Chap 5.1

```
md"# Chap 5.1"
```

```
versioninfo()
```

```
Julia Version 1.11.0
Commit 501a4f25c2b (2024-10-07 11:40 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
  LLVM: libLLVM-16.0.6 (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
```

```
html"""
<style>
    main {
        margin: 0 auto;
        max-width: max(1800px, 75%);
        padding-left: max(5px, 1%);
        padding-right: max(350px, 10%);
    }
</style>
"""
```

Table of Contents


```
begin
  using Pkg, DrWatson
  using PlutoUI
  TableOfContents()
end
```

Note

Dagitty.jl needs to be replaced by CausalInference.jl. Dagitty is not part of SR2TuringPluto.jl.

```
md"

!!! note

Dagitty.jl needs to be replaced by CausalInference.jl. Dagitty is not part of
SR2TuringPluto.jl."
```

```
#Pkg.activate(expanduser("~/.julia/dev/SR2TuringPluto"))
```

```
begin
    using Distributions
    using Optim
    using StatsPlots
    using StatsBase
    using LaTeXStrings
    using CSV
    using DataFrames
    using LinearAlgebra
    using Logging
    using Random
    using Turing
    using Dagitty
    using StatisticalRethinking
    using StatisticalRethinkingPlots
end
```

```
Error requiring `Turing` from `StatisticalRethinking`
exception:
```

Error message from Main

```
LoadError: UndefVarError: `TuringOptimExt` not defined in
`StatisticalRethinking`

Suggestion: check for spelling errors or missing imports.

Hint: a global variable of this name also exists in TuringOptimExt.

in expression starting at
/y/home/huangyu/.julia/packages/StatisticalRethinking/Bzph1/src/require
/turing/turing_optim_sample.jl:3

in expression starting at
/y/home/huangyu/.julia/packages/StatisticalRethinking/Bzph1/src/require
/turing/turing.jl:7
```

Stack trace

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

1. `include(mod::Module, _path::String)`
   from | julia → Base.jl:557 docs
2. `include(x::String)`
   from | StatisticalRethinking → StatisticalRethinking.jl:1 docs
3. from | turing.jl:7
4. `include(mod::Module, _path::String)`
   from | julia → Base.jl:557 docs
5. `include(x::String)`
   from | StatisticalRethinking → StatisticalRethinking.jl:1 docs
6. from | Requires.jl:40
7. eval

Set defaults for plot and logging.

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

5.1 Spurious association.

Code 5.1

```
[-0.60629, -0.686699, -0.204241, -1.41039, 0.599857, -0.284651, 1.24313, 0.439037, 2.9317
```

```
begin
    d = CSV.read(sr_datadir("WaffleDivorce.csv"), DataFrame)
    d[!,:D] = standardize(ZScoreTransform, d.Divorce)
    d[!,:M] = standardize(ZScoreTransform, d.Marriage)
    d[!,:A] = standardize(ZScoreTransform, d.MedianAgeMarriage);
end
```

Code 5.2

```
1.2436303013880823
```

```
std(d.MedianAgeMarriage)
```

Code 5.3

```
m5_1 (generic function with 2 methods)
```

```
@model function 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
```

	variable	mean	min	median	max	nmissing	eltype
1	:a	-0.00665887	-0.550246	-0.00575653	0.58982	0	Float64
2	:bA	0.00168184	-1.71989	0.00890987	1.40209	0	Float64
3	:σ	1.06709	0.000728128	0.732709	6.1367	0	Float64

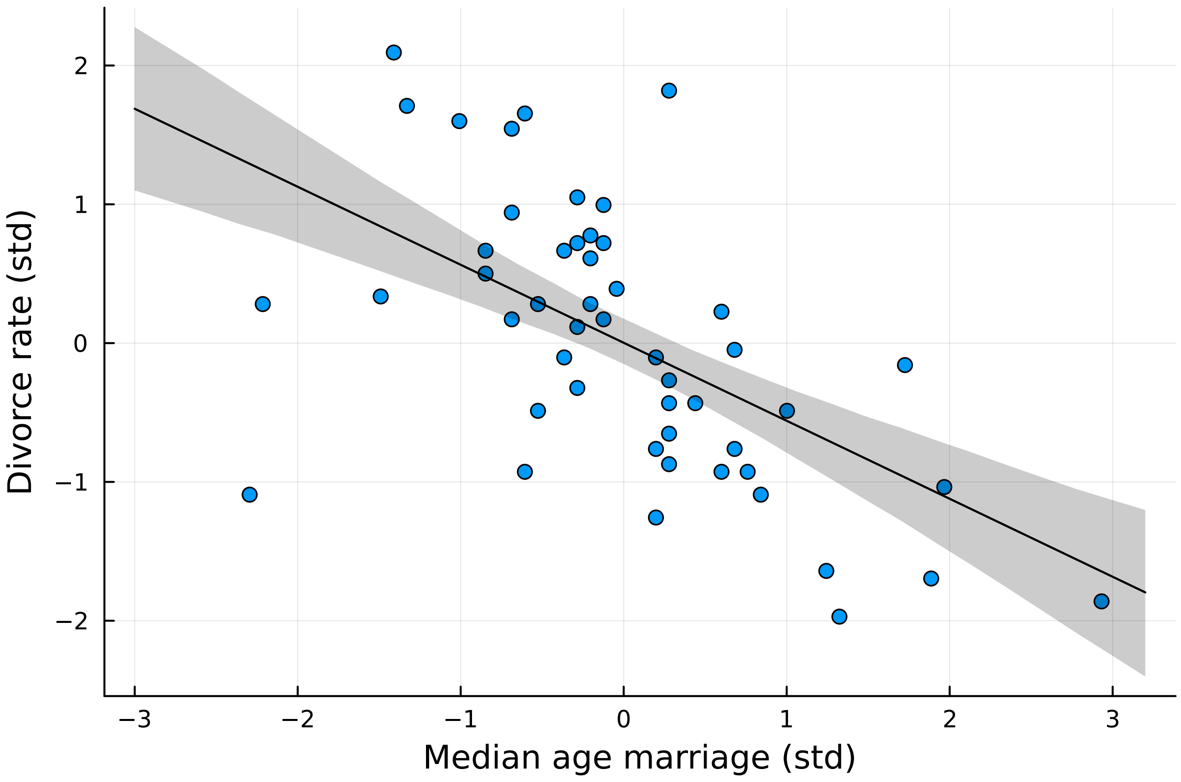
```
begin
    @time m5_1t = sample(m5_1(d.A, d.D), NUTS(), 1000)
    m5_1_df = DataFrame(m5_1t)
    @time prior = sample(m5_1([0], [0]), Prior(), 1000)
    prior_df = DataFrame(prior)
    describe(prior_df)
end
```

```
Sampling 100%

Found initial step size
ϵ: 0.05

Sampling 100%

0.421065 seconds (1.45 M allocations: 120.137 MiB, 7.21% gc time)
0.209363 seconds (590.67 k allocations: 26.836 MiB)
```

Code 5.4

```
let
    # calculate μ for every prior sample on age=-2 and age=2

    bounds = [-2, 2]
    μ = StatisticalRethinking.link(prior_df, [:a, :bA], bounds)
    μ = hcat(μ...);

    p = plot(xlab="Median age marriage (std)", ylab="Divorce rate (std)")
    for μₚ ∈ first(eachrow(μ), 50)
        plot!(bounds, μₚ; c=:black, alpha=0.3)
    end
end
```

Code 5.5




```
let
    A_seq = range(-3, 3.2; length=30)

    μ = StatisticalRethinking.link(m5_1_df, [:a, :bA], A_seq)
    μ = hcat(μ...)
    μ_mean = mean.(eachcol(μ))
    μ_PI = PI.(eachcol(μ))
    μ_PI = vcat(μ_PI'...)

    @df d scatter(:A, :D; xlab="Median age marriage (std)",
        ylab="Divorce rate (std)")
    plot!(A_seq, [μ_mean μ_mean]; c=:black, fillrange=μ_PI, fillalpha=0.2)
end
```

Code 5.6

```
m5_2 (generic function with 2 methods)
```

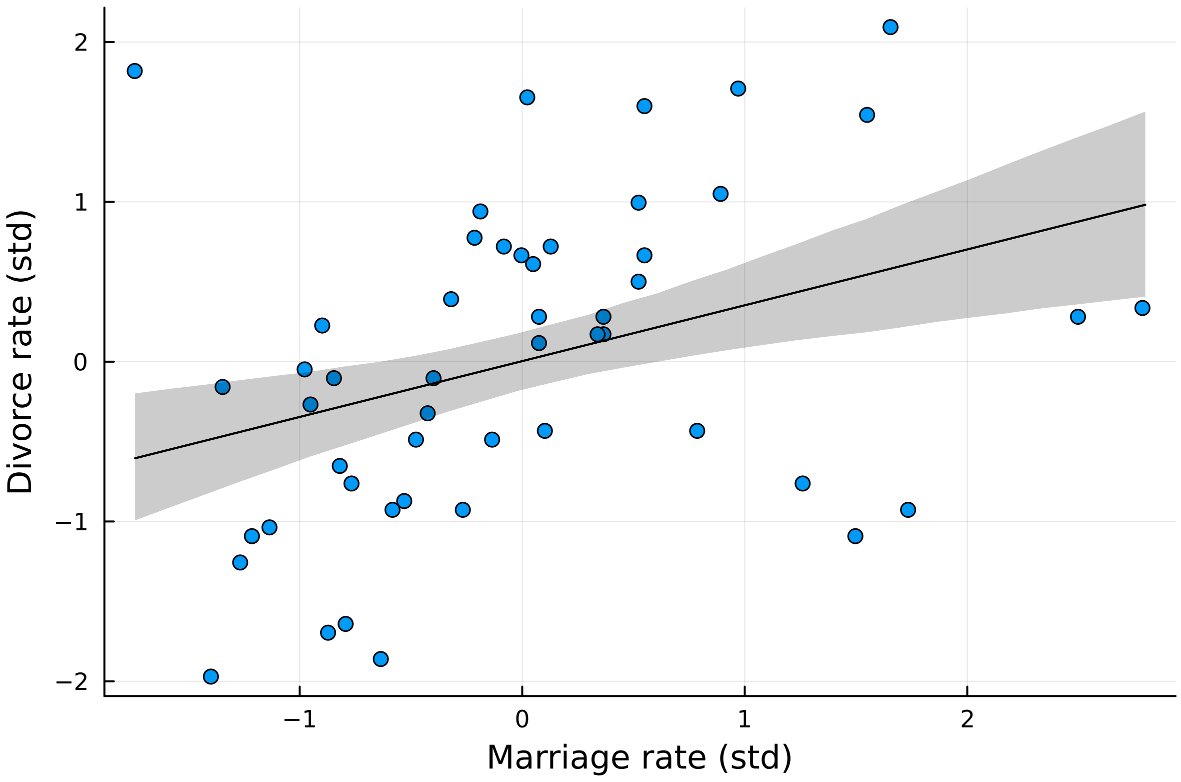
```
@model function m5_2(M, D)
    σ ~ Exponential(1)
    a ~ Normal(0, 0.2)
    bM ~ Normal(0, 0.5)
    μ = @. a + bM * M
    D ~ MvNormal(μ, σ)
end
```

	variable	mean	min	median	max	nmissing	eltype
1	:a	0.00353923	-0.397905	0.00728162	0.414083	0	Float64
2	:bM	0.349397	-0.147691	0.354223	0.692693	0	Float64
3	:σ	0.948557	0.701307	0.943838	1.2883	0	Float64

```
begin
    m5_2t = sample(m5_2(d.M, d.D), NUTS(), 1000)
    m5_2_df = DataFrame(m5_2t)
    describe(m5_2_df)
end
```

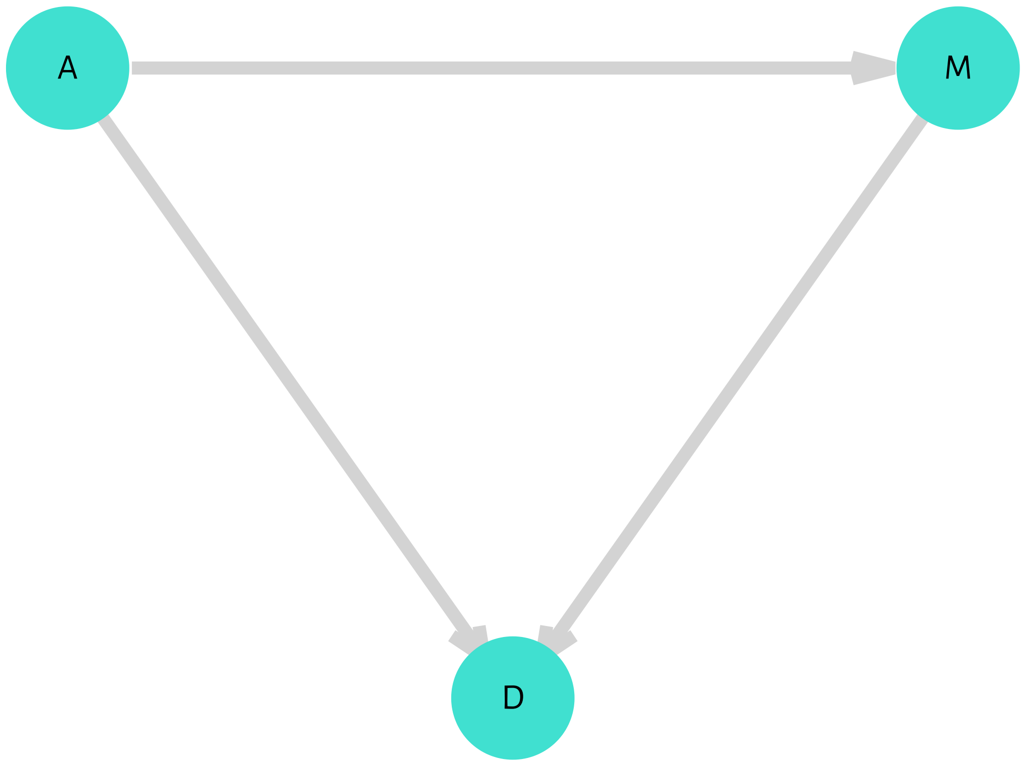
Sampling 100%

Found initial step size
ϵ: 0.4




```
let
    M_seq = range(-1.74, 2.8; length=30)

    μ = StatisticalRethinking.link(m5_2_df, [:a, :bM], M_seq)
    μ = hcat(μ...)
    μ_mean = mean.(eachcol(μ))
    μ_PI = PI.(eachcol(μ))
    μ_PI = vcat(μ_PI'...)

    @df d scatter(:M, :D; xlab="Marriage rate (std)", ylab="Divorce rate (std)")
    plot!(M_seq, [μ_mean μ_mean]; c=:black, fillrange=μ_PI, fillalpha=0.2)
end
```

Code 5.7




```
let
    g = Dagitty.DAG(:A => :M, :A => :D, :M => :D)
    drawdag(g, [0, 1, 2], [0, 1, 0])
end
```

Code 5.8

```
[ConditionalIndependence(:D, :M, [:A])]
```

```
let
    g = Dagitty.DAG(:A => :M, :A => :D)
    implied_conditional_independencies(g)
end
```

Code 5.9

```
[]
```

```
let
    g = Dagitty.DAG(:A => :M, :A => :D, :M => :D)
    implied_conditional_independencies(g)
end
```

Code 5.10

```
m5_3 (generic function with 2 methods)
```

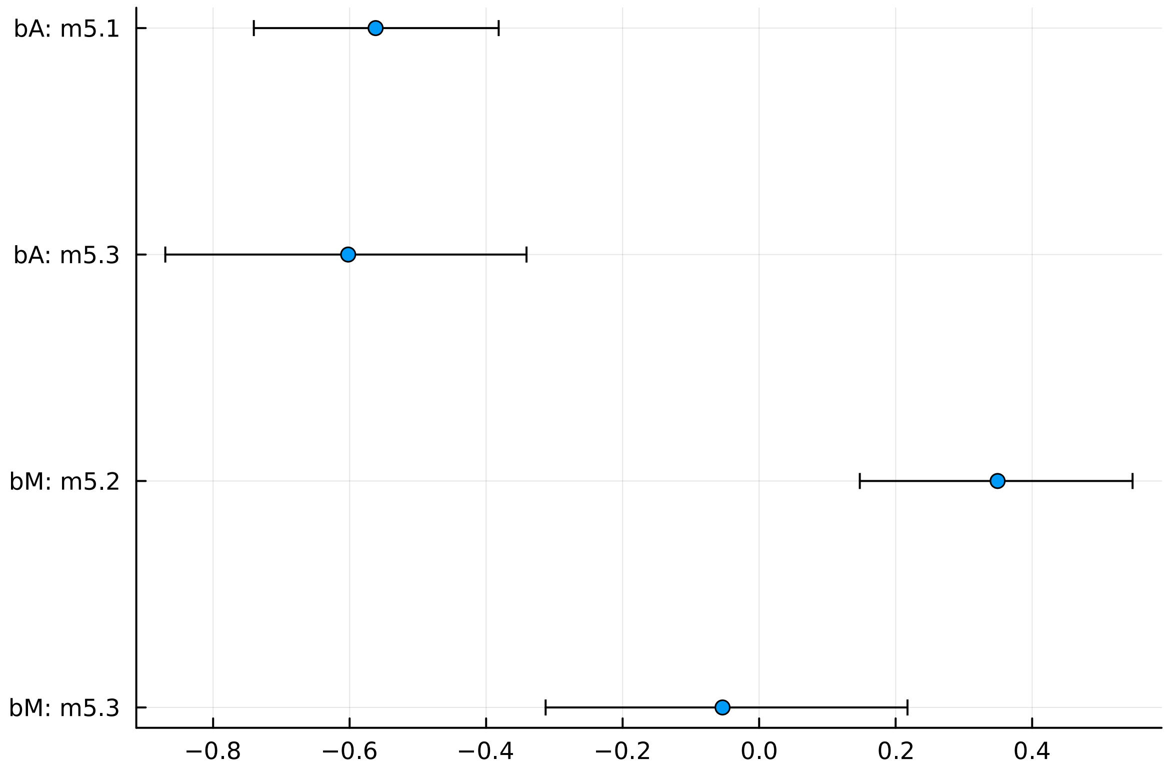
```
@model function m5_3(A, M, D)
    σ ~ Exponential(1)
    a ~ Normal(0, 0.2)
    bA ~ Normal(0, 0.5)
    bM ~ Normal(0, 0.5)
    μ = @. a + bA * A + bM * M
    D ~ MvNormal(μ, σ)
end
```

	variable	mean	min	median	max	nmissing	eltype
1	:a	-0.00320073	-0.3646	-0.00389141	0.293064	0	Float64
2	:bA	-0.602126	-1.0686	-0.606997	0.010838	0	Float64
3	:bM	-0.0536286	-0.67119	-0.0565273	0.481899	0	Float64
4	:σ	0.830328	0.615499	0.820962	1.26056	0	Float64

```
begin
    m5_3t = sample(m5_3(d.A, d.M, d.D), NUTS(), 1000)
    m5_3_df = DataFrame(m5_3t)
    describe(m5_3_df)
end
```

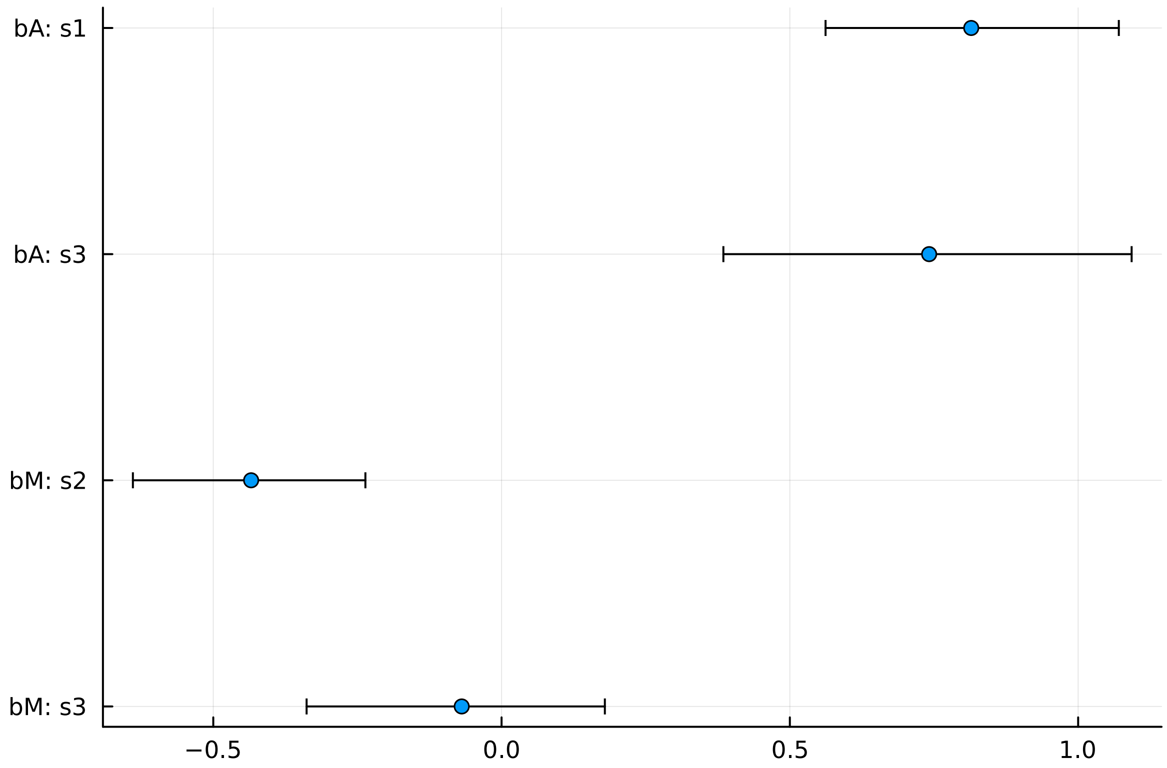
```
Sampling 100%

Found initial step size
ϵ: 0.05
```

Code 5.11



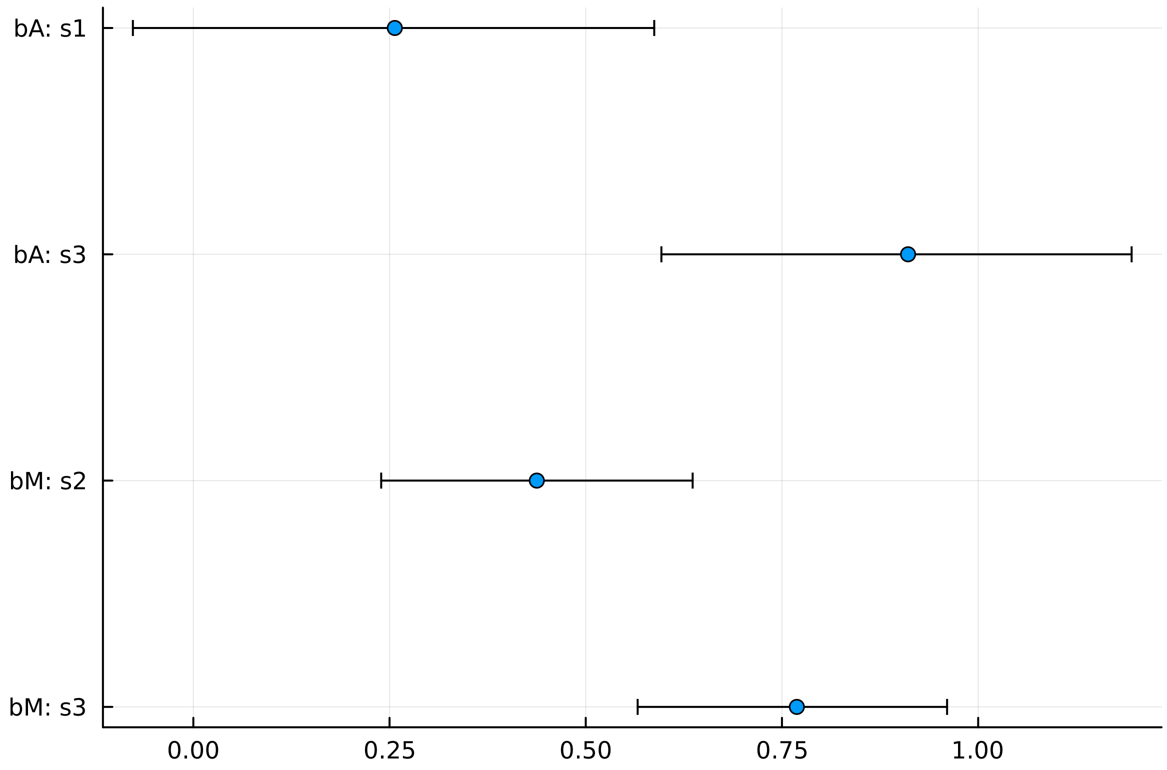

```
coeftab_plot(m5_1_df, m5_2_df, m5_3_df; pars=(:bA, :bM),
    names=["m5.1", "m5.2", "m5.3"])
```

Code 5.12




```
let
    N = 50
    age = rand(Normal(), N)
    mar = rand.(Normal.(-age))
    div = rand.(Normal.(age));

    s1 = DataFrame(sample(m5_1(age, div), NUTS(), 1000))
    s2 = DataFrame(sample(m5_2(mar, div), NUTS(), 1000))
    s3 = DataFrame(sample(m5_3(age, mar, div), NUTS(), 1000));
    coeftab_plot(s1, s2, s3; pars=(:bA, :bM), names=["s1", "s2", "s3"])
end
```

```
Sampling 100%

Found initial step size
ϵ: 0.4

Sampling 100%

Found initial step size
ϵ: 0.4

Sampling 100%

Found initial step size
ϵ: 0.4
```




```
let
    N = 50
    age = rand(Normal(), N)
    mar = rand.(Normal.(-age))
    div = rand.(Normal.(age .+ mar));

    s1 = DataFrame(sample(m5_1(age, div), NUTS(), 1000))
    s2 = DataFrame(sample(m5_2(mar, div), NUTS(), 1000))
    s3 = DataFrame(sample(m5_3(age, mar, div), NUTS(), 1000));
    coeftab_plot(s1, s2, s3; pars=(:bA, :bM), names=["s1", "s2", "s3"])
end
```

```
Sampling 100%

Found initial step size
ϵ: 0.4

Sampling 100%

Found initial step size
ϵ: 0.4

Sampling 100%

Found initial step size
ϵ: 0.025
```

Code 5.13

```
m5_4 (generic function with 2 methods)
```

```
@model function m5_4(A, M)
    σ ~ Exponential(1)
    a ~ Normal(0, 0.2)
    bAM ~ Normal(0, 0.5)
    μ = @. a + bAM * A
    M ~ MvNormal(μ, σ)
end
```

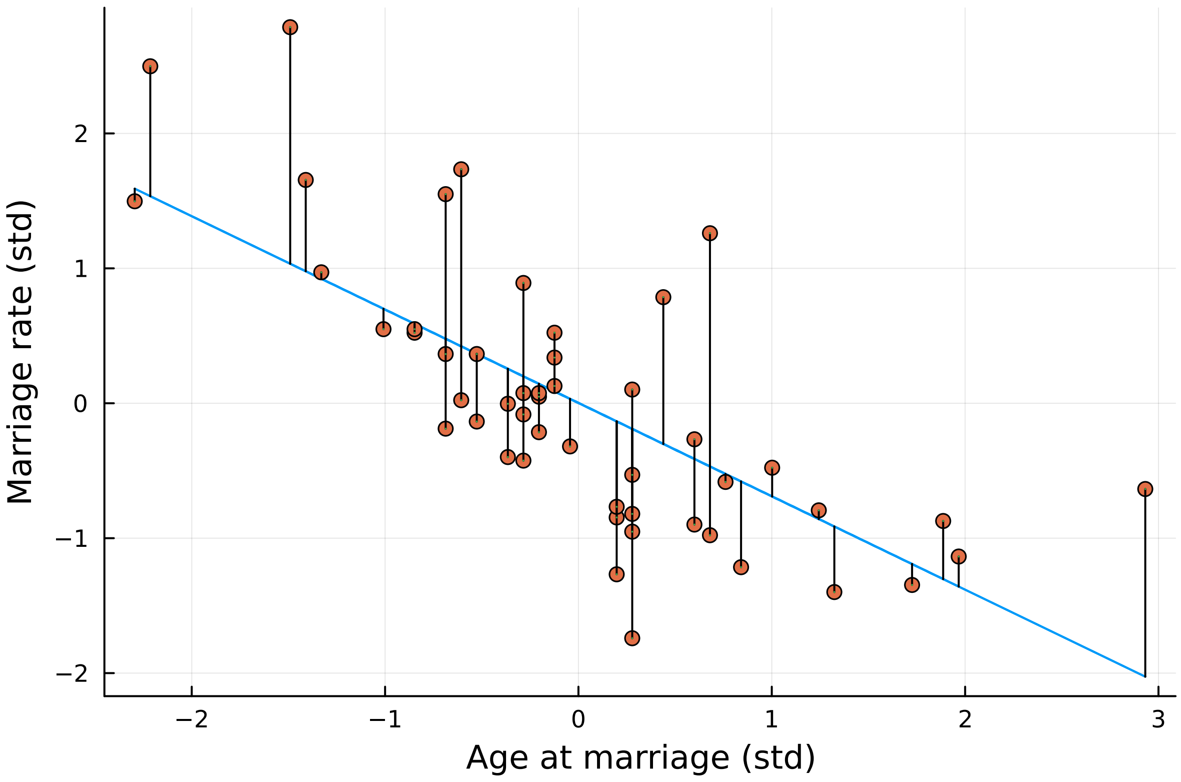
	a	bAM	σ
1	-0.163794	-0.69913	0.680176
2	-0.0146904	-0.738857	0.716546
3	-0.0146904	-0.738857	0.716546
4	0.100295	-0.696143	0.727446
5	-0.0477447	-0.741093	0.76938
6	0.0909004	-0.770785	0.702593
7	0.106044	-0.813987	0.637198
8	-0.10963	-0.585643	0.712284
9	0.0655574	-0.837556	0.683335
10	0.155692	-0.773712	0.780996
more			
1000	-0.0142223	-0.884129	0.774043

```
begin
    m5_4t = sample(m5_4(d.A, d.M), NUTS(), 1000)
    m5_4_df = DataFrame(m5_4t);
end
```

Sampling 100%

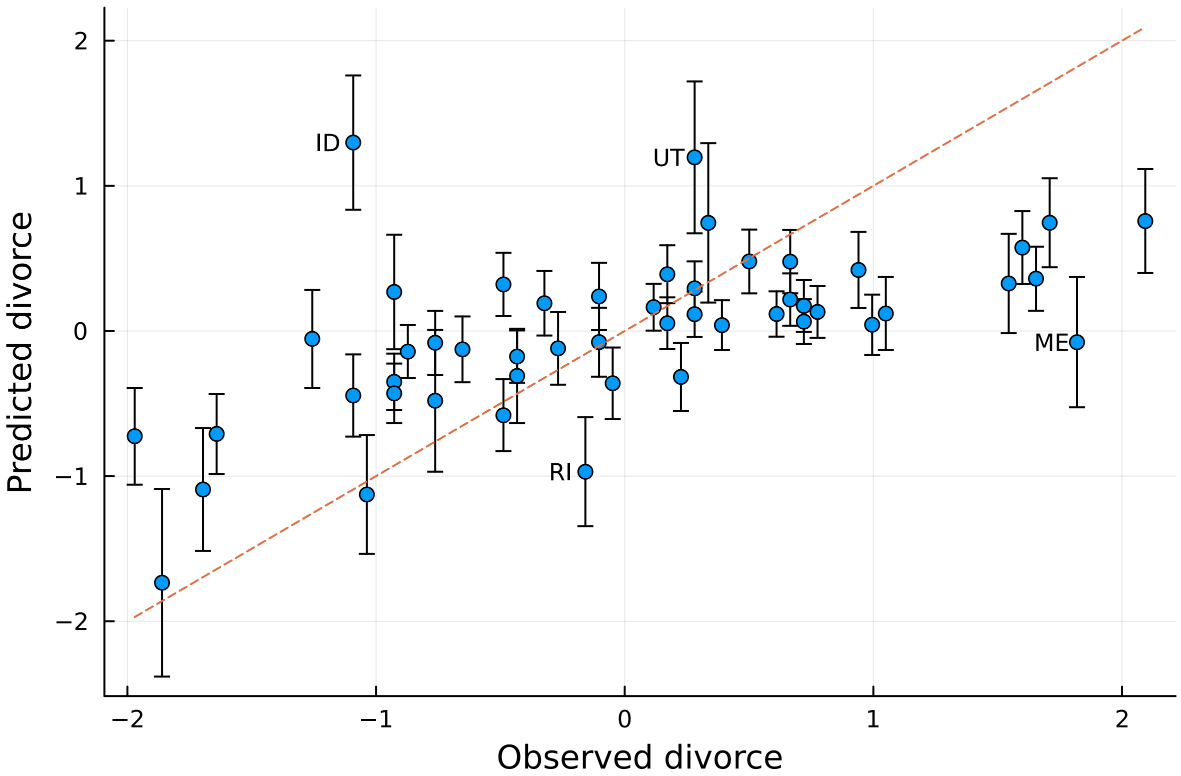
Found initial step size
ϵ: 0.2

Code 5.14




```
let
    mu = StatisticalRethinking.link(m5_4_df, [:a, :bAM], d.A);
    mu = hcat(mu...)
    mu_mean = mean.(eachcol(mu))
    mu_resid = mu_mean .- d.M;

    # +
    # Side-note: how to plot the residuals
    # getting yerr - list of 2-tuples with distance to the regression line
    yerr = collect(zip(-clamp.(mu_resid, -Inf, -0.0), clamp.(mu_resid, 0, Inf)));

    plot(d.A, mu_mean; xlab="Age at marriage (std)", ylab="Marriage rate (std)")
    scatter!(d.A, d.M)
    scatter!(d.A, d.M; yerr=yerr, markersize=0)
end
```

Code 5.15




```
let

    # explicit link form before I improved it

    mu = [
        @. r.a + r.bA * d.A + r.bM * d.M
        for r ∈ eachrow(m5_3_df)
    ]

    mu = vcat(mu'...)
    mu_mean = mean.(eachcol(mu))
    mu_PI = PI.(eachcol(mu))
    mu_PI = vcat(mu_PI'...);

    D_sim = [
        rand(MvNormal((@. r.a + r.bA * d.A + r.bM * d.M), r.σ))
        for r ∈ eachrow(m5_3_df)
    ]
    D_sim = vcat(D_sim'...);
    D_PI = PI.(eachcol(D_sim))
    D_PI = vcat(D_PI'...);
    # -

    # Code 5.16

    yerr = mu_PI[:,2] .- mu_mean
    scatter(d.D, mu_mean; xlab="Observed divorce", ylab="Predicted divorce",
        yerr=yerr)
    plot!(x->x; style=:dash)

    # Code 5.17

    loc_flags = d.Loc .∈ (["ID", "UT", "RI", "ME"],);
    loc_idxes = findall(loc_flags);
    anns = [
        (d.D[idx] - 0.1, mu_mean[idx], (d.Loc[idx], 8))
        for idx in loc_idxes
    ]
    annotate!(anns)
end
```

Code 5.18

	variable	mean	min	median	max	nmissing	eltype
1	:y	0.15905	-3.48325	0.0670999	3.66854	0	Float64
2	:x_real	0.0826649	-2.13272	-0.0501045	2.71956	0	Float64
3	:x_spur	-0.0222719	-3.59033	-0.140535	3.03337	0	Float64

```
let
    N = 100
    x_real = rand(Normal(), N)
    x_spur = rand.(Normal.(x_real))
    y = rand.(Normal.(x_real))
    df = DataFrame(:y => y, :x_real => x_real, :x_spur => x_spur)
    describe(df)
end
```

Code 5.19

m5_3A (generic function with 2 methods)

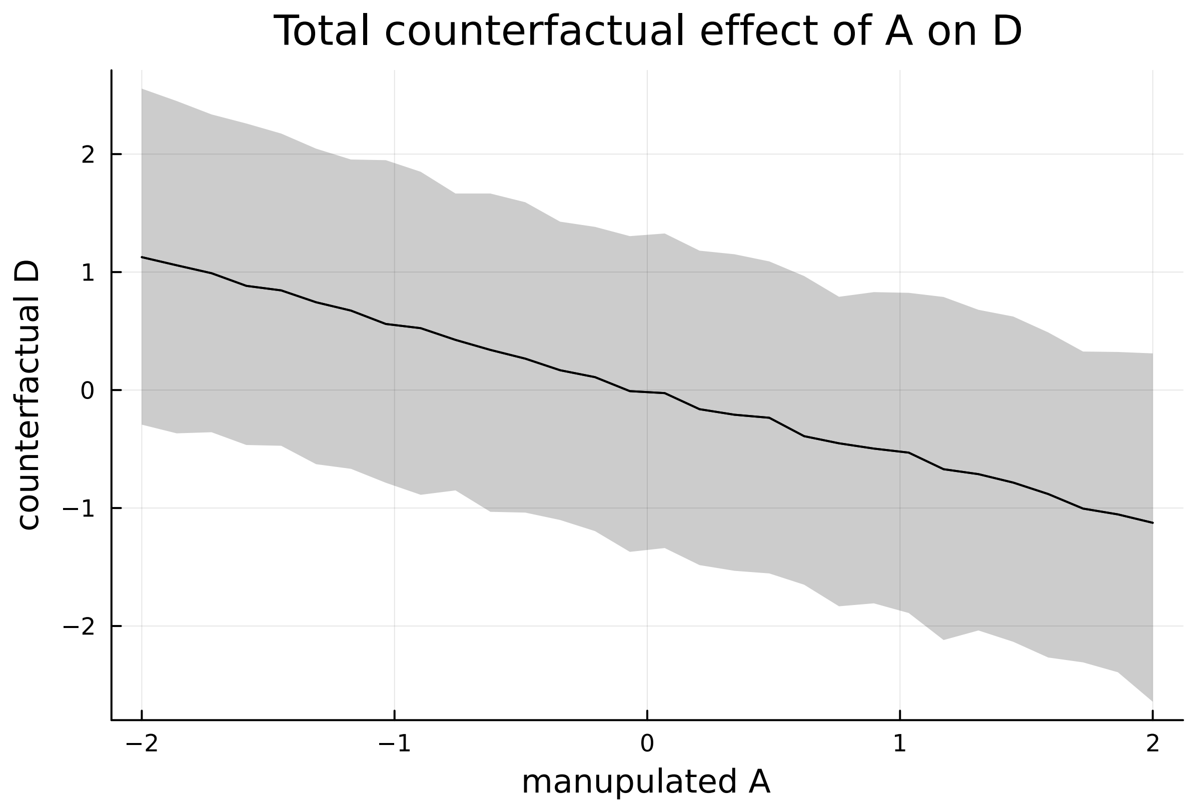
```
@model function m5_3A(A, M, D)
    # A → D ← M
    σ ~ Exponential(1)
    a ~ Normal(0, 0.2)
    bA ~ Normal(0, 0.5)
    bM ~ Normal(0, 0.5)
    μ = @. a + bA * A + bM * M
    D ~ MvNormal(μ, σ)
    # A → M
    σ_M ~ Exponential(1)
    aM ~ Normal(0, 0.2)
    bAM ~ Normal(0, 0.5)
    μ_M = @. aM + bAM * A
    M ~ MvNormal(μ_M, σ_M)
end
```

	variable	mean	min	median	max	nmissing	eltype
1	:a	0.00248769	-0.28923	0.00456427	0.348169	0	Float64
2	:aM	0.000473606	-0.305489	0.000514117	0.292643	0	Float64
3	:bA	-0.612116	-1.06976	-0.612527	-0.0825706	0	Float64
4	:bAM	-0.692251	-0.991939	-0.689197	-0.396236	0	Float64
5	:bM	-0.0656085	-0.592612	-0.0655231	0.480857	0	Float64
6	:σ	0.823608	0.602889	0.81688	1.13432	0	Float64
7	:σ_M	0.712641	0.509175	0.705253	1.12785	0	Float64

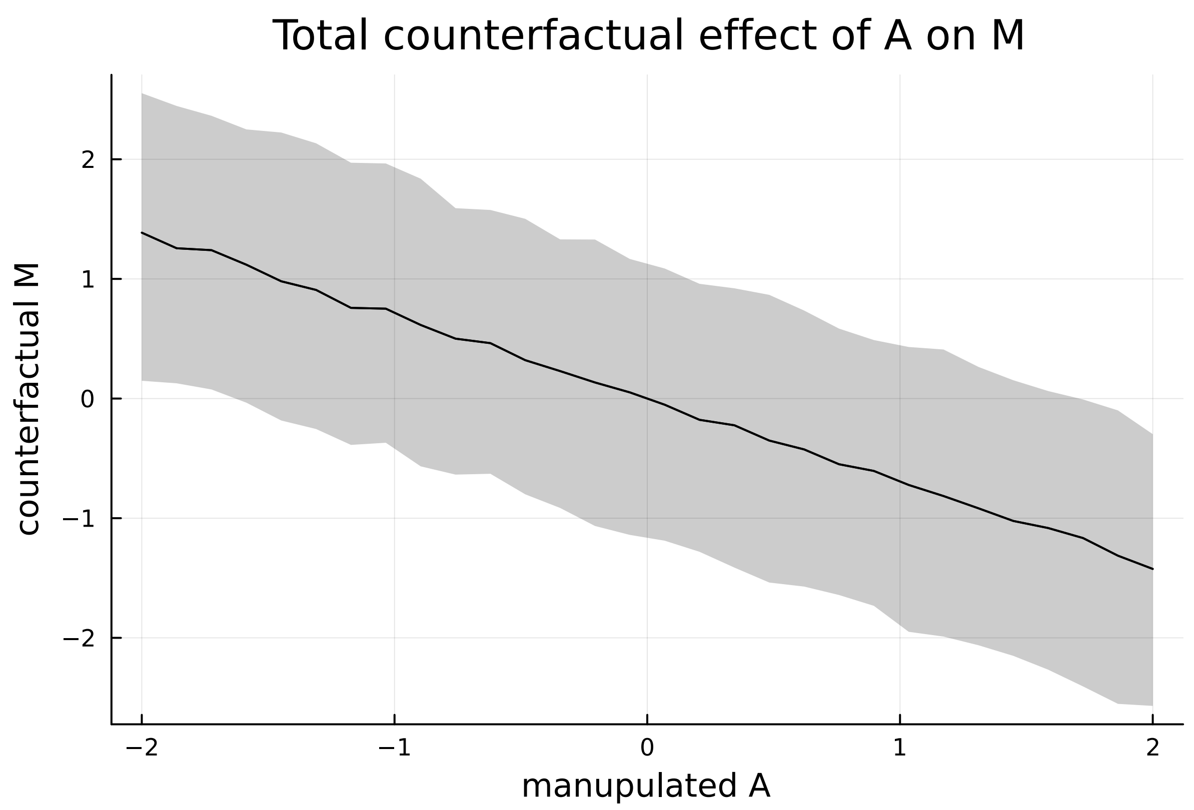
```
begin
    d1 = CSV.read(sr_datadir("WaffleDivorce.csv"), DataFrame)
    d2 = DataFrame(
        :D => standardize(ZScoreTransform, d1.Divorce),
        :M => standardize(ZScoreTransform, d1.Marriage),
        :A => standardize(ZScoreTransform, d1.MedianAgeMarriage),
    );

    m5_3At = sample(m5_3A(d2.A, d2.M, d2.D), NUTS(), 1000)
    m5_3A_df = DataFrame(m5_3At)
    describe(m5_3A_df)
end
```

```
Sampling 100%

Found initial step size
ϵ: 0.2
```

Code 5.20 - 5.22

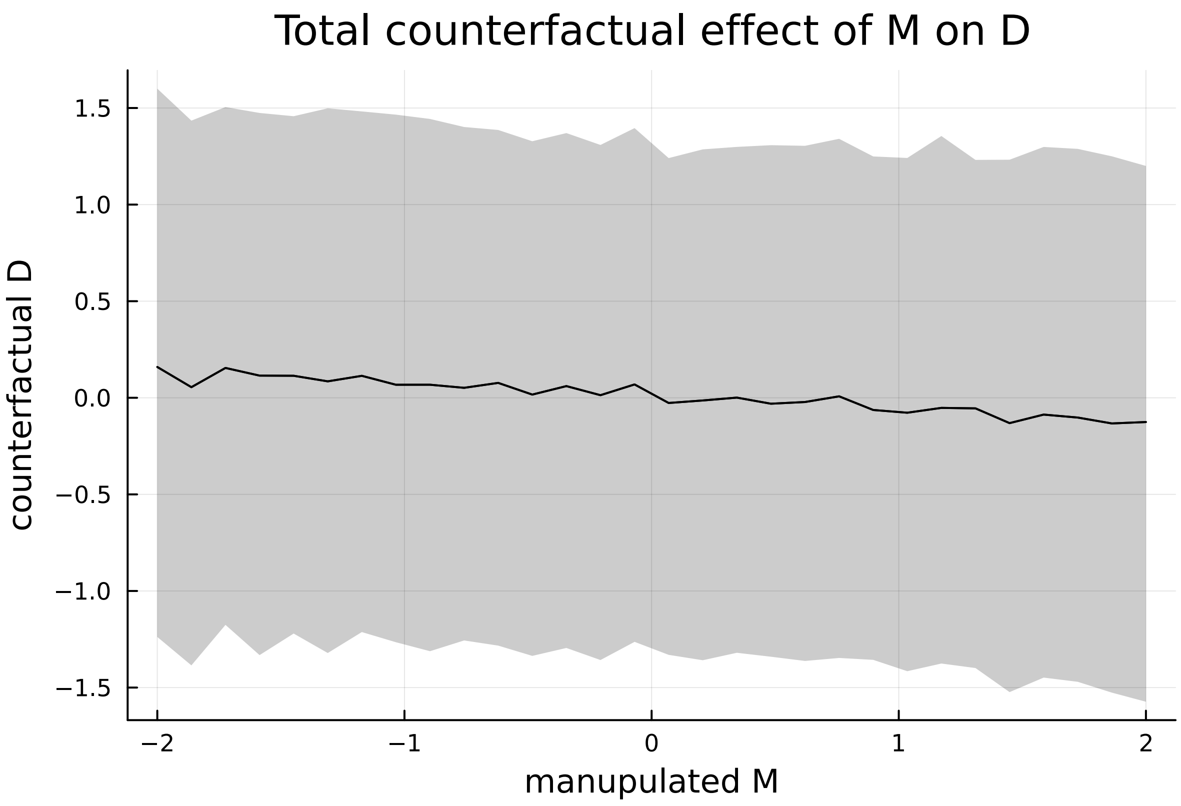



```
let
    A_seq = range(-2, 2; length=30);
    global s_M, s_D = [], []

    for r ∈ eachrow(m5_3A_df)
        M = rand(MvNormal((@. r.aM + r.bAM * A_seq), r.σ_M))
        D = rand(MvNormal((@. r.a + r.bA * A_seq + r.bM * M), r.σ))
        push!(s_M, M)
        push!(s_D, D)
    end

    s_M = vcat(s_M'...)
    s_D = vcat(s_D'...);
    μ_D = mean.(eachcol(s_D))
    PI_D = vcat(PI.(eachcol(s_D))'...)

    plot(
        A_seq, [μ_D, μ_D];
        fillrange=PI_D, fillalpha=0.2, color=:black,
        xlab="manupulated A", ylab="counterfactual D",
        title="Total counterfactual effect of A on D"
    )
end
```




```
let
    μ_M = mean.(eachcol(s_M))
    PI_M = vcat(PI.(eachcol(s_M))'...)

    plot(
        A_seq, [μ_M, μ_M];
        fillrange=PI_M, fillalpha=0.2, color=:black,
        xlab="manupulated A", ylab="counterfactual M",
        title="Total counterfactual effect of A on M"
    )
end
```

Code 5.23




```
let
    sim2_A = @. ([20, 30] - 26.1) / 1.24;
    s2_M, s2_D = [], []

    for r ∈ eachrow(m5_3A_df)
        M = rand(MvNormal((@. r.aM + r.bAM * sim2_A), r.σ_M))
        D = rand(MvNormal((@. r.a + r.bA * sim2_A + r.bM * M), r.σ))
        push!(s2_M, M)
        push!(s2_D, D)
    end

    s2_M = vcat(s2_M'...)
    s2_D = vcat(s2_D'...);
    mean(s2_D[:,2] - s2_D[:,1])
    # -

    # Code 5.24

    # +
    M_seq = range(-2, 2; length=30)
    s_D = []

    for r ∈ eachrow(m5_3A_df)
        # A is zero, so, we drop it from the μ term
        D = rand(MvNormal((@. r.a + r.bM * M_seq), r.σ))
        push!(s_D, D)
    end

    s_D = vcat(s_D'...);

    μ_D = mean.(eachcol(s_D))
    PI_D = vcat(PI.(eachcol(s_D))'...)

    plot(
        M_seq, [μ_D, μ_D];
        fillrange=PI_D, fillalpha=0.2, color=:black,
        xlab="manupulated M", ylab="counterfactual D",
        title="Total counterfactual effect of M on D"
    )
end
```

Code 5.25

```
A_seq = range(-2, 2; length=30);
```

Code 5.26

```
1000×30 Matrix{Float64}:
  1.1554    -0.0967398  1.7475      -0.609207  …  -0.155994  -1.4304    -2.64872
  0.627854   1.03038    1.57602      0.656687     -0.980522  -0.444945  -1.34432
  2.51643    0.668524   1.31688      3.21321      -1.37382   -1.49756   -0.481347
  1.38541    1.63349    2.08705      1.55058      -0.473135  -0.10908    0.0909892
  1.64257    2.29597    0.0414396    1.08773      -1.25094    0.209401  -1.51585
  3.60858    2.7533     0.00811689   1.32349   …  -2.16212   -1.80128   -2.15166
 -0.173748   2.19753    2.16598      1.82233      -1.2943    -0.463673  -0.411417
  ⋮                                           ⋱
  2.01813    0.102639   0.982616     1.00084      -0.113485  -1.75049   -0.737452
  0.724954   0.986207   1.17019      1.6036    …  -1.37783    0.619275   0.22252
 -1.17693    0.765622   0.706637     1.789        -3.3898    -0.931097  -1.42229
  1.58779    1.89716    2.47812      0.868086     -1.89673   -1.57939   -0.945112
 -0.465909   3.01401    1.83149      0.806376     -2.61712   -1.68308   -2.02923
  1.66261    1.32864    0.0211641    1.70964       0.496661  -0.487398  -1.06269
```

```julia
let
    s_M = [
        rand(MvNormal((@. r.aM + r.bAM * A_seq), r.σ_M))
        for r ∈ eachrow(m5_3A_df)
    ]
    s_M = vcat(s_M'...);

    # Code 5.27

    s_D = [
        rand(MvNormal((@. r.a + r.bA * A_seq + r.bM * M), r.σ))
        for (r, M) ∈ zip(eachrow(m5_3A_df), eachrow(s_M))
    ]
    s_D = vcat(s_D'...);
end
```