See chapter 9 in Regression and Other Stories.

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

```

```
using Pkg ✓ , DrWatson ✓
```

```
begin
    using GLM ✓

    # Specific to ROSStanPluto
    using StanSample ✓

    # Graphics related
    using GLMakie ✓

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

```
Replacing docs for `RegressionAndOtherStories.tr
DataFrame, AbstractString}` in module `Regressic
```

9.1 Propagating uncertainty in inference using posterior simulations.

hibbs =

	year	growth	vote	inc_party_candidat
1	1952	2.4	44.6	"Stevenson"
2	1956	2.89	57.76	"Eisenhower"
3	1960	0.85	49.91	"Nixon"
4	1964	4.21	61.34	"Johnson"
5	1968	3.02	49.6	"Humphrey"
6	1972	3.62	61.79	"Nixon"
7	1976	1.08	48.95	"Ford"
8	1980	-0.39	44.7	"Carter"
9	1984	3.86	59.17	"Reagan"
10	1988	2.27	53.94	"Bush, Sr."
⋮ more				
16	2012	0.95	52.0	"Obama"

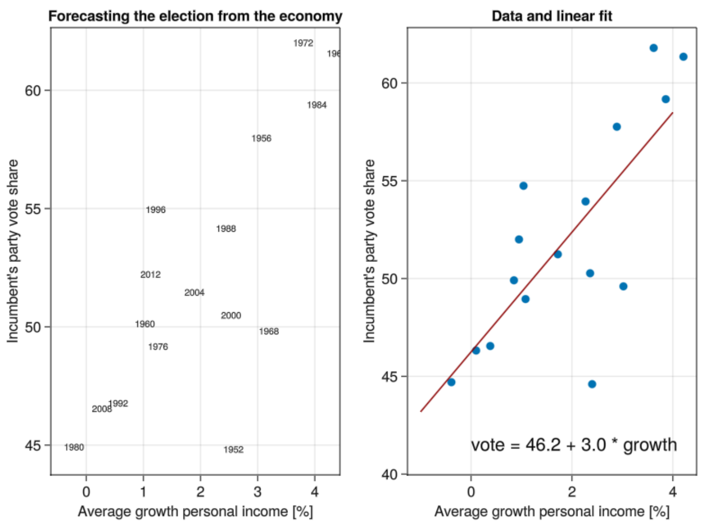
```
hibbs =
CSV.read(ros_datadir("ElectionsEconomy",
"hibbs.csv"), DataFrame)
```

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

vote ~ 1 + growth

Coefficients:
────────────────────────────────────────────────
                 Coef.  Std. Error      t  Pr(>|t
────────────────────────────────────────────────
(Intercept)  46.2476     1.62193    28.51    <1e-
growth        3.06053    0.696274    4.40    0.00
────────────────────────────────────────────────
```

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




```
let
    fig = Figure()
    hibbs.label = string.(hibbs.year)
    xlabel = "Average growth personal
    income [%]"
    ylabel = "Incumbent's party vote share"
    let
        title = "Forecasting the election
        from the economy"
        ax = Axis(fig[1, 1]; title, xlabel,
        ylabel)
        for (ind, yr) in
        enumerate(hibbs.year)
            annotations!("$(yr)"; position=
            (hibbs.growth[ind],
            hibbs.vote[ind]), textsize=10)
        end
    end
    let
        x = LinRange(-1, 4, 100)
        title = "Data and linear fit"
        ax = Axis(fig[1, 2]; title, xlabel,
        ylabel)
        scatter!(hibbs.growth, hibbs.vote)
        lines!(x, coef(hibbs_lm)[1] .+
        coef(hibbs_lm)[2] .* x;
        color=:darkred)
        annotations!("vote = 46.2 + 3.0 *
        growth"; position=(0, 41))
    end
    fig
end
```

```
stan7_1 = "
data {
    int<lower=1> N;       // total number of
    observations
    vector[N] growth;     // Independent
    variable: growth
    vector[N] vote;       // Dependent
    variable: votes
}
parameters {
    real b;               // Coefficient
    independent variable
    real a;               // Intercept
    real<lower=0> sigma; // dispersion
    parameter
}
model {
    vector[N] mu;

    // priors including constants
    a ~ normal(50, 20);
    b ~ normal(2, 10);
    sigma ~ exponential(1);

    mu = a + b * growth;

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

	parameters	mean	mcse	std
1	"b"	3.01923	0.016644	0.671776
2	"a"	46.3485	0.0384014	1.57594
3	"sigma"	3.60832	0.0133485	0.621676

```
let
    data = (N=nrow(hibbs), vote=hibbs.vote,
    growth=hibbs.growth)
    global m7_1s = SampleModel("hibbs",
    stan7_1)
    global rc7_1s = stan_sample(m7_1s; data)
    success(rc7_1s) && describe(m7_1s)
end
```

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

```
2169.02
```

```
let
    ss7_1s = describe(m7_1s)
    ss7_1s[:sigma, :ess]
end
```

```
2169.02
```

```
let
    ss7_1s = describe(m7_1s; showall=true)
    ss7_1s[:sigma, :ess]
end
```

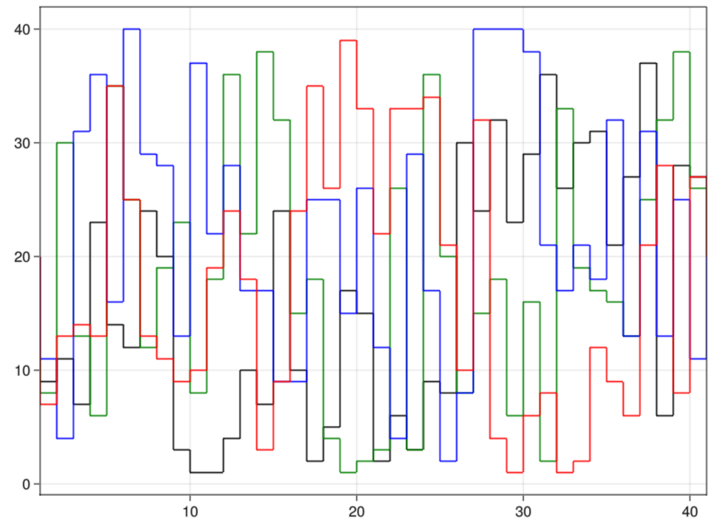
	parameters	mean	mcse	
1	"lp__"	-31.9347	0.0343773	1.3
2	"accept_stat__"	0.913995	0.00182292	0.1
3	"stepsize__"	0.418669	0.00720391	0.0
4	"treedepth__"	2.587	0.011743	0.6
5	"n_leapfrog__"	7.42	0.0637245	3.9
6	"divergent__"	0.0	NaN	0.0
7	"energy__"	33.4318	0.0484177	1.7
8	"b"	3.01923	0.016644	0.6
9	"a"	46.3485	0.0384014	1.5
10	"sigma"	3.60832	0.0133485	0.6

```
describe(m7_1s; showall=true)
```

post7_1s =

	b	a	sigma
1	2.4296	46.9814	3.79408
2	2.58025	47.5254	4.10587
3	2.30726	47.2282	4.65373
4	3.07832	47.1264	3.61459
5	2.69067	45.5456	3.6129
6	2.58458	46.0218	3.85655
7	3.09983	45.7698	3.98921
8	2.94073	46.6183	4.04931
9	2.00617	49.353	4.9111
10	1.18727	48.4547	5.04701
⋮ more			
4000	3.28506	46.152	3.04823

```
post7_1s = success(rc7_1s) &&
read_samples(m7_1s, :dataframe)
```




```
trankplot(post7_1s, "b")
```

ms7_1s =

	parameters	median	mad_sd	mean	st
1	"a"	46.32	1.486	46.348	1.57
2	"b"	3.023	0.654	3.019	0.67
3	"sigma"	3.531	0.602	3.608	0.62

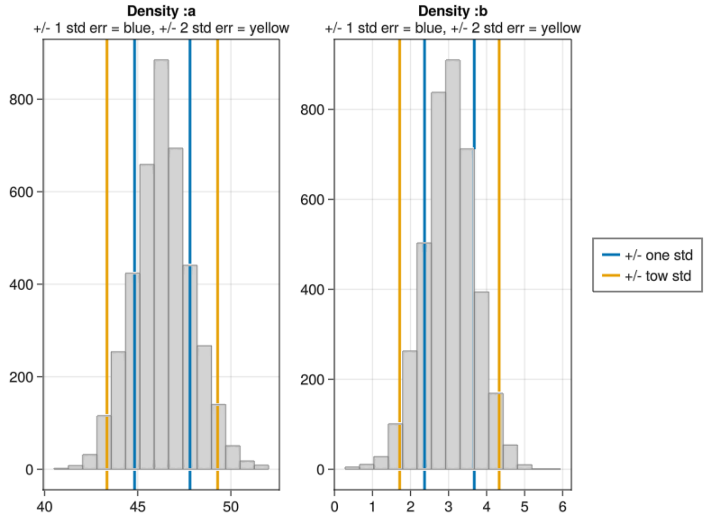
```
ms7_1s = model_summary(post7_1s, [:a, :b,
:sigma])
```

```
sims = 4000×3 Matrix{Float64}:
        2.4296   46.9814  3.79408
        2.58025  47.5254  4.10587
        2.30726  47.2282  4.65373
        3.07832  47.1264  3.61459
        2.69067  45.5456  3.6129
        2.58458  46.0218  3.85655
        3.09983  45.7698  3.98921
        ⋮
        1.76017  48.5191  3.48972
        3.07896  46.1482  3.34962
        3.36978  45.2588  3.34669
        3.21821  45.3887  2.78174
        2.85801  46.2397  4.31696
        3.28506  46.152   3.04823
```

```
sims = Array(post7_1s)
```

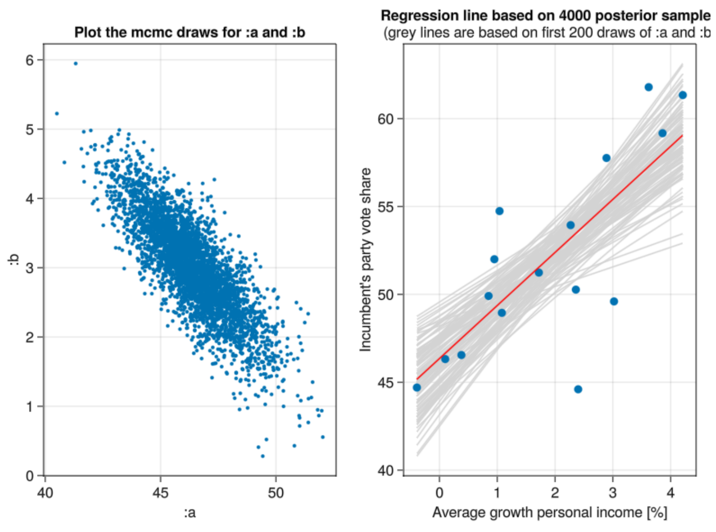
```
1×3 Matrix{Float64}:
 3.02268  46.3198  3.53123
```

```julia
median(sims; dims=1)
```




```julia
let
    f = Figure()
    ax = Axis(f[1, 1]; title="Density :a",
    subtitle="+/- 1 std err = blue, +/- 2
    std err = yellow")
    hist!(post7_1s.a; bins=15, color =
    :lightgrey, strokewidth = 1, strokecolor
     = :grey)
    one = vlines!([ms7_1s[:a, :median] -
    ms7_1s[:a, :mad_sd], ms7_1s[:a,
    :median] + ms7_1s[:a, :mad_sd]];
    linewidth=3)
    two = vlines!([ms7_1s[:a, :median] -
    2ms7_1s[:a, :mad_sd], ms7_1s[:a,
    :median] + 2ms7_1s[:a, :mad_sd]];
    linewidth=3)

    ax = Axis(f[1, 2]; title="Density :b",
    subtitle="+/- 1 std err = blue, +/- 2
    std err = yellow")
    hist!(post7_1s.b; bins=15, color =
    :lightgrey, strokewidth = 1, strokecolor
     = :grey)
    vlines!([ms7_1s[:b, :median] -
    ms7_1s[:b, :mad_sd], ms7_1s[:b,
    :median] + ms7_1s[:b, :mad_sd]];
    linewidth=3)
    vlines!([ms7_1s[:b, :median] -
    2ms7_1s[:b, :mad_sd], ms7_1s[:b,
    :median] + 2ms7_1s[:b, :mad_sd]];
    linewidth=3)
    Legend(f[1, 3], [one, two], ["+/- one
    std", "+/- tow std"])
    f
end
```




```
let
    growth_range =
    LinRange(minimum(hibbs.growth),
    maximum(hibbs.growth), 200)
    votes = mean.(link(post7_1s, (r,x) ->
    r.a + x * r.b, growth_range))

    xlabel = "Average growth personal
    income [%]"
    ylabel = "Incumbent's party vote share"

    fig = Figure()

    ax = Axis(fig[1, 1]; title="Plot the
    mcmc draws for :a and :b", xlabel=":a",
    ylabel=":b")
    scatter!(post7_1s.a, post7_1s.b;
    markersize=4)

    xlabel = "Average growth personal
    income [%]"
    ylabel="Incumbent's party vote share"
    ax = Axis(fig[1, 2]; title="Regression
    line based on 4000 posterior samples",
        subtitle = "(grey lines are based
        on first 200 draws of :a and :b)",
        xlabel, ylabel)
    for i in 1:100
        lines!(growth_range, post7_1s.a[i]
        .+ post7_1s.b[i] .* growth_range,
        color = :lightgrey)
    end
    scatter!(hibbs.growth, hibbs.vote)
    lines!(growth_range, votes, color =
    :red)
    fig
```

```
end
```

9.2 Prediction and uncertainty.

	x	y
1	-2.0	50
2	-1.0	44
3	0.0	50
4	1.0	47
5	2.0	56

```
let
    x = LinRange(-2, 2, 5)
    y = [50, 44, 50, 47, 56]
    global sexratio = DataFrame(x = x, y =
    y)
end
```

```
stan9_1 = "
data {
    int<lower=1> N; // total number of
    observations
    vector[N] x;    // Independent
    variable: growth
    vector[N] y;    // Dependent variable:
    votes
}
parameters {
    real b;                 // Coefficient
    independent variable
    real a;                 // Intercept
    real<lower=0> sigma; // dispersion
    parameter
}
model {
    vector[N] mu;

    // priors including constants
    a ~ normal(50, 5);
    b ~ normal(0, 5);
    sigma ~ uniform(0, 10);

    mu = a + b * x;

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

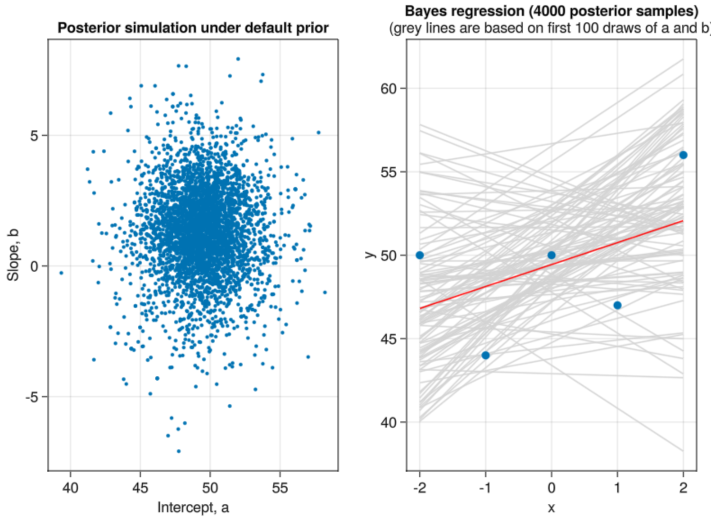
	parameters	mean	mcse	std
1	"b"	1.3147	0.0492384	1.70412
2	"a"	49.439	0.036813	2.2286
3	"sigma"	5.44629	0.0921171	1.85281

```
let
    data = (N = nrow(sexratio), x =
    sexratio.x, y = sexratio.y)
    global m9_1s = SampleModel("m9_1s",
    stan9_1)
    global rc9_1s = stan_sample(m9_1s; data)
    success(rc9_1s) && describe(m9_1s)
end
```

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

	parameters	median	mad_sd	mean
1	"a"	49.429	1.945	49.439
2	"b"	1.364	1.501	1.315
3	"sigma"	5.152	1.966	5.446

```
if success(rc9_1s)
    post9_1s = read_samples(m9_1s,
    :dataframe)
    sm9_1s = model_summary(post9_1s, [:a,
    :b, :sigma])
end
```




```
let
    x_range = LinRange(minimum(sexratio.x),
    maximum(sexratio.x), 200)
    y = mean.(link(post9_1s, (r,x) -> r.a +
    x * r.b, x_range))

    xlabel = "x"
    ylabel = "y"

    fig = Figure()

    ax = Axis(fig[1, 1]; title="Posterior
    simulation under default prior",
    xlabel="Intercept, a", ylabel="Slope,
    b")
    scatter!(post9_1s.a, post9_1s.b;
    markersize=4)

    ax = Axis(fig[1, 2]; title="Bayes
    regression (4000 posterior samples)",
        subtitle = "(grey lines are based
        on first 100 draws of a and b)",
        xlabel, ylabel)
    for i in 1:100
        lines!(x_range, post9_1s.a[i] .+
        post9_1s.b[i] .* x_range, color =
        :lightgrey)
    end
    scatter!(sexratio.x, sexratio.y)
    lines!(x_range, y, color = :red)
    fig
end
```

```
stan9_2 = "
data {
    int<lower=1> N; // total number of
    observations
    vector[N] x;    // Independent
    variable: growth
    vector[N] y;    // Dependent variable:
    votes
}
parameters {
    real b;              // Coefficient
    independent variable
    real a;              // Intercept
    real<lower=0> sigma; // dispersion
    parameter
}
model {
    vector[N] mu;

    // priors including constants
    a ~ normal(48.8, 0.2);
    b ~ normal(0, 0.2);
    sigma ~ uniform(0, 10);

    mu = a + b * x;

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

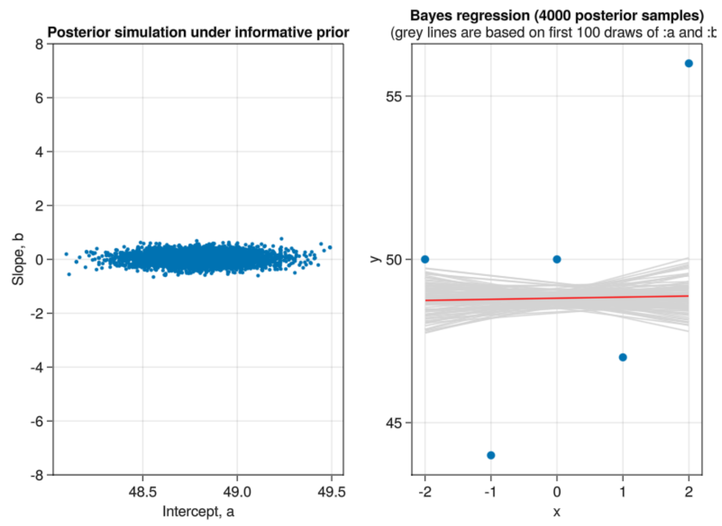
	parameters	mean	mcse	std
1	"b"	0.033602	0.00333889	0.199175
2	"a"	48.8097	0.0033849	0.202378
3	"sigma"	5.14479	0.0342823	1.68717

```
let
    data = (N = nrow(sexratio), x =
    sexratio.x, y = sexratio.y)
    global m9_2s = SampleModel("m9_2s",
    stan9_2)
    global rc9_2s = stan_sample(m9_2s; data)
    success(rc9_2s) && describe(m9_2s)
end
```

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

	parameters	median	mad_sd	mean	st
1	"a"	48.808	0.203	48.81	0.2(
2	"b"	0.029	0.194	0.034	0.1(
3	"sigma"	4.844	1.623	5.145	1.6(

```
if success(rc9_2s)
    post9_2s = read_samples(m9_2s,
    :dataframe)
    sm9_2s = model_summary(post9_2s, [:a,
    :b, :sigma])
end
```




```
let
    x_range = LinRange(minimum(sexratio.x),
    maximum(sexratio.x), 200)
    y = mean.(link(post9_2s, (r,x) -> r.a +
    x * r.b, x_range))

    xlabel = "x"
    ylabel = "y"

    fig = Figure()

    ax = Axis(fig[1, 1]; title="Posterior
    simulation under informative prior",
    xlabel="Intercept, a", ylabel="Slope,
    b")
    ylims!(ax, -8, 8)
    scatter!(post9_2s.a, post9_2s.b;
    markersize=4)

    ax = Axis(fig[1, 2]; title="Bayes
    regression (4000 posterior samples)",
        subtitle = "(grey lines are based
        on first 100 draws of :a and :b)",
        xlabel, ylabel)
    for i in 1:100
        lines!(x_range, post9_2s.a[i] .+
        post9_2s.b[i] .* x_range, color =
        :lightgrey)
    end
    scatter!(sexratio.x, sexratio.y)
    lines!(x_range, y, color = :red)
    fig
end
```

9.3 Prior information and Bayesian synthesis.

Prior based on a previously-fitted model using economic and political condition.

```
begin
    theta_hat_prior = 0.524
    se_prior = 0.041
end;
```

Survey of 400 people, of whom 190 say they will vote for the Democratic candidate.

```
begin
    n = 400
    y = 190
end;
```

Data estimate.

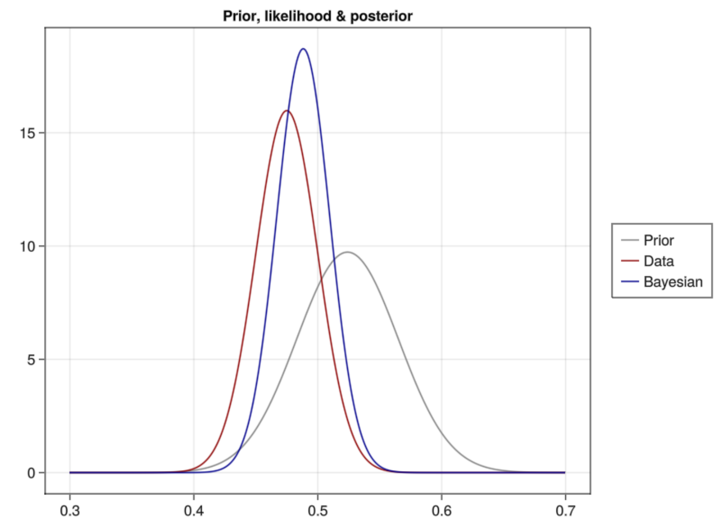
```
theta_hat_data = 0.475
theta_hat_data = y/n
```

```
se_data = 0.02496873044429772
se_data = √((y/n)*(1-y/n)/n)
```

Bayes estimate.

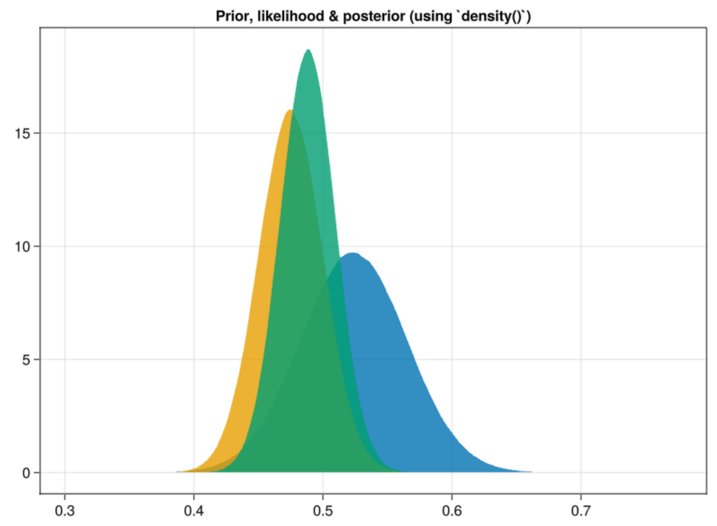
```
theta_hat_bayes = 0.48825635323153693
theta_hat_bayes =
(theta_hat_prior/se_prior^2 +
    theta_hat_data/se_data^2) /(1/se_prior^2
     + 1/se_data^2)
```

```
se_bayes = 0.02132543263776925
se_bayes = sqrt(1/(1/se_prior^2 +
1/se_data^2))
```




```
let
    x = 0.3:0.001:0.7
    f = Figure()
    ax = Axis(f[1, 1], title="Prior,
    likelihood & posterior")
    prior = lines!(f[1, 1], x, pdf.
    (Normal(theta_hat_prior, se_prior), x),
    color=:gray)
    data = lines!(x, pdf.
    (Normal(theta_hat_data, se_data),
    x),color=:darkred)
    bayes = lines!(x, pdf.
    (Normal(theta_hat_bayes, se_bayes), x),
    color=:darkblue)
    Legend(f[1, 2], [prior, data, bayes],
    ["Prior", "Data", "Bayesian"])

    current_figure()
end
```




```
let
    f = Figure()
    ax = Axis(f[1, 1], title="Prior,
    likelihood & posterior (using
    `density()`)")
    density!(rand(Normal(theta_hat_prior,
    se_prior), Int(1e6)), lab="prior")
    density!(rand(Normal(theta_hat_data,
    se_data), Int(1e6)), lab="likelihood")
    density!(rand(Normal(theta_hat_bayes,
    se_bayes), Int(1e6)), lab="bayes")
    current_figure()
end
```

9.4 Example of Bayesian inference: beauty and sex ratio.