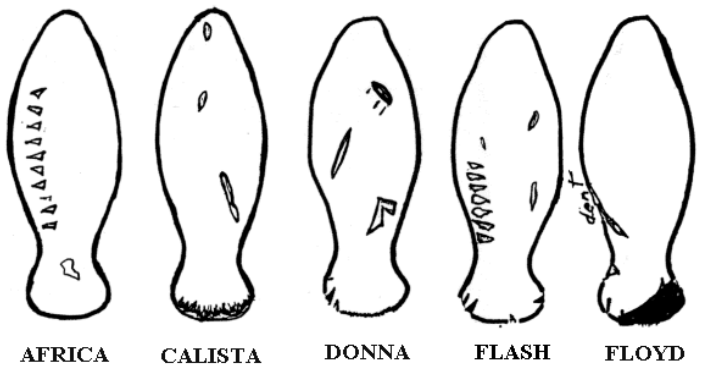
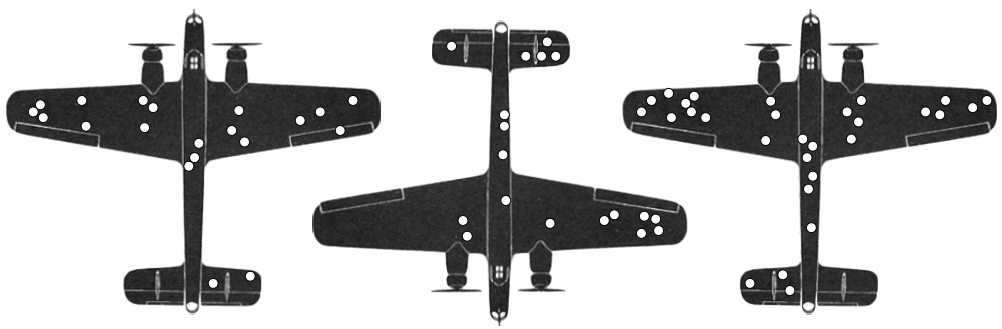
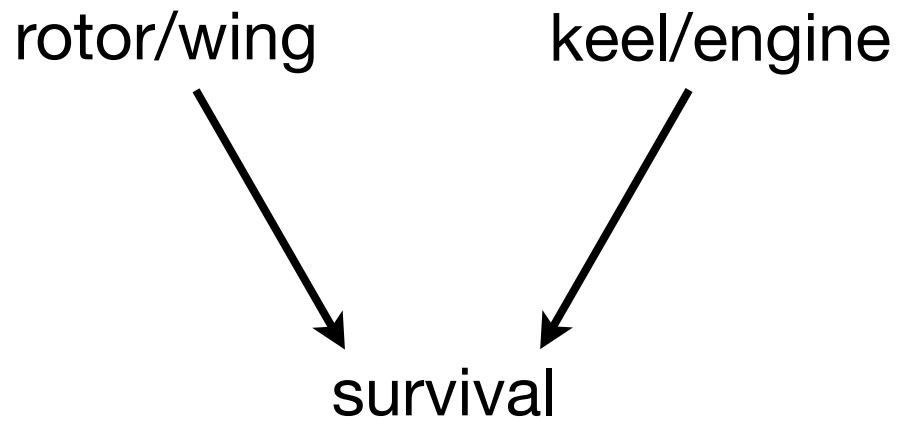






Observe only:
undamaged
rotor/wing damaged

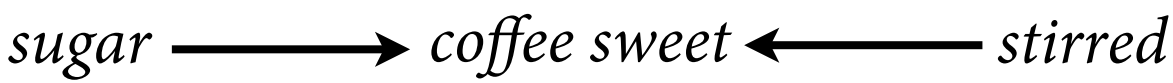
Figure 8.1

Interaction effects

- *Interactions*: Influence of predictor conditional on other predictor(s)
  - Influence of *sugar* in *coffee* depends on *stirring*
  - Influence of *gene* on *phenotype* depends on *environment*
  - Influence of *skin color* on *cancer* depends on *latitude*
- Generalized linear models (GLMs): All predictors interact to some degree
- Multilevel models: Massive interaction engines

Interaction effects in DAGs

- In DAG, interaction doesn't look special:



- This just means:

$$coffee\ sweet = f(sugar, stirred)$$

- We have to figure out the function f.

R code 8.13

```
m8.5 <- quap(
    alist(
        log_gdp_std ~ dnorm( mu , sigma ) ,
        mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
        a[cid] ~ dnorm( 1 , 0.1 ) ,
        b[cid] ~ dnorm( 0 , 0.3 ) ,
        sigma ~ dexp( 1 )
    ) ,
    data=dd )
```

R code 8.14

```
precis( m8.5 , depth=2 )
```

```
        mean   sd  5.5% 94.5%
a[1]    0.89 0.02  0.86  0.91
a[2]    1.05 0.01  1.03  1.07
b[1]    0.13 0.07  0.01  0.25
b[2]   -0.14 0.05 -0.23 -0.06
sigma   0.11 0.01  0.10  0.12
```

Interpreting interactions

- Is hard
  - Add interaction => other parameters change meaning
  - Influence of predictor depends upon multiple parameters and their covariation

Additionally, you need a sample that's 4x larger to measure the same size of an effect as for a beta parameter!

R code 8.14

```
precis( m8.5 , depth=2 )
```

```
        mean   sd  5.5% 94.5%
a[1]    0.89 0.02  0.86  0.91
a[2]    1.05 0.01  1.03  1.07
b[1]    0.13 0.07  0.01  0.25
b[2]   -0.14 0.05 -0.23 -0.06
sigma   0.11 0.01  0.10  0.12
```

Tulip model – interaction

R code 8.24

```
m8.7 <- quap(
    alist(
        blooms_std ~ dnorm( mu , sigma ) ,
        mu <- a + bw*water_cent + bs*shade_cent + bws*water_cent*shade_cent ,
        a ~ dnorm( 0.5 , 0.25 ) ,
        bw ~ dnorm( 0 , 0.25 ) ,
        bs ~ dnorm( 0 , 0.25 ) ,
        bws ~ dnorm( 0 , 0.25 ) ,
        sigma ~ dexp( 1 )
    ) ,
    data=d )
```

Interpreting parameters very hard! Plot.

Posterior predictions

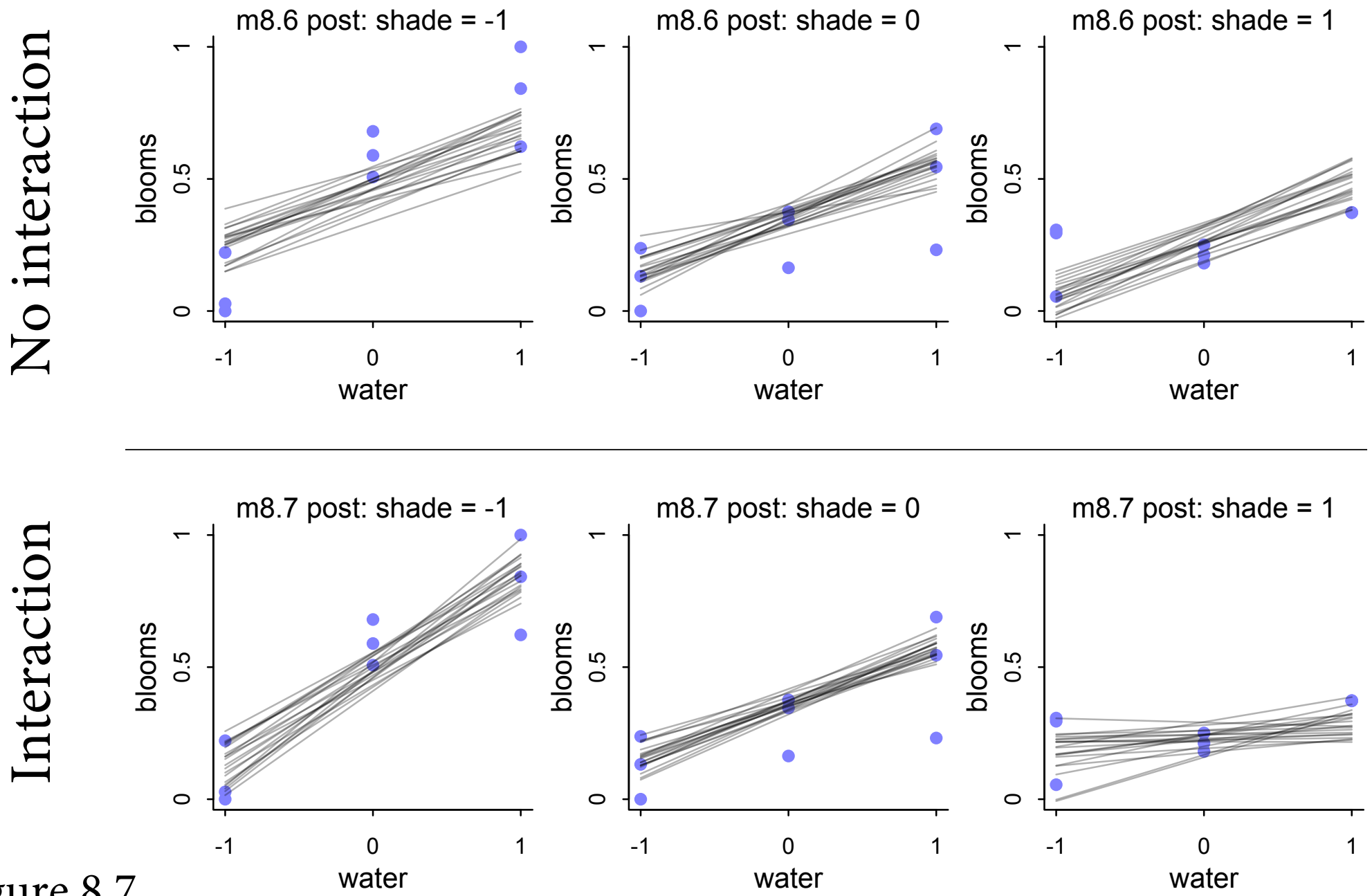
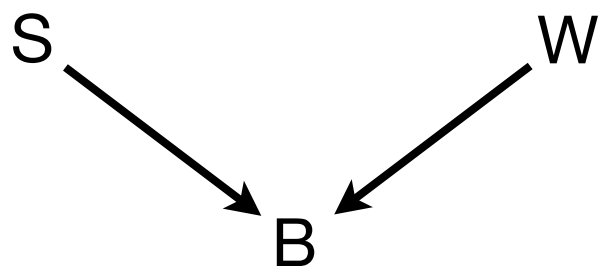


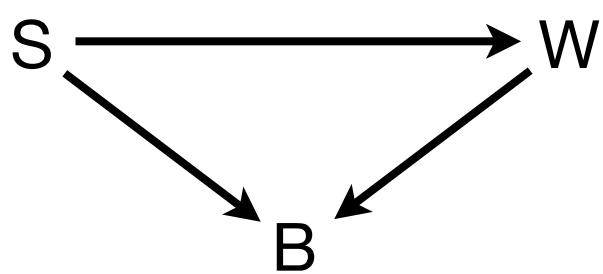
Figure 8.7

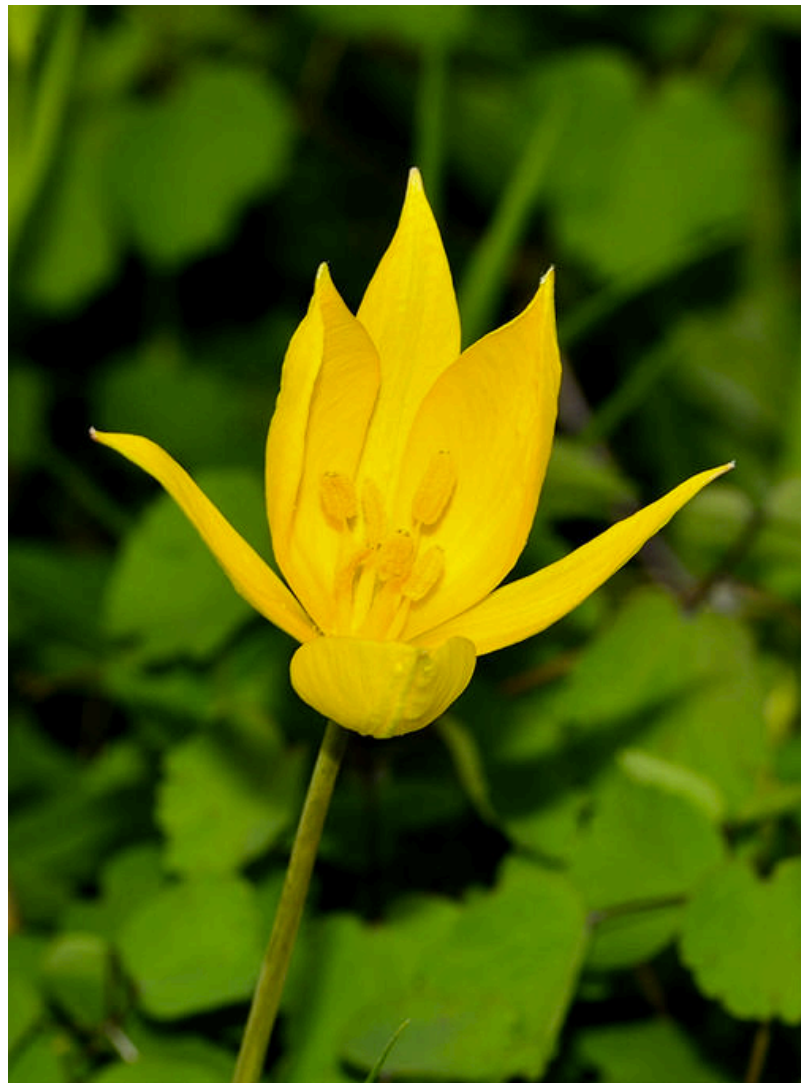
Causal thinking

- Tulip experiment:



- Tulip reality:



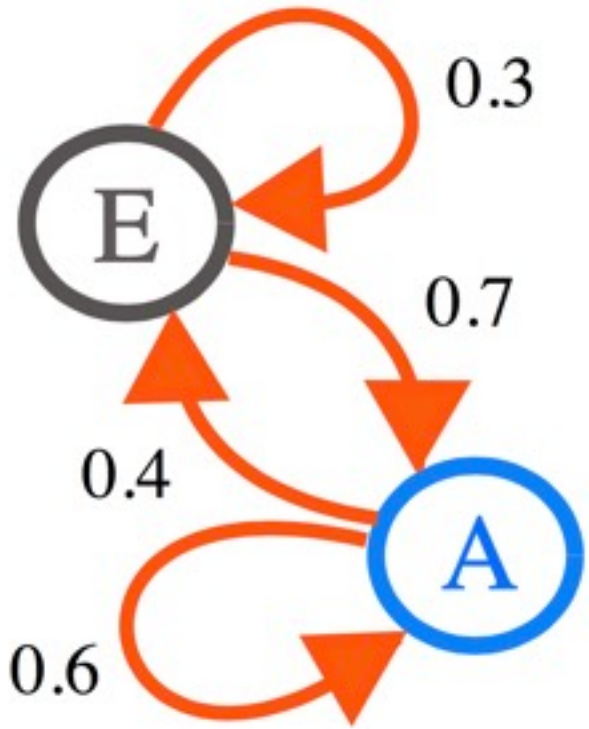


Interactions not always linear

- Suppose all tulip data collected under “cool” temperatures
- Under “hot” temperature, tulips do not bloom
- Interaction, but not a linear one
  - blooms goes to zero at threshold

Markov chain Monte Carlo

- Markov chain Monte Carlo (MCMC)
  - Understand the approach
  - Meet different algorithms
- Interfaces to MCMC: Stan & `ulam`
- How to sample responsibly
- How to recognize and fix problems



Why MCMC?

- Sometimes can't write an integrated posterior
- Even when can, often cannot use it
- Many problems are like this: Multilevel models, networks, phylogenies, spatial models
- Optimization not a good strategy in high dimensions — must have full distribution
- **MCMC is not fancy. It is old and essential.**

Hamiltonian Monte Carlo

- Problem with Gibbs sampling (GS)
  - High dimension spaces are *concentrated*
  - GS gets stuck, degenerates towards random walk
  - Inefficient because re-explores
- Hamiltonian dynamics to the rescue
  - represent parameter state as particle
  - flick it around frictionless log-posterior
  - record positions
  - no more "guess and check"
  - all proposals are good proposals

William Rowan Hamilton (1805–1865) Commemorated on Irish Euro coin

https://chi-feng.github.io/mcmc-demo/

Hamiltonian Flows

- What happens when you use ulam?
  - Translates formula into raw Stan model code
  - Stan then builds a custom NUTS sampler
  - Sampler runs
  - Samples fed back to R

```
m9.1 <- ulam(
    alist(
        log_gdp_std ~ dnorm( mu , sigma ) ,
        mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
        a[cid] ~ dnorm( 1 , 0.1 ) ,
        b[cid] ~ dnorm( 0 , 0.3 ) ,
        sigma ~ dexp( 1 )
    ) ,
    data=dat_slim , chains=4 , cores=4 , iter=1000 )
```

R code 9.14

Hamiltonian Flows

R code 9.16

```
precis( m9.1 , 2 )
```

```
       mean   sd  5.5% 94.5% n_eff Rhat
sigma  0.11 0.01  0.10  0.12  2641    1
b[1]   0.13 0.07  0.02  0.26  2484    1
b[2]  -0.14 0.05 -0.23 -0.06  2546    1
a[1]   0.89 0.02  0.86  0.91  3331    1
a[2]   1.05 0.01  1.03  1.07  3243    1
```

- n_eff: number of effective samples
  - can be larger than actual samples!
- Rhat: Convergence diagnostic — “1” is good

```
pairs( m9.1 )
```

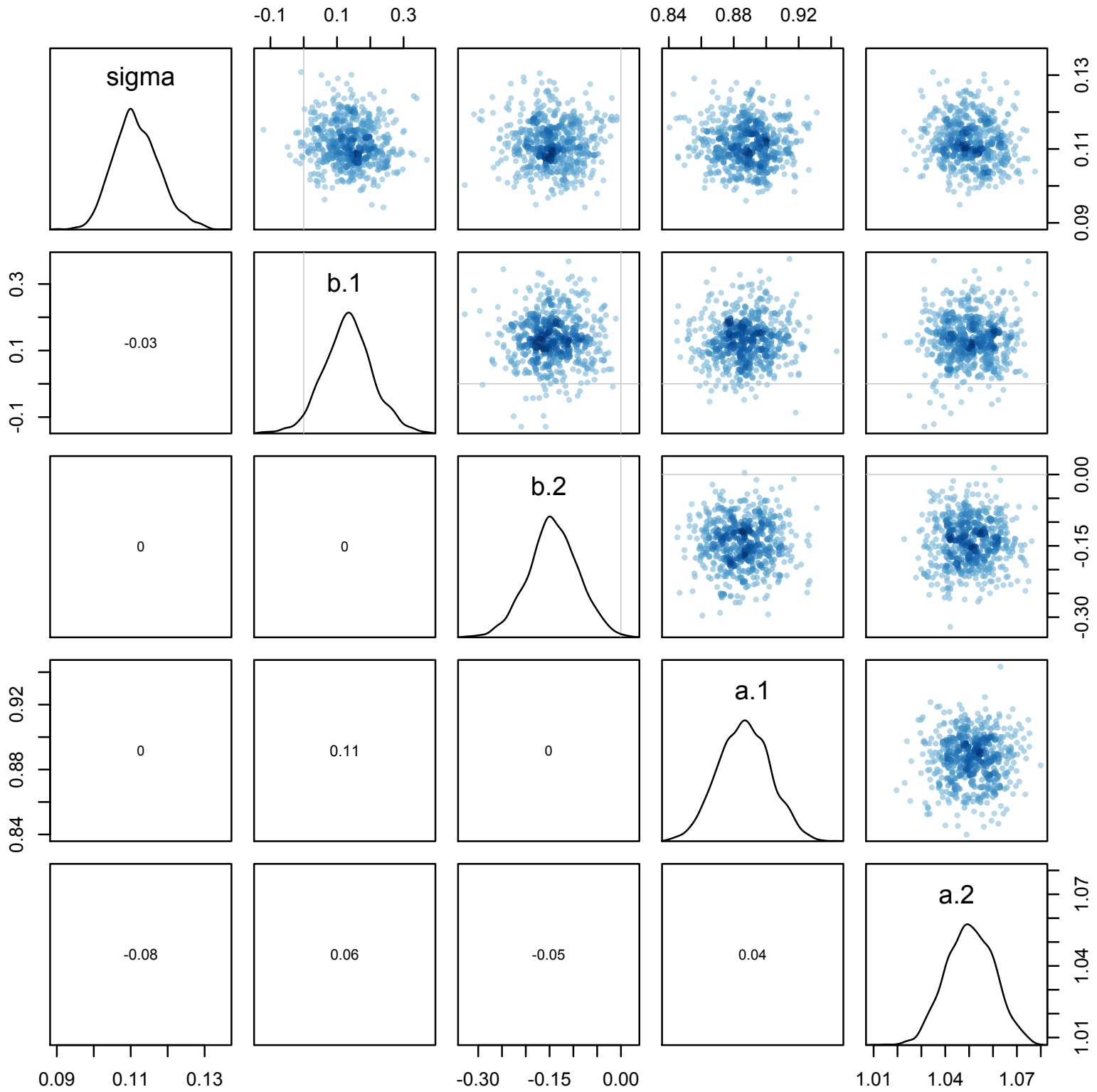
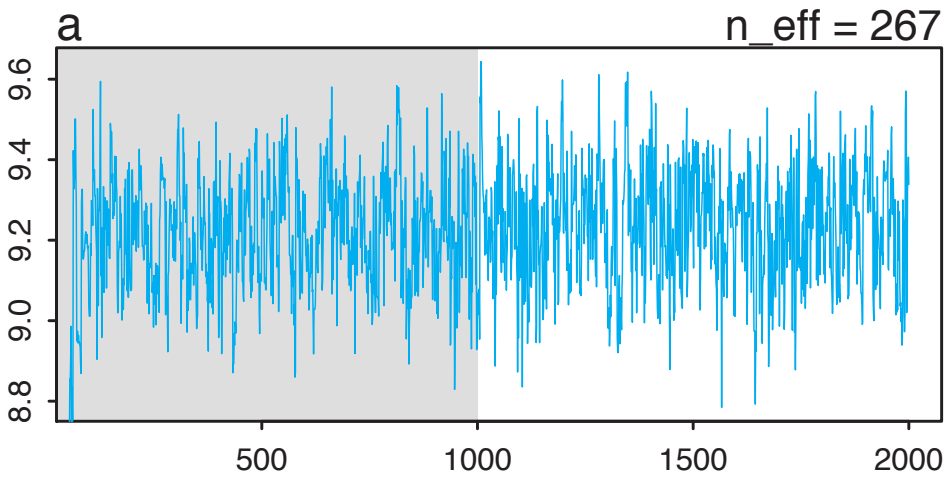
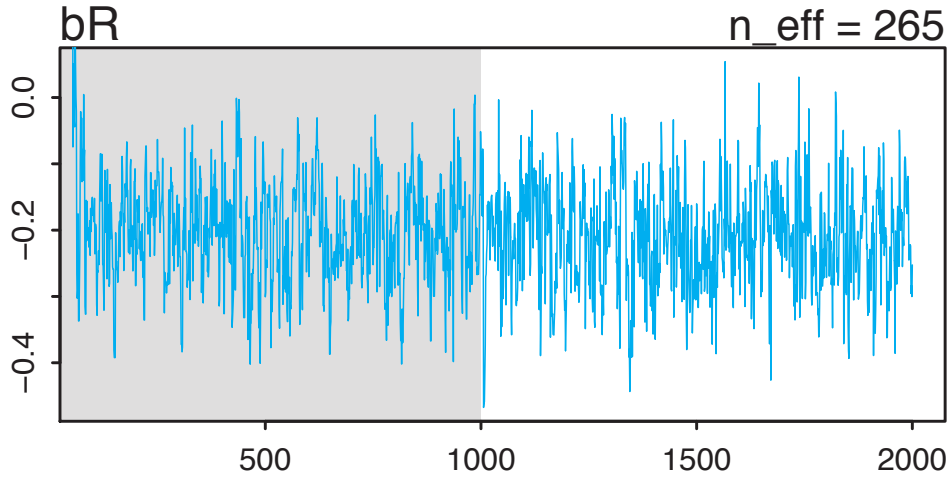
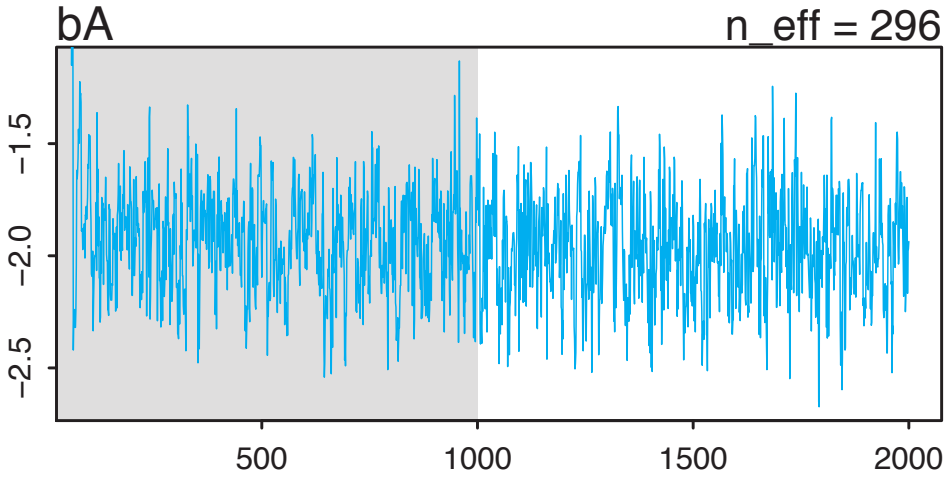
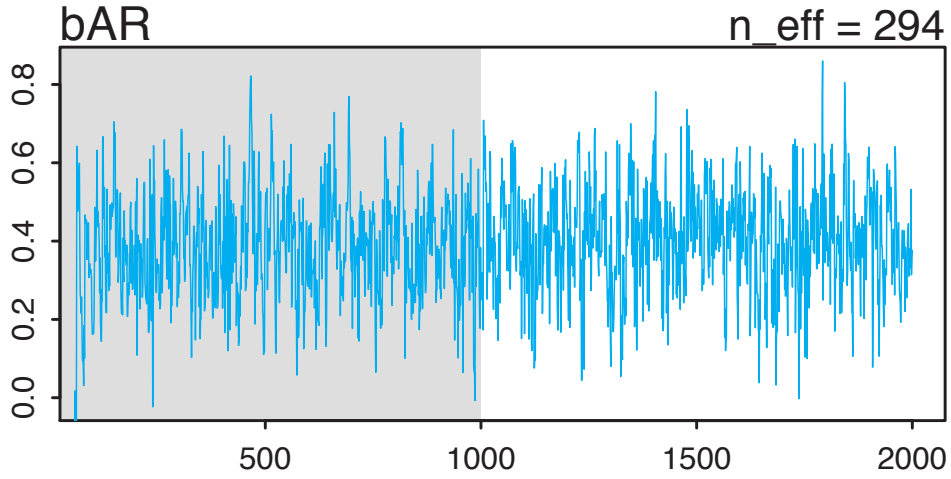


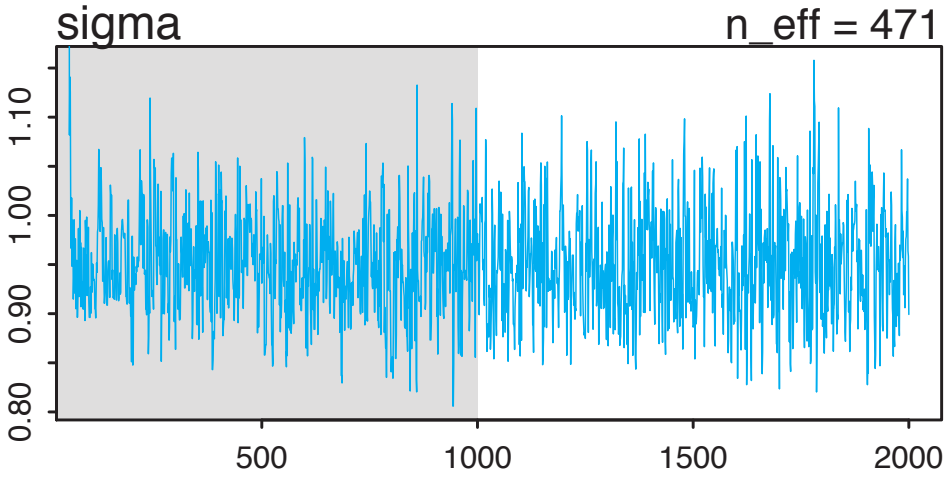
Figure 9.7











"Hairy caterpillar ocular inspection test"

Convergence diagnostics

- n_eff: "effective" number of samples
  - n_eff/n < 0.1, be alarmed
- R-hat
  - R-hat: crudely, ratio of variance between chains to variance within chains
  - Should approach 1
- Both may mislead

R code 9.16

```
precis( m9.1 , 2 )
```

```
       mean   sd  5.5% 94.5% n_eff Rhat
sigma  0.11 0.01  0.10  0.12  2641    1
b[1]   0.13 0.07  0.02  0.26  2484    1
b[2]  -0.14 0.05 -0.23 -0.06  2546    1
a[1]   0.89 0.02  0.86  0.91  3331    1
a[2]   1.05 0.01  1.03  1.07  3243    1
```

The Folk Theorem of Statistical Computing

"When you have computational problems, often there's a problem with your model."

–Andrew Gelman