Chap 8 Conditional Manatees

```
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
```

```
using Pkg, DrWatson, PlutoUI
```

```
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)
}
"""
```

Table of Contents


```
PlutoUI.TableOfContents()
```

```
begin
    using Optim
    using CSV
    using Random
    using StatsBase
    using DataFrames
    using Turing
    using StatsPlots
    using StatsFuns
    using LaTeXStrings
    using StatisticalRethinking
    using StatisticalRethinking: link
    using StatisticalRethinkingPlots
    using ParetoSmooth
    using ParetoSmoothedImportanceSampling
    using Logging
end
```

```
begin
    Plots.default(labels=false)
    #Logging.disable_logging(Logging.Warn);
end
```

8.1 Building an interaction.

Code 8.1

```
begin
    rugged = CSV.read(sr_datadir("rugged.csv"), DataFrame)
    dd = rugged[completecases(rugged, :rgdppc_2000),:]
    dd[:,:log_gdp] = log.(dd.rgdppc_2000);
    dd[:,:log_gdp_std] = dd.log_gdp / mean(dd.log_gdp)
    dd[:,:rugged_std] = dd.rugged / maximum(dd.rugged)
end;
```

Code 8.2

```
model_m8_1 (generic function with 2 methods)
```

```
begin
    r̄ = mean(dd.rugged_std)

    @model function model_m8_1(rugged_std, log_gdp_std)
        σ ~ Exponential()
        a ~ Normal(1, 1)
        b ~ Normal(0, 1)
        μ = @. a + b * (rugged_std - r̄)
        log_gdp_std ~ MvNormal(μ, σ)
    end
end
```

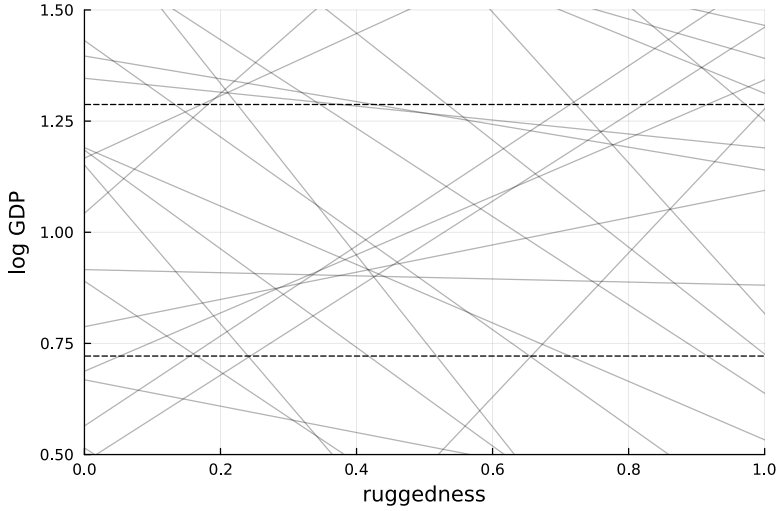
Code 8.3

	a	b	σ
1	0.853493	0.30661	0.0247257
2	0.781981	1.01245	0.389051
3	0.908497	-0.0350713	2.79544
4	2.23783	-0.368784	1.14774
5	-0.210674	0.458104	0.717742
6	2.26552	-1.29227	2.50645
7	0.693337	0.97804	1.77601
8	0.185418	-0.808299	1.14948
9	1.73788	-0.441655	2.38364
10	0.828259	0.655375	0.622402
	more		
1000	0.382903	-0.304555	0.117581

```
begin
    @time m8_1_p = sample(model_m8_1(dd.rugged_std,
    dd.log_gdp_std), Prior(), 1000)
    m8_1_p_df = DataFrame(m8_1_p);
end
```

```
100%
```

```
  6.129117 seconds (7.71 M allocations: 520.859 Mi
B, 4.08% gc time, 95.65% compilation time)
```




```
begin
    rugged_seq = range(-0.1, 1.1; length=30)
    μ = link(m8_1_p_df, (r, x) -> r.a + r.b*(x - r̄),
    rugged_seq)
    μ = hcat(μ...)

    p = plot(
        xlim=(0, 1),
        ylim=(0.5, 1.5),
        #title=L"a \sim \mathcal{N}(1,1), b \sim \mathcal{N}
    (0, 1)",
        xlab="ruggedness", ylab="log GDP",
    )
    hline!(collect(extrema(dd.log_gdp_std)); c=:black,
    s=:dash)
    for μ₀ ∈ first(eachrow(μ), 50)
        plot!(rugged_seq, μ₀; c=:black, alpha=0.3)
    end
    p
end
```

Code 8.4

```
0.551
```

```
mean(abs.(m8_1_p_df.b) .> 0.6)
```

Code 8.5

```
r̄_std = 0.21496006980670376
```

```
r̄_std = mean(dd.rugged_std)
```

```
model_m8_1a (generic function with 2 methods)
```

```
@model function model_m8_1a(rugged_std, log_gdp_std)
    σ ~ Exponential()
    a ~ Normal(1, 0.1)
    b ~ Normal(0, 0.3)
    μ = @. a + b * (rugged_std - r̄_std)
    log_gdp_std ~ MvNormal(μ, σ)
end
```

	a	b	σ
1	1.01243	-0.054505	0.135482
2	0.984841	0.0586252	0.139013
3	0.990034	0.0624086	0.138432
4	1.01159	-0.0615009	0.137907
5	0.972939	0.0212466	0.136916
6	1.02294	-0.0257198	0.139077
7	0.988567	0.00905515	0.145618
8	1.01332	-0.0122087	0.128834
9	1.00089	0.0170677	0.142197
10	1.00089	0.0170677	0.142197
	more		
1000	1.0071	-0.089297	0.13925

```
begin
    m8_1 = sample(model_m8_1a(dd.rugged_std,
    dd.log_gdp_std), NUTS(), 1000)
    m8_1_df = DataFrame(m8_1)
end
```

```
100%
```

```
Found initial step size
ϵ: 0.2
```

Code 8.6

```
md"### Code 8.6"
```

	variable	mean	min	median	max	n
1	:a	0.999822	0.966789	0.999462	1.03423	0
2	:b	-0.000280749	-0.185977	-0.000431617	0.156916	0
3	:σ	0.138222	0.118813	0.137937	0.164599	0

```
describe(m8_1_df)
```

Code 8.7

```
dd[:,:cid] = @. ifelse(dd.cont_africa == 1, 1, 2);
```

Code 8.8

```
md"### Code 8.8"
```

```
model_m8_2 (generic function with 2 methods)
```

```
@model function model_m8_2(rugged_std, cid,  log_gdp_std)
    σ ~ Exponential()
    a ~ MvNormal([1, 1], 0.1)
    b ~ Normal(0, 0.3)
    μ = @. a[cid] + b * (rugged_std - r̄)
    log_gdp_std ~ MvNormal(μ, σ)
end
```

	a[1]	a[2]	b	σ
1	0.869148	1.03811	-0.129278	0.113739
2	0.890688	1.03815	-0.161425	0.117538
3	0.903799	1.06823	0.0257777	0.108962
4	0.905462	1.05392	0.0160985	0.11585
5	0.855849	1.04963	-0.103246	0.111009
6	0.882284	1.03527	-0.024969	0.115243
7	0.877314	1.01834	-0.0259086	0.112089
8	0.917284	1.04736	-0.0231311	0.123717
9	0.84519	1.0554	-0.0550735	0.105396
10	0.893011	1.05987	-0.0922435	0.127024
more				
1000	0.878814	1.04341	-0.141336	0.113012

```
begin
    m8_2 = sample(model_m8_2(dd.rugged_std, dd.cid,
    dd.log_gdp_std), NUTS(), 1000)
    m8_2_df = DataFrame(m8_2);
end# -
```

```
100%
```

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

Code 8.9

```
1000×170 Matrix{Float64}:
 0.569479  1.05663   0.476201   0.873102  …  0.962425  -0.469477
 0.792205  1.0131    0.151764   0.650483     0.865191   0.0571353
 0.76781   1.00239   0.195007   0.682297     0.891345   0.0041531
 0.571736  1.04437   0.477895   0.861599     0.944162  -0.433966
 0.845656  1.06316   0.0524827  0.594386     0.802938   0.114346
 0.508774  0.990434  0.539962   0.891532  …  0.980638  -0.501597
 0.731975  0.990808  0.256293   0.686515     0.836768   0.0029978
 ⋮                                        ⋱
 0.67187   1.07911   0.333772   0.815441     0.966436  -0.333991
 0.700786  0.99153   0.30042    0.76026   …  0.943435  -0.151039
 0.684618  1.09645   0.323028   0.803841     0.936383  -0.3161
 0.640791  1.05046   0.356403   0.848462     1.02185   -0.418221
 0.664393  1.03402   0.360415   0.791269     0.921369  -0.23918
 0.587945  1.04871   0.464239   0.846697     0.913531  -0.403832
```

```
let
    # Compute log likelihoods for both models
    fun = (r, (x,y)) -> normlogpdf(r.a + r.b * (x - x̄),
    r.σ, y)
    global m8_1_ll = link(m8_1_df, fun, zip(dd.rugged_std,
    dd.log_gdp_std))
    m8_1_ll = hcat(m8_1_ll...)
end
```

	models	WAIC	lppd	SE	dWAIC	dSE	pWAI
1	"m8.2"	-251.9	-260.53	14.82	0.0	0.0	4.29
2	"m8.1"	-189.2	-194.03	12.94	62.7	14.75	2.43

```
let
    # need DF with a as a vector of both a[1] and a[2]
    global df = DataFrame(m8_2_df)
    df[!,:a] = collect.(zip(m8_2_df.:"a[1]",
    m8_2_df.:"a[2]"))

    fun = (r, (x,c,y)) -> normlogpdf(r.a[c] + r.b * (x -
    r̄), r.σ, y)
    global m8_2_ll = link(df, fun, zip(dd.rugged_std,
    dd.cid, dd.log_gdp_std))
    m8_2_ll = hcat(m8_2_ll...);

    compare([m8_1_ll, m8_2_ll], :waic, mnames=["m8.1",
    "m8.2"])
end
```

Code 8.10

	variable	mean	min	median	max
1	Symbol("a[1]")	0.879936	0.830731	0.879657	0.946312
2	Symbol("a[2]")	1.04927	1.01434	1.04916	1.09353
3	:b	-0.047577	-0.21004	-0.0473458	0.110664
4	:σ	0.114434	0.0979853	0.114	0.141119

```
describe(m8_2_df)
```

Code 8.11

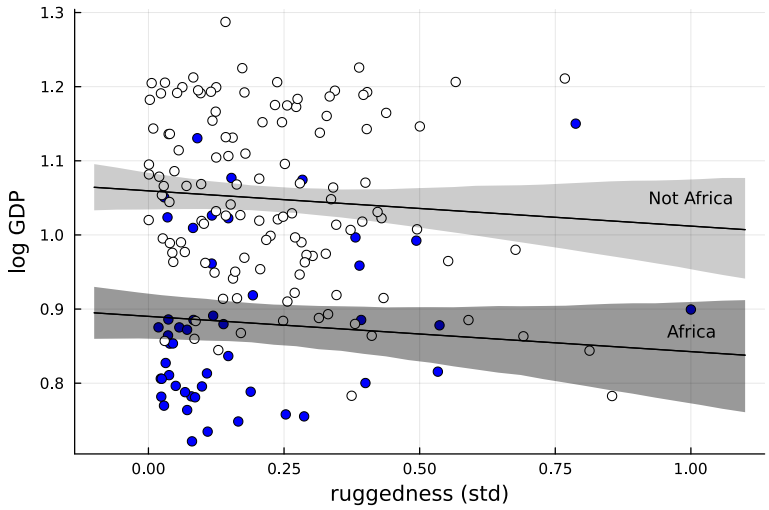
```
[-0.200335, -0.137371]
```

```
PI(map(r -> r[1] - r[2], df.a))
```

Code 8.12

```
30×2 Matrix{Float64}:
 1.03318   1.09556
 1.03409   1.0909
 1.03421   1.08575
 1.0346    1.08119
 1.03489   1.07711
 1.03494   1.07301
 1.0341    1.07022
 ⋮
 0.966847  1.07313
 0.961841  1.07337
 0.95702   1.07415
 0.951363  1.07529
 0.945999  1.07685
 0.941104  1.07688
```

```
begin
    rugged_seq_2 = range(-0.1, 1.1, length=30)
    africa     = link(df, (r, x) -> r.a[1] + r.b*(x-r̄),
    rugged_seq_2)
    africa     = hcat(africa...)'
    not_africa = link(df, (r, x) -> r.a[2] + r.b*(x-r̄),
    rugged_seq_2)
    not_africa = hcat(not_africa...)'

    μₐ = mean.(eachrow(africa))
    μₙ = mean.(eachrow(not_africa))
    PIₐ = PI.(eachrow(africa))
    PIₐ = vcat(PIₐ'...)
    PIₙ = PI.(eachrow(not_africa))
    PIₙ = vcat(PIₙ'...);
end
```




```
let
    p = plot(xlab="ruggedness (std)", ylab="log GDP")
    scatter!(dd.rugged_std[dd.cid.==1],
    dd.log_gdp_std[dd.cid.==1], c=:blue)
    scatter!(dd.rugged_std[dd.cid.==2],
    dd.log_gdp_std[dd.cid.==2], c=:white)

    plot!(rugged_seq, [μₐ, μₐ], c=:black, fillrange=PIₐ,
    fillalpha=0.4)
    plot!(rugged_seq, [μₙ, μₙ], c=:black, fillrange=PIₙ,
    fillalpha=0.2)
    annotate!([
        (1, 0.87, ("Africa", 9)),
        (1, 1.05, ("Not Africa", 9))
    ])
end
```

Code 8.13

```
md"### Code 8.13"
```

```
model_m8_3 (generic function with 2 methods)
```

```
@model function model_m8_3(rugged_std, cid,  log_gdp_std)
    σ ~ Exponential()
    a ~ MvNormal([1, 1], 0.1)
    b ~ MvNormal([0, 0], 0.3)
    μ = @. a[cid] + b[cid] * (rugged_std - r̄)
    log_gdp_std ~ MvNormal(μ, σ)
end
```

	a[1]	a[2]	b[1]	b[2]	σ
1	0.87616	1.06906	0.0980719	-0.199136	0.109488
2	0.883134	1.06642	0.0872888	-0.199434	0.108964
3	0.889054	1.03403	0.154247	-0.0813968	0.107458
4	0.899512	1.05664	0.142775	-0.089059	0.105636
5	0.880975	1.06122	0.0398175	-0.0580968	0.101088
6	0.891443	1.05275	0.187197	-0.148464	0.107889
7	0.880414	1.04969	0.0938872	-0.156629	0.111196
8	0.896248	1.04282	-0.023982	-0.0818723	0.107911
9	0.8547	1.04483	0.148851	-0.22548	0.118567
10	0.921099	1.06052	0.0998769	-0.0935121	0.133213
more					
1000	0.867388	1.04699	0.20179	-0.194131	0.118014

```
begin
    m8_3 = sample(model_m8_3(dd.rugged_std, dd.cid,
    dd.log_gdp_std), NUTS(), 1000)
    m8_3_df = DataFrame(m8_3);
end
```

```
100%
```

```
Found initial step size
ϵ: 0.2
```

Code 8.14

```
md"### Code 8.14"
```

	variable	mean	min	median	max
1	Symbol("a[1]")	0.886383	0.832409	0.88631	0.937909
2	Symbol("a[2]")	1.05054	1.01631	1.05062	1.08969
3	Symbol("b[1]")	0.130497	-0.147801	0.132145	0.343225
4	Symbol("b[2]")	-0.141965	-0.304598	-0.141985	0.0235636
5	:σ	0.111411	0.0934929	0.111052	0.133213

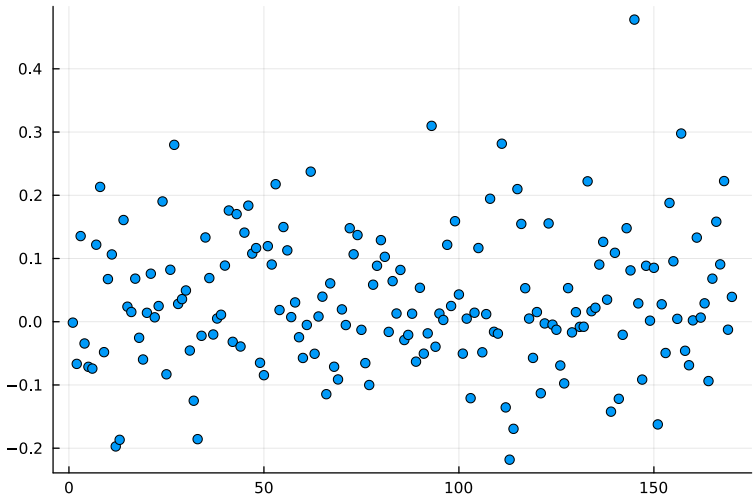
```
describe(m8_3_df)
```

Code 8.15

	models	PSIS	lppd	SE	dPSIS	dSE	pPSI
1	"m8.3"	-259.8	-269.25	14.62	0.0	0.0	5.06
2	"m8.2"	-252.5	-260.53	14.73	7.3	6.38	4.29
3	"m8.1"	-189.5	-194.03	12.92	70.3	14.92	2.43

```
let
    global df3 = DataFrame(m8_3_df)
    df3[!,:a] = collect.(zip(m8_3_df.:"a[1]",
    m8_3_df.:"a[2]"))
    df3[!,:b] = collect.(zip(m8_3_df.:"b[1]",
    m8_3_df.:"b[2]"))

    fun = (r, (x,c,y)) -> normlogpdf(r.a[c] + r.b[c] * (x -
    r̄), r.σ, y)
    global m8_3_ll = link(df3, fun, zip(dd.rugged_std,
    dd.cid, dd.log_gdp_std))
    m8_3_ll = hcat(m8_3_ll...);

    compare([m8_1_ll, m8_2_ll, m8_3_ll], :psis, mnames=
    ["m8.1", "m8.2", "m8.3"])
end
```

Code 8.16

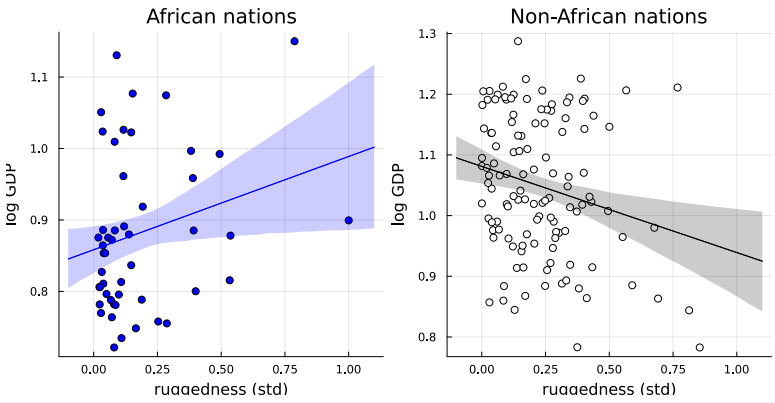



```
let
    t = m8_3_ll'
    m8_3_t = collect(reshape(t, size(t)..., 1))
    PSIS_m8_3 = psis_loo(m8_3_t)
    scatter(PSIS_m8_3.pointwise(:pareto_k))
end
```

```
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`.
```

Code 8.17

```
md"### Code 8.17"
```



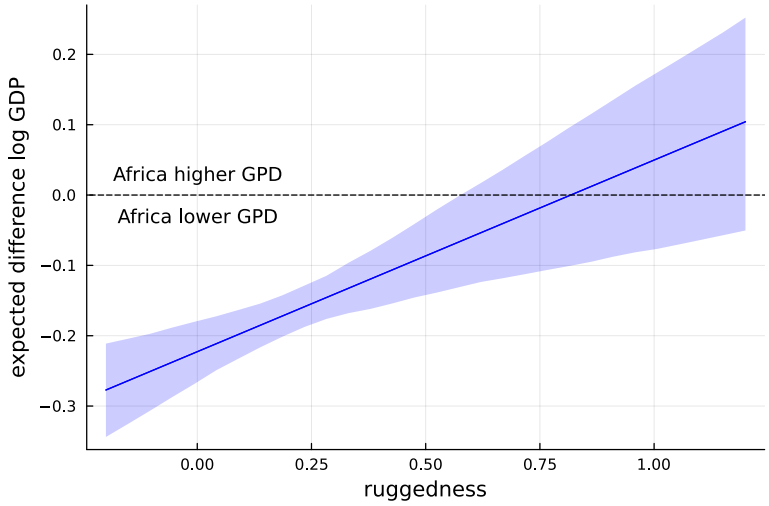

```
let
    # build data
    africa     = link(df3, (r, x) -> r.a[1] + r.b[1]*(x-r̄), rugged_seq)
    africa     = hcat(africa...)'
    not_africa = link(df3, (r, x) -> r.a[2] + r.b[2]*(x-r̄), rugged_seq)
    not_africa = hcat(not_africa...)'

    μₐ = mean.(eachrow(africa))
    μₙ = mean.(eachrow(not_africa))
    PIₐ = PI.(eachrow(africa))
    PIₐ = vcat(PIₐ'...)
    PIₙ = PI.(eachrow(not_africa))
    PIₙ = vcat(PIₙ'...);

    # plot Africa, cid=1
    p1 = plot(xlab="ruggedness (std)", ylab="log GDP",
    title="African nations")
    scatter!(dd.rugged_std[dd.cid.==1],
    dd.log_gdp_std[dd.cid.==1], c=:blue)
    plot!(rugged_seq, [μₐ, μₐ], c=:blue, fillrange=PIₐ,
    fillalpha=0.2)

    # plot non Africa, cid=2
    p2 = plot(xlab="ruggedness (std)", ylab="log GDP",
    title="Non-African nations")
    scatter!(dd.rugged_std[dd.cid.==2],
    dd.log_gdp_std[dd.cid.==2], c=:white)
    plot!(rugged_seq, [μₙ, μₙ], c=:black, fillrange=PIₙ,
    fillalpha=0.2)

    plot(p1, p2, size=(800, 400))
end
```

8.2 Symmetry of interations

```
md"## 8.2 Symmetry of interations"
```

Code 8.18

```
md"### Code 8.18"
```




```
let
    rugged_seq = range(-0.2, 1.2, length=30)
    μA = link(df3, (r, x) -> r.a[1] + r.b[1]*(x-r̄),
    rugged_seq)
    μA = vcat(μA'...)
    μN = link(df3, (r, x) -> r.a[2] + r.b[2]*(x-r̄),
    rugged_seq)
    μN = vcat(μN'...)
    delta = μA .- μN;

    # +
    μ = mean.(eachrow(delta))
    PI_v = PI.(eachrow(delta))
    PI_v = vcat(PI_v'...)

    plot(xlab="ruggedness", ylab="expected difference log
    GDP",)
    plot!(rugged_seq, [μ, μ], c=:blue, fillrange=PI_v,
    fillalpha=0.2)
    hline!([0.0], s=:dash, c=:black)
    annotate!([
        (0.0, 0.03, ("Africa higher GPD", 10)),
        (0.0, -0.03, ("Africa lower GPD", 10)),
    ])
end
```

8.3 Continuous interaction

Code 8.19

	variable	mean	min	median	max	nmissing	eltype
1	:bed	nothing	"a"	nothing	"c"	0	String1
2	:water	2.0	1	2.0	3	0	Int64
3	:shade	2.0	1	2.0	3	0	Int64
4	:blooms	128.994	0.0	111.04	361.66	0	Float64

```
begin
    tulips = CSV.read(sr_datadir("tulips.csv"), DataFrame)
    describe(tulips)
end
```

Code 8.20

```
begin
    tulips.blooms_std = tulips.blooms /
    maximum(tulips.blooms)
    tulips.water_cent = tulips.water .- mean(tulips.water)
    tulips.shade_cent = tulips.shade .- mean(tulips.shade);
end;
```

Code 8.21

0.6242

```
let
    Random.seed!(1)
    a = rand(Normal(0.5, 1), 10^4)
    sum(@. (a < 0) | (a > 1))/length(a)
end
```

Code 8.22

0.0496

```
let
    Random.seed!(1)
    a = rand(Normal(0.5, 0.25), 10^4)
    sum(@. (a < 0) | (a > 1))/length(a)
end
```

Code 8.23

m8_4 (generic function with 2 methods)

```
# +
@model function m8_4(water_cent, shade_cent, blooms_std)
    a ~ Normal(0.5, 0.25)
    bw ~ Normal(0, 0.25)
    bs ~ Normal(0, 0.25)
    μ = @. a + bw*water_cent + bs*shade_cent
    σ ~ Exponential(1)
    blooms_std ~ MvNormal(μ, σ)
end
```

	variable	mean	min	median	max	nmissi
1	:a	0.360393	0.240109	0.360141	0.502665	0
2	:bs	-0.113858	-0.26726	-0.112644	0.0528656	0
3	:bw	0.204355	0.0368693	0.203398	0.366633	0
4	:σ	0.177915	0.115244	0.175107	0.3326	0

```
begin
    m8_4_c = sample(m8_4(tulips.water_cent,
    tulips.shade_cent, tulips.blooms_std),
        NUTS(), 1000)
    m8_4_df = DataFrame(m8_4_c)
    describe(m8_4_df)
end
```

100%

Found initial step size
ϵ: 0.2

Code 8.24

```
m8_5 (generic function with 2 methods)
```

```
@model function m8_5(water_cent, shade_cent, blooms_std)
    a ~ Normal(0.5, 0.25)
    bw ~ Normal(0, 0.25)
    bs ~ Normal(0, 0.25)
    bws ~ Normal(0, 0.25)
    μ = @. a + bw*water_cent + bs*shade_cent +
bws*water_cent*shade_cent
    σ ~ Exponential(1)
    blooms_std ~ MvNormal(μ, σ)
end
```

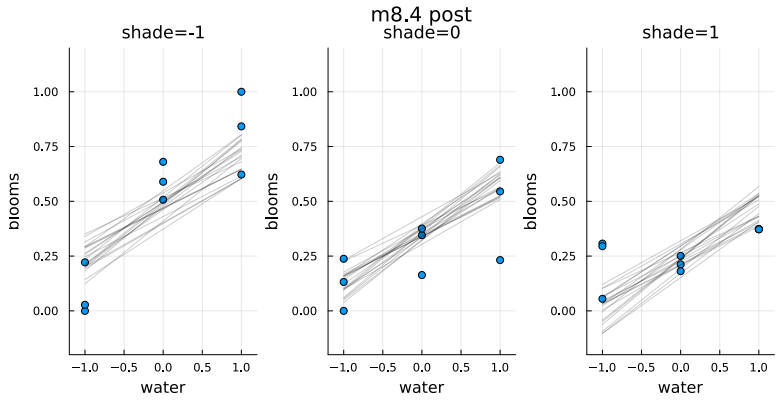
	variable	mean	min	median	max	nmis
1	:a	0.357703	0.202265	0.358896	0.46537	0
2	:bs	-0.110535	-0.213824	-0.111706	0.000542442	0
3	:bw	0.205985	0.10006	0.20617	0.326268	0
4	:bws	-0.142334	-0.264822	-0.142658	-0.00337029	0
5	:σ	0.143068	0.091804	0.140641	0.251181	0

```
begin
    m8_5_c = sample(m8_5(tulips.water_cent,
    tulips.shade_cent, tulips.blooms_std),
        NUTS(), 1000)
    m8_5_df = DataFrame(m8_5_c)
    describe(m8_5_df)
end
```

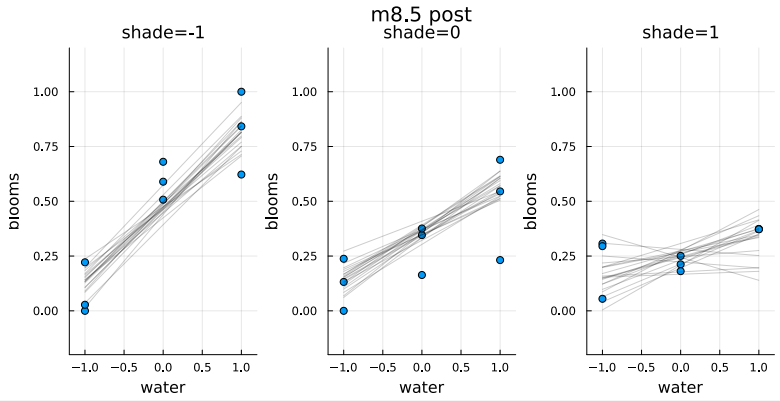
```
100%
```

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

Code 8.25




```
let
    plts = []

    for shade ∈ -1:1
        idx = findall(==(shade), tulips.shade_cent)
        p = plot(xlims=(-1.2,1.2), ylims=(-.2,1.2),
    xlab="water", ylab="blooms",
                 title="shade=$shade", titlefontsize=12)
        scatter!(tulips.water_cent[idx],
    tulips.blooms_std[idx])
        water_seq = -1:1
        mu = link(m8_4_df, (r, water) -> r.a + r.bw * water
    + r.bs * shade,
             water_seq)
        mu = hcat(mu...);
        for μ ∈ first(eachrow(mu), 20)
            plot!(water_seq, μ, c=:black, alpha=0.2)
        end
        push!(plts, p)
    end
    plot(plts..., layout=(1, 3), size=(800, 400),
    plot_title="m8.4 post",
        plot_titlefontsize=14)
end
```



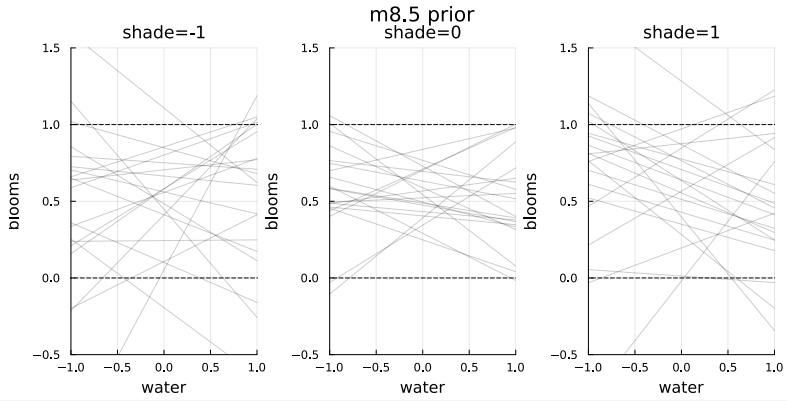

```
let
    plts = []

    for shade ∈ -1:1
        idx = findall(==(shade), tulips.shade_cent)
        p = plot(xlims=(-1.2,1.2), ylims=(-.2,1.2),
    xlab="water", ylab="blooms",
                 title="shade=$shade", titlefontsize=12)
        scatter!(tulips.water_cent[idx],
    tulips.blooms_std[idx])
        water_seq = -1:1
        mu = link(m8_5_df, (r, water) -> r.a + r.bw*water +
    r.bs*shade +
            r.bws*water*shade, water_seq)
        mu = hcat(mu...);
        for μ ∈ first(eachrow(mu), 20)
            plot!(water_seq, μ, c=:black, alpha=0.2)
        end
        push!(plts, p)
    end
    plot(plts..., layout=(1, 3), size=(800, 400),
    plot_title="m8.5 post",
        plot_titlefontsize=14)
end
```

Code 8.26

	a	bs	bw	bws	σ
1	0.390802	-0.410619	0.496406	0.282584	0.825049
2	0.475158	0.231532	-0.107238	-0.111999	0.718468
3	0.69497	0.587878	-0.0702015	-0.375238	1.20034
4	0.291231	0.48476	-0.307271	0.140144	1.83333
5	0.344055	0.291104	0.373787	-0.761392	0.276272
6	0.588535	0.176305	-0.273696	-0.0345404	0.673412
7	0.709762	-0.144856	0.272654	0.0777131	0.889652
8	0.522407	-0.326765	0.0288682	0.197716	0.143758
9	0.6318	-0.119233	-0.115755	-0.0735099	0.682547
10	0.567537	-0.0129589	0.0821298	-0.340122	1.40812
more					
1000	0.479217	-0.523019	0.126061	0.108316	0.475501

```
begin
    Random.seed!(7)
    m8_5p_c = sample(m8_5(tulips.water_cent,
    tulips.shade_cent, tulips.blooms_std),
        Prior(), 1000)
    m8_5p_df = DataFrame(m8_5p_c);
end
```

100%




```
let
    plts = []

    for shade ∈ -1:1
        p = plot(xlims=(-1, 1), ylims=(-0.5, 1.5),
    xlab="water", ylab="blooms",
                 title="shade=$shade", titlefontsize=12)
        water_seq = -1:1
        mu = link(m8_5p_df, (r, water) -> r.a + r.bw*water
    + r.bs*shade +
             r.bws*water*shade, water_seq)
        mu = hcat(mu...);
        for μ ∈ first(eachrow(mu), 20)
            plot!(water_seq, μ, c=:black, alpha=0.2)
        end
        hline!([0.0, 1.0], s=:dash, c=:black)
        push!(plts, p)
    end
    plot(plts..., layout=(1, 3), size=(800, 400),
    plot_title="m8.5 prior",
        plot_titlefontsize=14)
end
```