try Truncated cauchy as σ (represented as log_σ so
that est_summary can be invoked)."
```

	a	bA	bM	log_σ
1	0.053781	-0.841872	-0.118816	0.474882
2	-0.0709019	-0.726279	-0.0931306	0.553218
3	-0.120763	-0.605659	-0.00644269	0.571397
4	0.0110662	-0.904736	-0.214782	0.655921
5	-0.109545	-0.819336	-0.280735	0.573337
6	0.197469	-0.583952	0.462479	0.630095
7	0.224469	-0.41519	0.410673	0.727186
8	0.230619	-0.554135	0.0743648	0.651612
9	0.00885437	-0.73151	0.2222	0.48228
10	0.0831263	-0.732038	-0.185884	0.659455
⋮ more				
1000	0.0279407	-0.566284	-0.0881914	0.60237

```
begin
    #tdist_custom(μ) = TDist(μ, σ, 2::Int64)
    #lazyarray(f, x) = LazyArray(Base.broadcasted(f, x))
    @model function model_m5_3_array_singleT_cauchy(A, M, D)
        #σ ~ Exponential(1)
        log_σ ~ truncated(Cauchy(0, 1), 0, Inf)
        a ~ Normal(0, 0.2)
        bA ~ Normal(0, 0.5)
        bM ~ Normal(0, 0.5)
        μ = @. a + bA * A + bM * M
        D ~ arraydist(map(x -> TDist(x, log_σ, 2::Int64),
    μ))
    end

    @time m5_3_array_singleT_cauchy =
    sample(model_m5_3_array_singleT_cauchy(d_divorce.A,
    d_divorce.M, d_divorce.D), NUTS(), 1_000);
    m5_3_array_singleT_cauchy_df =
    DataFrame(m5_3_array_singleT_cauchy);
end
```

```
100%
```

```
Found initial step size
ϵ: 0.8
```

```
  9.418990 seconds (8.01 M allocations: 659.759 Mi
B, 2.66% gc time, 71.78% compilation time)
```

"stddev of σ is 0.16799673935612133"

```
est_summary(m5_3_array_singleT_cauchy_df)
```

```
exp(median(log_σ estimate) 1.7785216766467562
σ median est is 1.778521743905237
mcmc_df_sum = 4×8 DataFrame
 Row │ variable  mean        min        median      max        nmissing  eltype
     │ Symbol    Float64     Float64    Float64     Float64    Int64     DataType
─────┼──────────────────────────────────────────────────────────────────────────
   1 │ a          0.0260662  -0.272912   0.0262593  0.319545          0  Float64
   2 │ bA        -0.694552   -1.23316   -0.69386   -0.0608241         0  Float64
   3 │ bM         0.0526198  -0.539655   0.0438684  0.817846          0  Float64
   4 │ log_σ      0.583859    0.400195   0.575783   1.09432           0  Float64
```

- σ estimate (0.58) is very similar to the prior σ estimate (0.64).
- Probably as many outliers as before.

```
md"- σ estimate (0.58) is very similar to the prior σ
estimate (0.64).
  - Probably as many outliers as before."
```

7.37 `m5_1t`: `m5_1` but uses T-distribution

```
md"### 7.37 `m5_1t`: `m5_1` but uses T-distribution"
```

	a	bA	log_σ
1	0.0309265	-0.433028	0.226964
2	0.173826	-0.643376	-0.0378036
3	-0.238171	-0.670755	5.86626e-5
4	-0.109542	-0.67757	-0.162775
5	0.109961	-0.534454	-0.252549
6	-0.021251	-0.668724	0.251216
7	0.0391024	-0.52464	-0.143188
8	0.0684505	-0.596609	0.0976087
9	-0.0992144	-0.34197	-0.197242
10	-0.0550326	-0.495661	0.0724789
⋮	more		
1000	0.0386741	-0.769246	0.0050955

```
begin
    @model function model_m5_1t(A, D)
        a ~ Normal(0, 0.2)
        bA ~ Normal(0, 0.5)
        log_σ ~ Normal(0, 0.2)
        μ = @. a + bA * A
        # The vector . below is optional for NUTS sample.
        But psis_loo() requires it.
        D ~ IsoTDist(2, μ, exp(log_σ))
    end

    @time m5_1t = sample(model_m5_1t(d_divorce.A,
    d_divorce.D), NUTS(), 1000)
    m5_1t_df = DataFrame(m5_1t);
end
```

```
100%
```

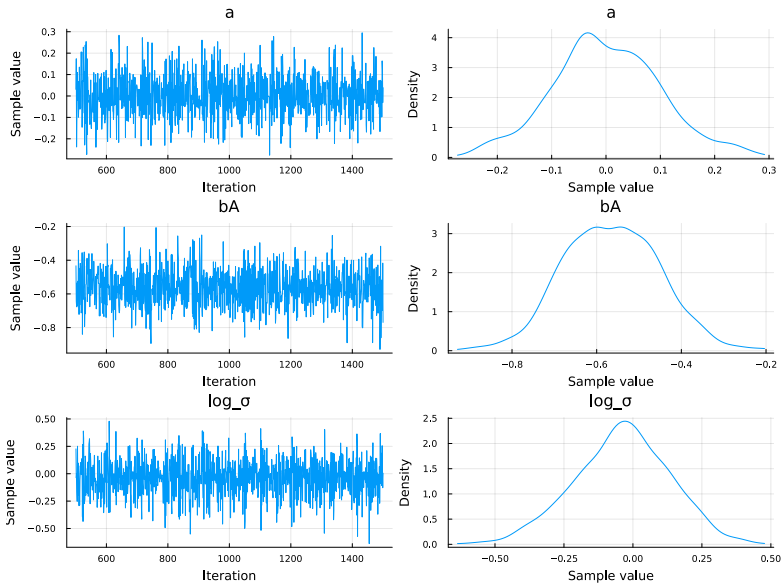
```
Found initial step size
ϵ: 0.05
```

```
  4.209546 seconds (3.30 M allocations: 265.811 Mi
B, 1.46% gc time, 70.28% compilation time)
```

```
"stddev of σ is 0.16707628322371754"
```

```
est_summary(m5_1t_df; σ_ind=3)
```

```
exp(median(log_σ estimate) 0.965735554601083
σ median est is 0.9657355548513866
mcmc_df_sum = 3×8 DataFrame
 Row │ variable  mean         min        median
max        nmissing  eltype    σ
     │ Symbol    Float64      Float64    Float64
Float64    Int64     DataType  Float64
─────┼──────────────────────────────────────────────
   1 │ a          0.00118648  -0.27642   -0.00260447
0.294911         0  Float64   0.0995105
   2 │ bA        -0.565421    -0.929385  -0.564129
-0.202886         0  Float64   0.11323
   3 │ log_σ     -0.0392006   -0.637995  -0.0348652
0.479218         0  Float64   0.172446
```



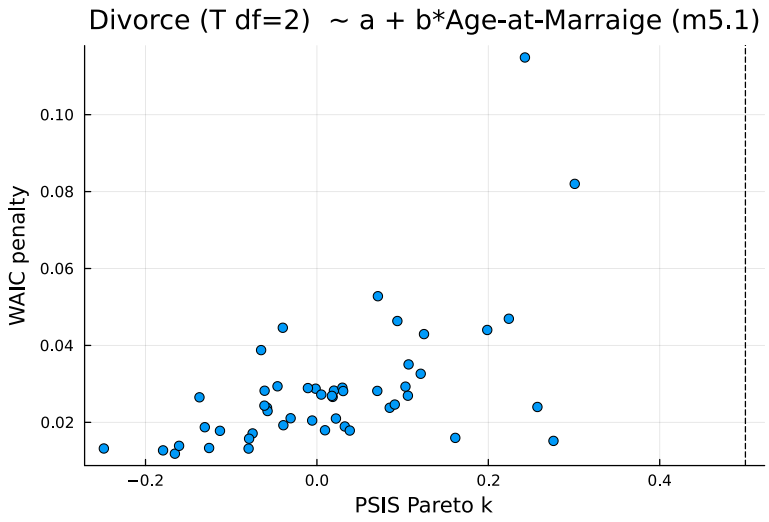

```
plot(m5_1t)
```

▸ (WAIC = [4.04416, 3.70151, 2.42175, 4.03876, 2.57078, 3.12294, 3

```
begin
    m5_1t_ll = StatisticalRethinking.link(m5_1t_df,
        (r, (a,d)) -> Distributions.logpdf(IsoTDist(2, [r.a
        + r.bA * a], exp(r.log_σ)), [d]),
        zip(d_divorce.A, d_divorce.D));
    m5_1t_ll = hcat(m5_1t_ll...);

    m5_1t_t = ll_to_psis(m5_1t_ll)
    @show size(m5_1t_t)
    PSIS_m5_1t = ParetoSmooth.psis_loo(m5_1t_t)
    WAIC_m5_1t = psis.waic(m5_1t_ll, pointwise=true)
end
```

```
No source provided for samples; variables are assumed t
o be from a Markov Chain. If the samples are independen
t, specify this with keyword argument `source=:other`.
```

```
size(ll) = (1000, 50)
size(t) = (50, 1000)
size(m5_1t_t) = (50, 1000, 1)
```




```
begin
        scatter(PSIS_m5_1t.pointwise(:pareto_k),
        WAIC_m5_1t.penalty,
            xlab="PSIS Pareto k", ylab="WAIC penalty",
        title="Divorce (T df=2)  ~ a + b*Age-at-Marraige
        (m5.1)")
        vline!([0.5], c=:black, s=:dash)
end
```

- If σ is modelled by Exponential(1), stddev of σ of `m5_1t_t` is much larger than `m5_1` (Normal distribution), which may explain why there are `more Pareto K outliers` in this student T model than Normal.
- But once log_σ ~ Normal(0, 0.2) is used instead, T distribution is better.
- A similar model by PyMC (another notebook) produces a much smaller stddev, thus no Pareto K outlier. Not sure why.

```
md"- If σ is modelled by Exponential(1), stddev of σ of
`m5_1t_t` is much larger than `m5_1` (Normal distribution),
which may explain why there are `more Pareto K outliers` in
this student T model than Normal.
- But once log_σ ~ Normal(0, 0.2) is used instead, T
distribution is better.
- A similar model by PyMC (another notebook) produces a
much smaller stddev, thus no Pareto K outlier. Not sure
why."
```

	Location	Loc	Population	MedianAgeMarriage	Ma
1	"Arkansas"	"AR"	2.92	24.3	26
2	"Idaho"	"ID"	1.57	23.2	25
3	"Minnesota"	"MN"	5.3	26.3	15
4	"North Dakota"	"ND"	0.67	25.3	26
5	"Rhode Island"	"RI"	1.05	28.2	15
6	"Utah"	"UT"	2.76	23.3	29

```
@time find_outliers(d_divorce,
PSIS_m5_1t.pointwise(:pareto_k), WAIC_m5_1t.penalty;
min_pareto_k=0.22, min_waic_penalty=0.05)
```

```
0.000205 seconds (162 allocations: 18.516 KiB)
```