This notebook is based on chapter 9 in Statistical Rethinking.

Care and feeding of Markov chains

```
html"""
<style>
    main {
        margin: 0 auto;
        max-width: 2000px;
        padding-left: max(160px, 10%);
        padding-right: max(100px, 10%);
    }
</style>
"""
```

```
using Pkg ✓
```

```
begin
    # Specific to ROSStanPluto
    using StanSample ✓

    # Graphics related
    using GLMakie ✓

    # Include basic packages
    using RegressionAndOtherStories ✓
end
```

```
Replacing docs for `RegressionAndOtherStories.trankpl
rames.DataFrame, AbstractString}` in module `Regressi
`
```

```
stan9_2 = "
data {
    int n;
    vector[n] y;
}
parameters {
    real alpha;
    real<lower=0> sigma;
}
model {
    real mu;
    alpha ~ normal(0, 1000);
    sigma ~ exponential(0.0001);
    mu = alpha;
    y ~ normal(mu, sigma);
}";
```

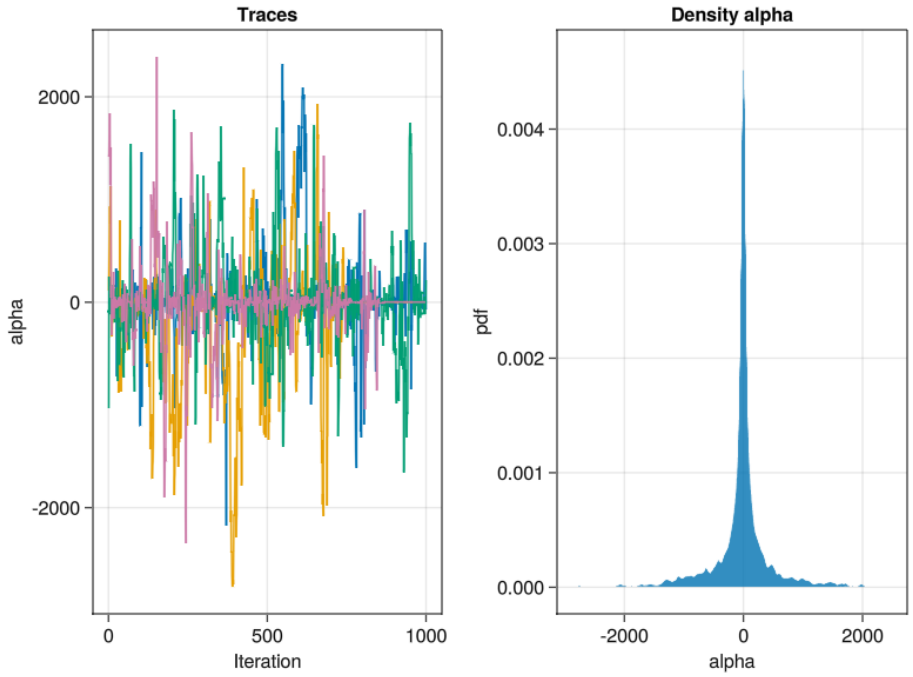
	parameters	mean	mcse	std	5%	
1	"alpha"	-22.243	32.0576	478.952	-874.001	0
2	"sigma"	798.287	93.8203	1672.87	10.4871	2

```
let
    Random.seed!(123)
    data = (n=2, y=[-1, 1])
    global m9_2s = SampleModel("m9_2s", stan9_2)
    global rc9_2s = stan_sample(m9_2s; data)
    success(rc9_2s) && describe(m9_2s)
end
```

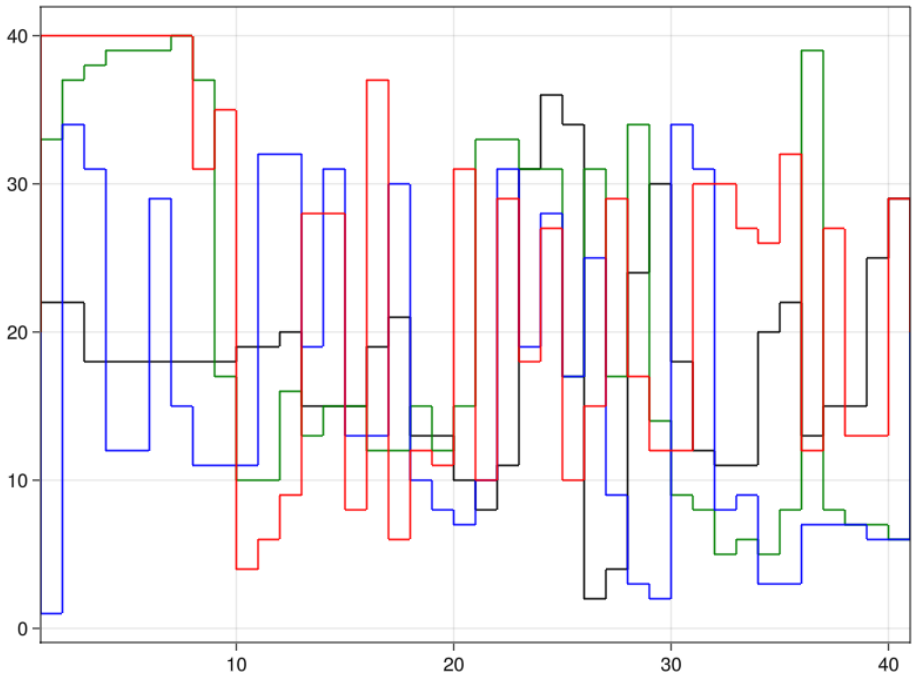
```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T/jl_e
updated.
```

	parameters	median	mad_sd	mean	std
1	"alpha"	0.762	127.055	-22.243	478.952
2	"sigma"	231.207	293.574	798.287	1672.87

```
if success(rc9_2s)
    post9_2s = read_samples(m9_2s, :dataframe)
    ms9_2s = model_summary(post9_2s, [:alpha,
    :sigma])
end
```




```
plot_chains(post9_2s, [:alpha])
```




```
trankplot(post9_2s, "alpha")
```

```
stan9_3 = "
data {
    int n;
    vector[n] y;
}
parameters {
    real alpha;
    real<lower=0> sigma;
}
model {
    real mu;
    alpha ~ normal(0, 1);
    sigma ~ exponential(1);
    mu = alpha;
    y ~ normal(mu, sigma);
}";
```

	parameters	mean	mcse	std	5
1	"alpha"	-0.000934169	0.0136699	0.650532	-1.
2	"sigma"	1.37925	0.0139107	0.672726	0.6

```
let
    Random.seed!(123)
    data = (n=2, y=[-1, 1])
    global m9_3s = SampleModel("m9.3s", stan9_3)
    global rc9_3s = stan_sample(m9_3s; data)
    success(rc9_3s) && describe(m9_3s)
end
```

```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T/jl_H
updated.
```

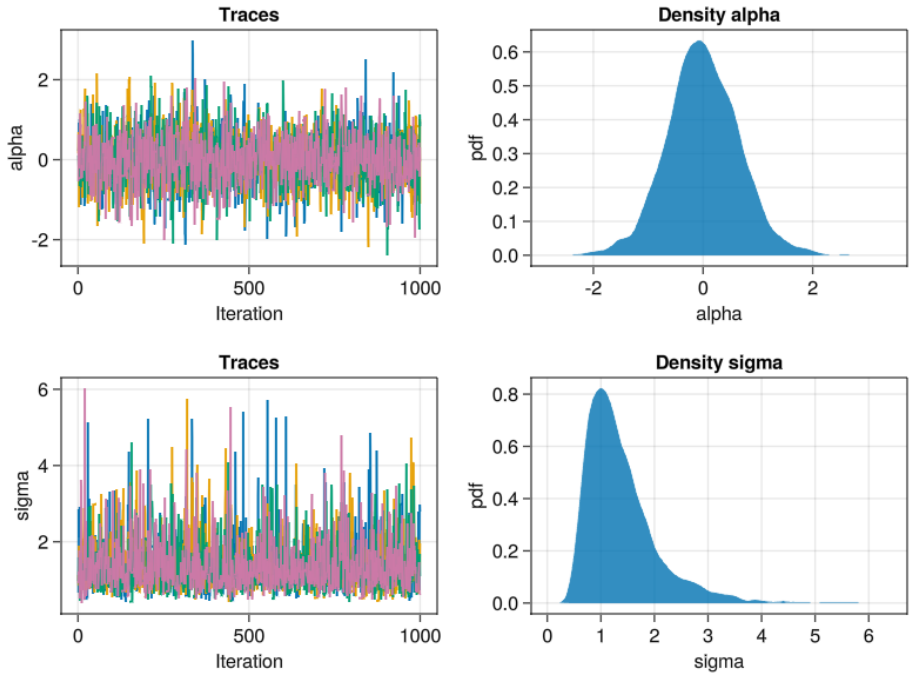
	alpha	sigma
1	0.217657	0.665899
2	0.328003	2.86695
3	0.923458	2.11215
4	0.476047	2.58393
5	1.01984	2.92127
6	-0.384908	0.835844
7	0.0187461	1.16952
8	-1.13623	1.02151
9	0.00714906	1.11492
10	0.246382	0.647248
⋮	more	
4000	-0.0506899	0.604653

```
if success(rc9_3s)
    post9_3s = read_samples(m9_3s, :dataframe)
end
```

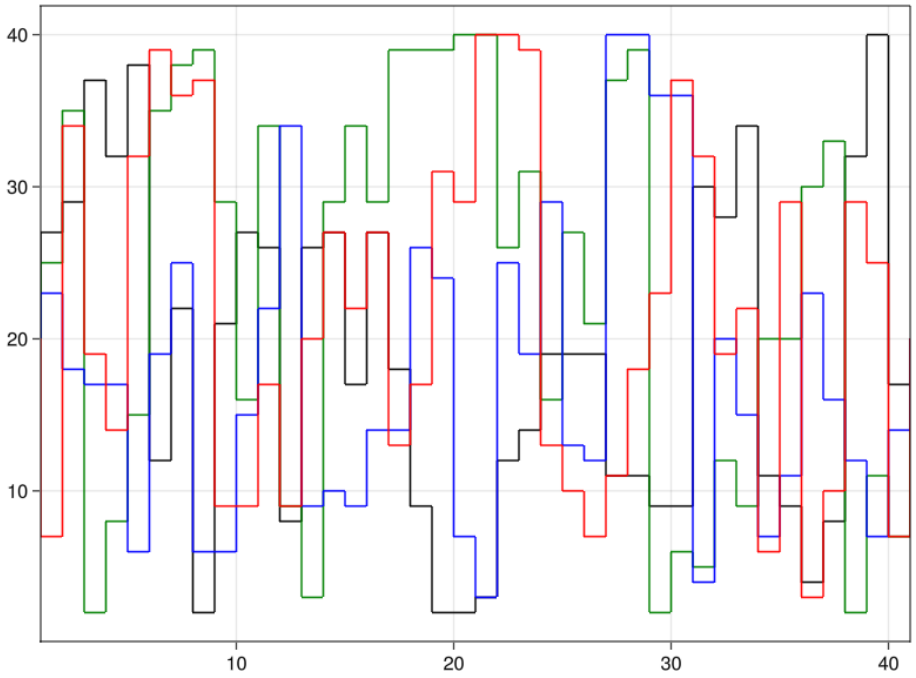
ms9_3s =

	parameters	median	mad_sd	mean	std
1	"alpha"	-0.017	0.624	-0.001	0.651
2	"sigma"	1.224	0.518	1.379	0.673

```
ms9_3s = model_summary(post9_3s, [:alpha, :sigma])
```




```
plot_chains(post9_3s, [:alpha, :sigma])
```



```
trankplot(post9_3s, "alpha")
```

```
stan9_4 = "
data {
    int n;
    vector[n] y;
}
parameters {
    real alpha;
    real beta;
    real<lower=0> sigma;
}
model {
    real mu;
    alpha ~ normal(0, 100);
    beta ~ normal(0, 1000);
    sigma ~ exponential(1);
    mu = alpha + beta;
    y ~ normal(mu, sigma);
}";
```

	parameters	mean	mcse	std	5%
1	"alpha"	-6.7726	10.2174	104.782	-183.4
2	"beta"	6.77717	10.2173	104.781	-163.7
3	"sigma"	1.09611	0.00527366	0.0783639	0.9755

```
begin
    Random.seed!(1)
    data9_4s = (n = 100, y = rand(Normal(0, 1),
100))
    m9_4s = SampleModel("m9.4s", stan9_4)
    rc9_4s = stan_sample(m9_4s; data=data9_4s)
    success(rc9_4s) && describe(m9_4s)
end
```

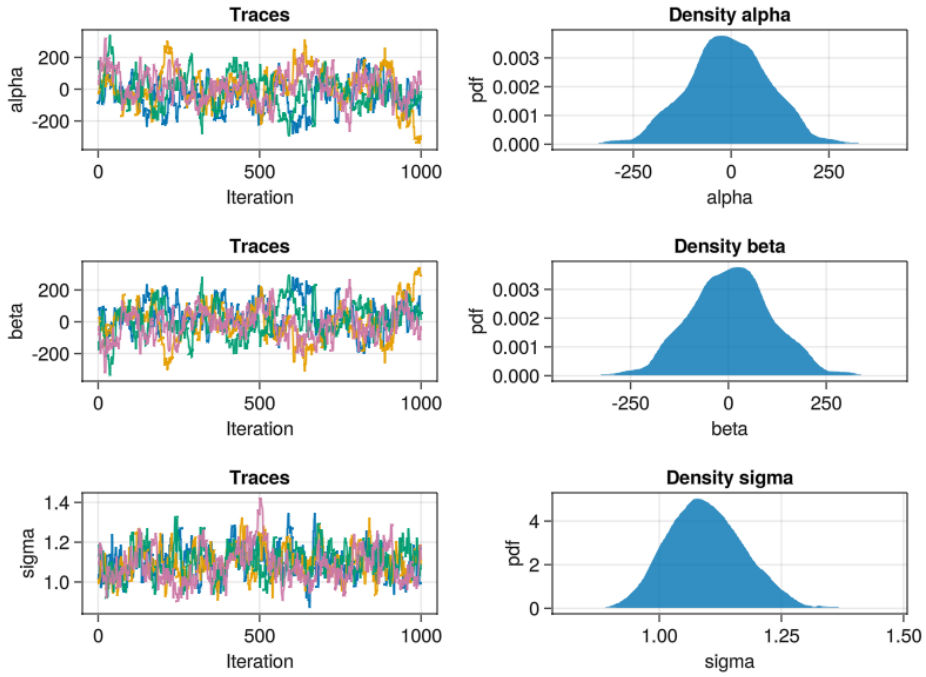
```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T/jl_d
updated.
```

	alpha	beta	sigma
1	-66.6067	66.668	1.01229
2	-85.607	85.6065	1.04282
3	-85.7241	85.6941	1.01502
4	-82.1121	82.1377	1.10425
5	-52.1738	52.1297	1.08983
6	-22.5539	22.5254	1.0715
7	-9.34204	9.33467	1.05689
8	-16.9432	16.9596	0.975596
9	-25.6934	25.6744	0.967967
10	-35.2967	35.3679	0.977145
⋮	more		
4000	54.3078	-54.3876	1.10962

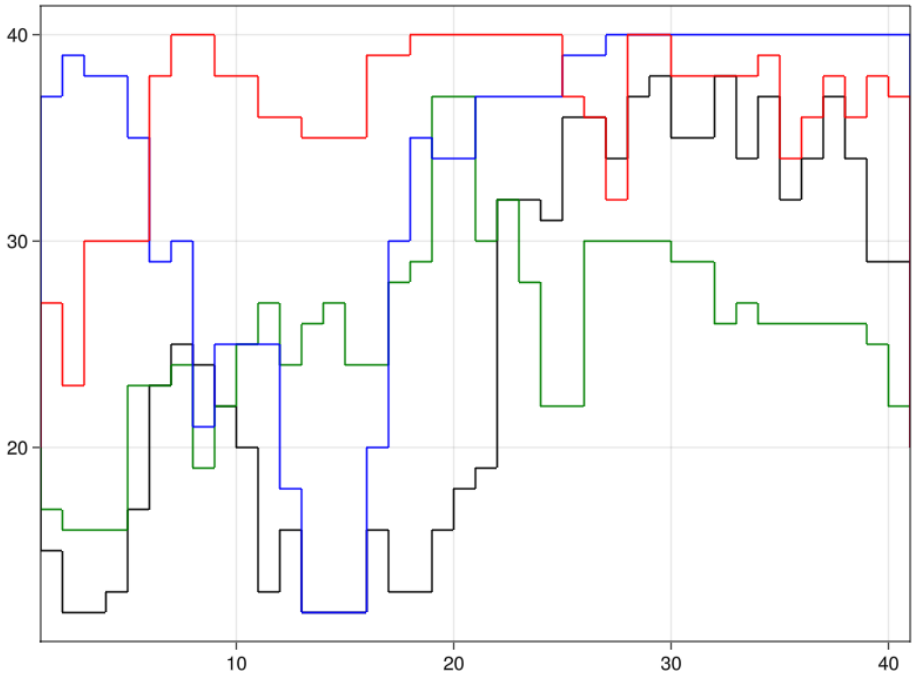
```
if success(rc9_4s)
    post9_4s = read_samples(m9_4s, :dataframe)
end
```

	parameters	median	mad_sd	mean	std
1	"alpha"	-7.828	101.512	-6.773	104.782
2	"beta"	7.787	101.535	6.777	104.781
3	"sigma"	1.092	0.08	1.096	0.078

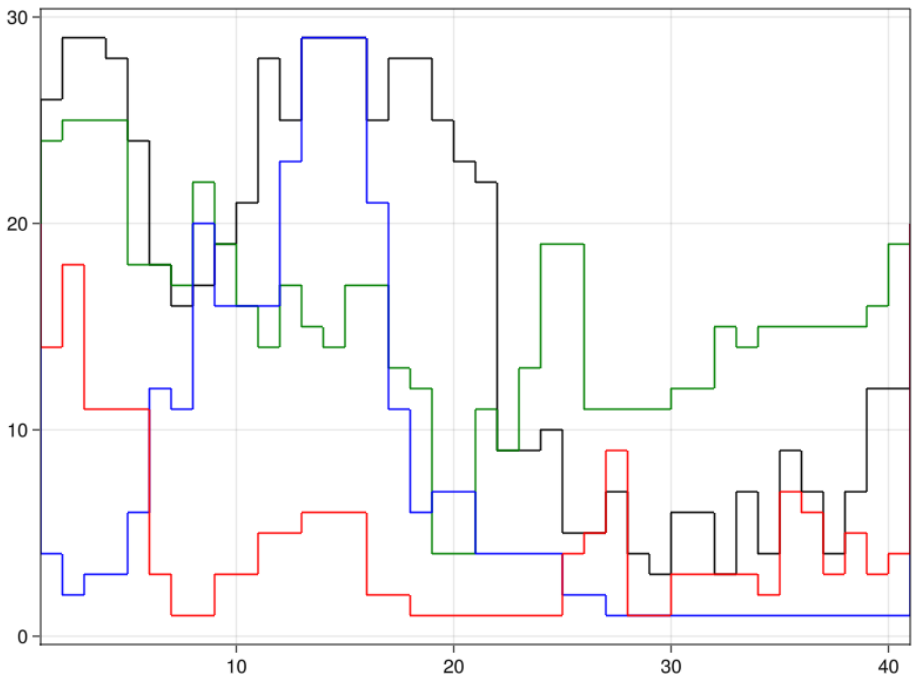
```
model_summary(post9_4s, [:alpha, :beta, :sigma])
```




```
plot_chains(post9_4s, [:alpha, :beta, :sigma])
```



```
trankplot(post9_4s, "alpha")
```




```
trankplot(post9_4s, "beta")
```

```
stan9_5 = "
data {
    int n;
    vector[n] y;
}
parameters {
    real alpha;
    real beta;
    real<lower=0> sigma;
}
model {
    real mu;
    alpha ~ normal(0, 10);
    beta ~ normal(0, 10);
    sigma ~ exponential(1);
    mu = alpha + beta;
    y ~ normal(mu, sigma);
}";
```

	parameters	mean	mcse	std	5%
1	"alpha"	0.0015794	0.254173	7.23111	-11.
2	"beta"	0.00349905	0.254632	7.23157	-11.
3	"sigma"	1.09401	0.0020217	0.0809901	0.96

```
begin
    # Re-use data from m9_4s
    m9_5s = SampleModel("m9.5s", stan9_5)
    rc9_5s = stan_sample(m9_5s; data=data9_4s)
    success(rc9_5s) && describe(m9_5s)
end
```

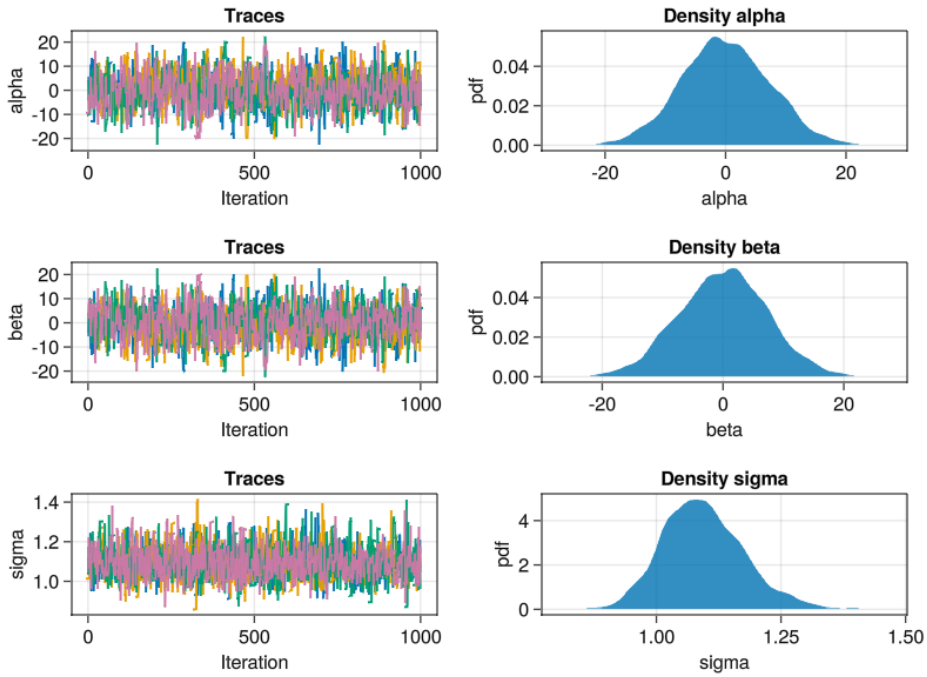
```
Informational Message: The current Metropolis propos
be rejected because of the following issue:
Exception: normal_lpdf: Scale parameter is 0, but mu
in '/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
stan', line 16, column 4 to column 26)
If this warning occurs sporadically, such as for hig
ariable types like covariance matrices, then the sam
but if this warning occurs often then your model may
ly ill-conditioned or misspecified.

Informational Message: The current Metropolis propos
rejected because of the following issue:
Exception: normal_lpdf: Scale parameter is 0, but mu
in '/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
stan', line 16, column 4 to column 26)
If this warning occurs sporadically, such as for hig
ariable types like covariance matrices, then the sam
but if this warning occurs often then your model may
ly ill-conditioned or misspecified.
```

	alpha	beta	sigma
1	5.56982	-5.40305	1.0352
2	-3.56507	3.55266	1.17554
3	-8.14825	8.10198	1.11049
4	-8.08085	8.15674	1.1103
5	-6.09395	6.0804	1.09802
6	-0.718372	0.651115	1.15697
7	6.00263	-6.09916	1.24334
8	10.2419	-10.3083	1.20502
9	10.6701	-10.7259	1.18993
10	5.47284	-5.41954	0.948773
⋮	more		
4000	6.57878	-6.69215	1.13901

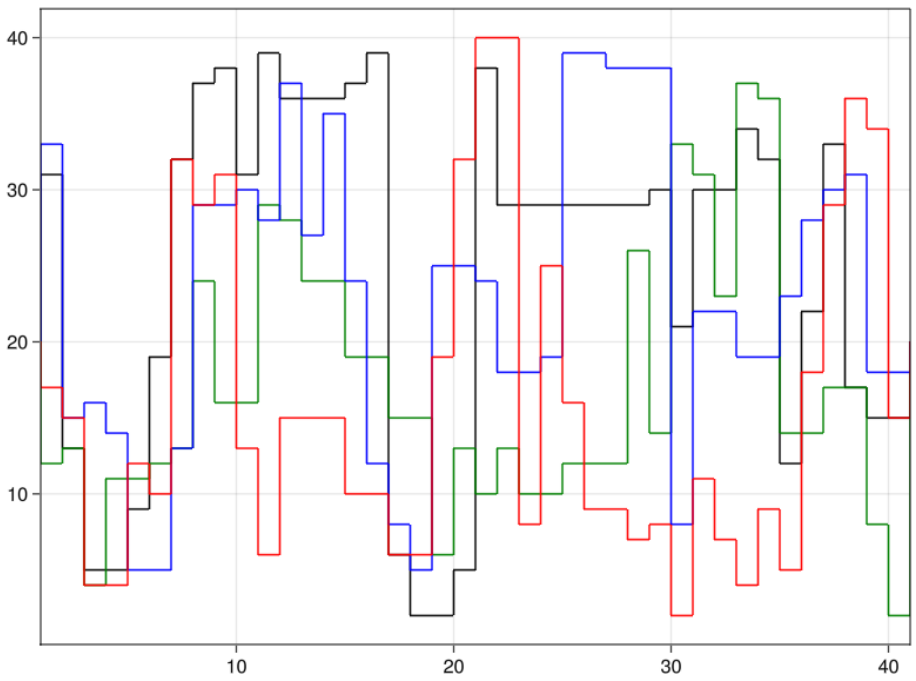
```
if success(rc9_5s)
    sdf9_5s = read_summary(m9_5s)
    post9_5s = read_samples(m9_5s, :dataframe)
end
```

	parameters	mean	mcse	std
1	:lp__	-60.0227	0.038637	1.23647
2	:accept_stat__	0.934843	0.00144222	0.0981003
3	:stepsize__	0.0110184	0.00037207	0.0005269
4	:treedepth__	6.35575	0.0355616	2.15682
5	:n_leapfrog__	204.535	2.63592	154.241
6	:divergent__	0.0	NaN	0.0
7	:energy__	61.5596	0.0530301	1.79131
8	:alpha	0.0015794	0.254173	7.23111
9	:beta	0.00349905	0.254632	7.23157
10	:sigma	1.09401	0.0020217	0.0809901

- `sdf9_5s`



- `plot_chains(post9_5s, [:alpha, :beta, :sigma])`




```
trankplot(post9_5s, "alpha")
```

	variable	mean	min	median	m
1	:rgdppc_2000	9094.89	466.647	5314.74	577
2	:log_gdp	8.51712	6.14557	8.57823	10.9
3	:log_gdp_s	1.0	0.721556	1.00718	1.28
4	:rugged	1.33318	0.003	0.9795	6.20
5	:rugged_s	0.21496	0.000483715	0.157933	1.0
6	:cid	1.71176	1	2.0	2

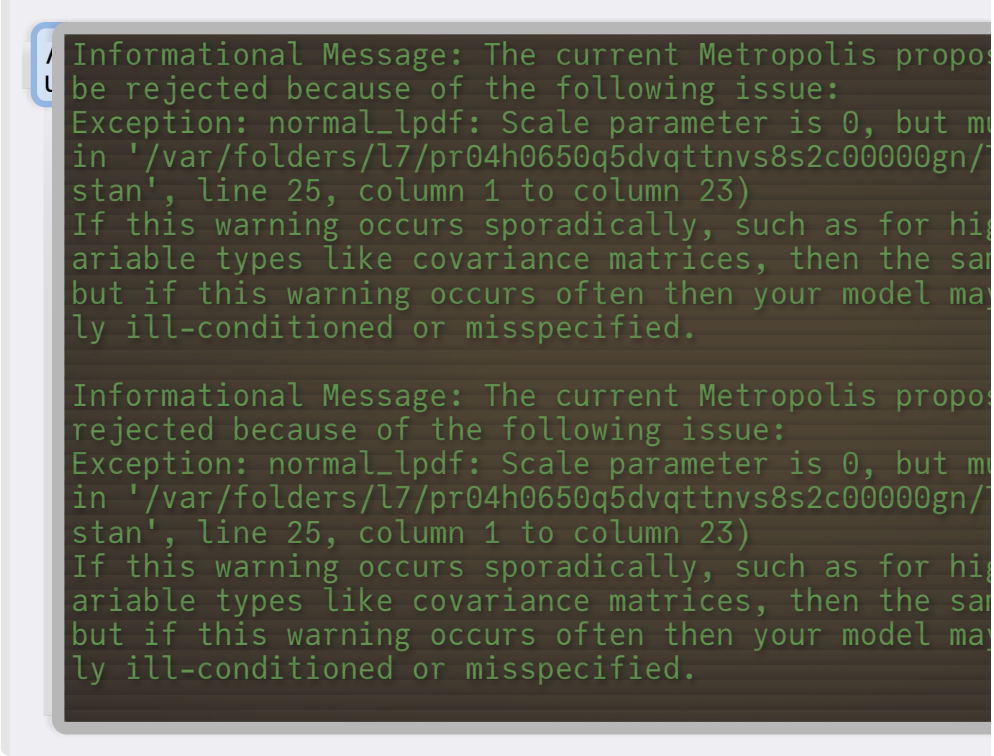
```
begin
    df = CSV.read(ros_datadir("SR2",
    "rugged.csv"), DataFrame)
    dropmissing!(df, :rgdppc_2000)
    dropmissing!(df, :rugged)
    df.log_gdp = log.(df[:, :rgdppc_2000])
    df.log_gdp_s = df.log_gdp / mean(df.log_gdp)
    df.rugged_s = df.rugged / maximum(df.rugged)
    df.cid = [df.cont_africa[i] == 1 ? 1 : 2 for i
    in 1:size(df, 1)]
    r̄ = mean(df.rugged_s)
    describe(df[:, [:rgdppc_2000, :log_gdp,
    :log_gdp_s, :rugged, :rugged_s, :cid]])
end
```

```
data8_3s = (N = size(df, 1), K =
length(unique(df.cid)), G = df.log_gdp_s, R =
df.rugged_s, cid=df.cid);
```

```
stan8_3 = "
data {
    int N;
    int K;
    vector[N] G;
    vector[N] R;
    int cid[N];
}

parameters {
    vector[K] a;
    vector[K] b;
    real<lower=0> sigma;
}

transformed parameters {
    vector[N] mu;
    for (i in 1:N)
        mu[i] = a[cid[i]] + b[cid[i]] * (R[i] -
        $(r̄));
}

model {
    a ~ normal(1, 0.1);
    b ~ normal(0, 0.3);
    sigma ~ exponential(1);
    G ~ normal(mu, sigma);
}
";
```

	parameters	mean	mcse	std	5%	
1	"a[1]"	0.89	0.00023	0.016	0.86	0
2	"a[2]"	1.1	0.00014	0.01	1.0	1
3	"b[1]"	0.13	0.0011	0.076	0.0038	0
4	"b[2]"	-0.14	0.00084	0.055	-0.23	-
5	"sigma"	0.11	0.0	0.01	0.1	0
6	"mu[1]"	0.88	0.00023	0.016	0.85	0
7	"mu[2]"	1.0	0.0003	0.02	0.97	1
8	"mu[3]"	1.1	0.00017	0.012	1.0	1
9	"mu[4]"	1.1	0.00017	0.012	1.0	1
10	"mu[5]"	1.0	0.00021	0.015	1.0	1
⋮	more					
175	"mu[170]"	0.88	0.00023	0.016	0.86	0

```
begin
    m8_3s = SampleModel("m8.3s", stan8_3)
    rc8_3s = stan_sample(m8_3s; data=data8_3s)
    success(rc8_3s) && describe(m8_3s)
end
```

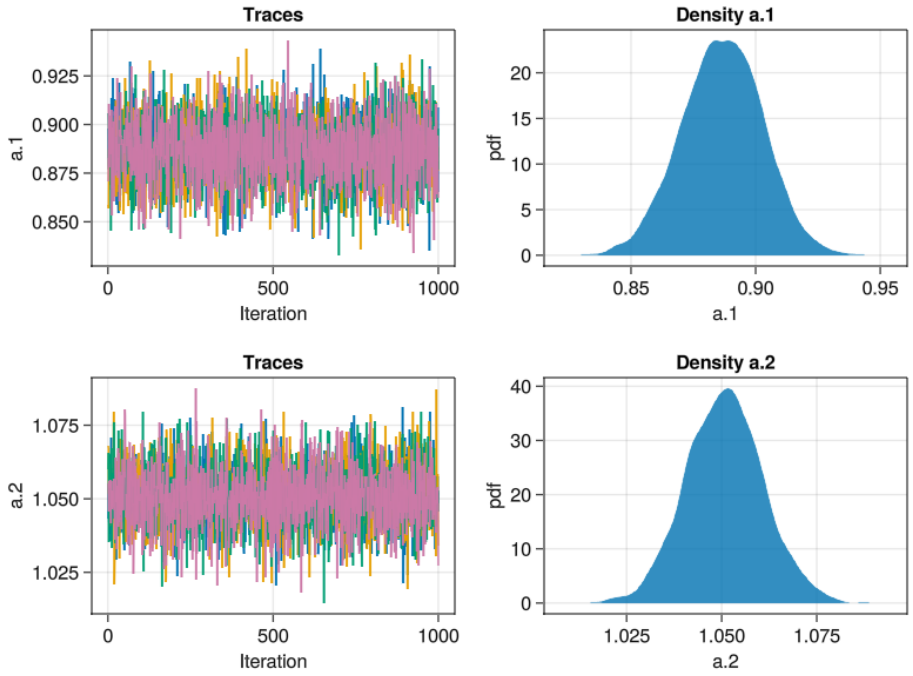

Informational Message: The current Metropolis propos
be rejected because of the following issue:
Exception: normal_lpdf: Scale parameter is 0, but mu
in '/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
stan', line 25, column 1 to column 23)
If this warning occurs sporadically, such as for hig
ariable types like covariance matrices, then the sam
but if this warning occurs often then your model may
ly ill-conditioned or misspecified.
Informational Message: The current Metropolis propos
rejected because of the following issue:
Exception: normal_lpdf: Scale parameter is 0, but mu
in '/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T
stan', line 25, column 1 to column 23)
If this warning occurs sporadically, such as for hig
ariable types like covariance matrices, then the sam
but if this warning occurs often then your model may
ly ill-conditioned or misspecified.

	sigma	a	b
1	0.115896	[0.898105, 1.05505]	[0.226078, -(
2	0.105665	[0.879329, 1.04336]	[0.0432967, ·
3	0.108974	[0.876619, 1.04594]	[0.0522773, ·
4	0.110399	[0.88959, 1.04211]	[0.156145, -(
5	0.110872	[0.884438, 1.06401]	[0.131963, -(
6	0.106499	[0.88545, 1.0381]	[0.125918, -(
7	0.112394	[0.898236, 1.06481]	[0.178581, -(
8	0.105976	[0.883061, 1.03549]	[0.108057, -(
9	0.111567	[0.869031, 1.06127]	[0.218732, -(
10	0.12295	[0.871215, 1.0709]	[0.205622, -(

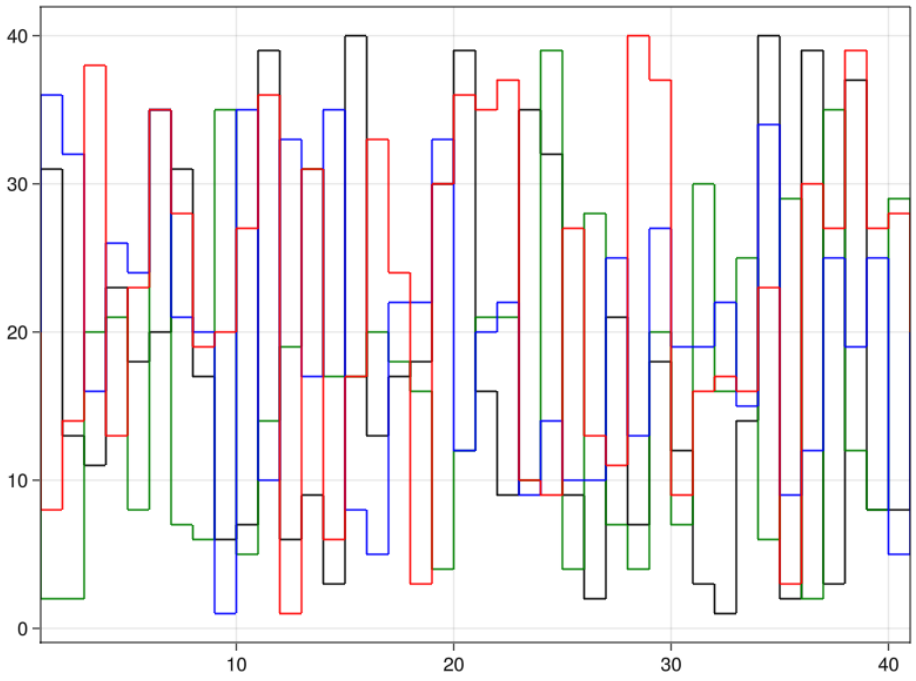
```
if success(rc8_3s)
    post8_3s = read_samples(m8_3s, :dataframe)
    nd8_3s = read_samples(m8_3s, :nesteddataframe)
end
```

```
4000×170 Matrix{Float64}:
 0.880784  1.00967   1.06728  1.06715  …  1.0336   0.9
 0.876012  0.98476   1.05915  1.05898     1.01566  0.8
 0.872613  0.977665  1.06433  1.06414     1.01366  0.8
 0.877626  1.01543   1.04929  1.04922     1.0295   0.9
 0.874327  1.04942   1.06794  1.0679      1.05711  0.8
 0.875803  0.963867  1.0581   1.05789  …  1.00301  0.8
 0.884553  1.03772   1.07211  1.07203     1.05201  0.9
 ⋮                                    ⋱
 0.858055  0.994668  1.06938  1.06921     1.0257   0.8
 0.876568  0.999425  1.06515  1.065    …  1.02672  0.8
 0.868499  1.00966   1.05866  1.05855     1.03001  0.8
 0.884145  0.994123  1.07014  1.06997     1.0257   0.9
 0.862615  1.00797   1.05886  1.05874     1.02911  0.8
 0.853886  0.978273  1.04031  1.04017     1.00404  0.8
```

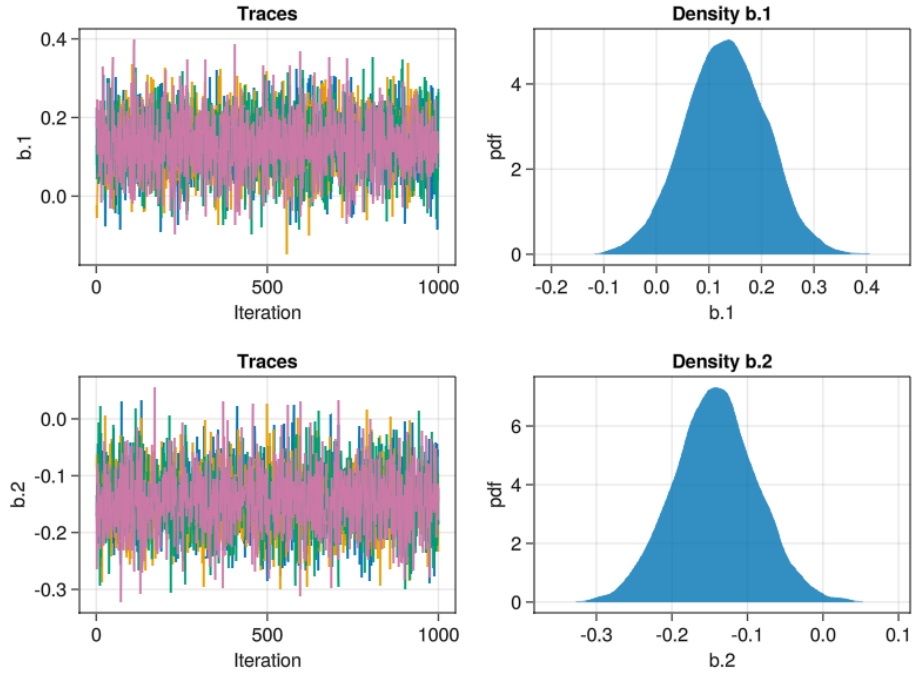
```
array(nd8_3s, :mu)
```



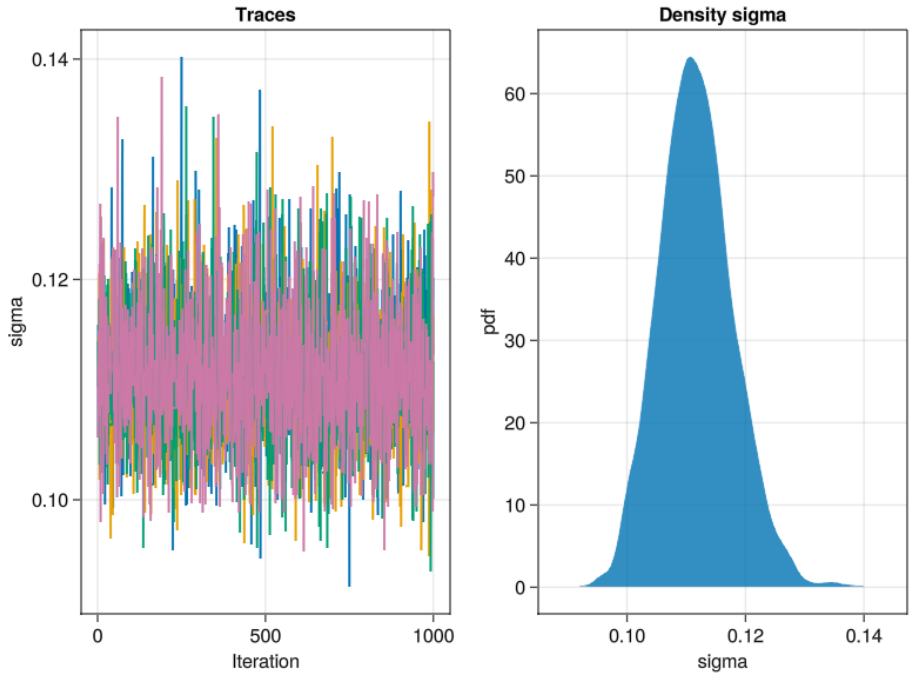

```
plot_chains(post8_3s, [Symbol("a.1"),
Symbol("a.2")])
```




```
trankplot(post8_3s, "a.1")
```




```
plot_chains(post8_3s, [Symbol("b.1"),
Symbol("b.2")])
```




```
plot_chains(post8_3s, [:sigma])
```