Day 6: Multiple Regression

Stephen R. Proulx

Today's objectives:

- Use quap to perform multiple regression

- Develop an understanding of how including parameters can alter our inference
- Practice producing plots from posterior samples.

Our general routine will be: 1. Write a likelihood model for the data and choose priors
2. Make sure the priors are sensible. Start with broad or flat priors, and then apply knowledge to shrink them.
3. Use a method to approximate the posterior probability density of the parameters given the data.
4. Get a sample of parameters from the posterior (for MCMC methods this happens in step 3).
5. Analyze the posterior distribution.
6. Conduct posterior predictive simulation to sanity check our work and to make predictions for future work.

Note on chapter 5: McElreath spends a lot of time developing the DAG concept and discussing how confounds arise when the structure of the causal model is not accounted for. We are largely going to bypass this. It's useful to read it and start thinking about it, but you won't be tested on this and we won't directly use it in the rest of our analysis.

Spurius waffles

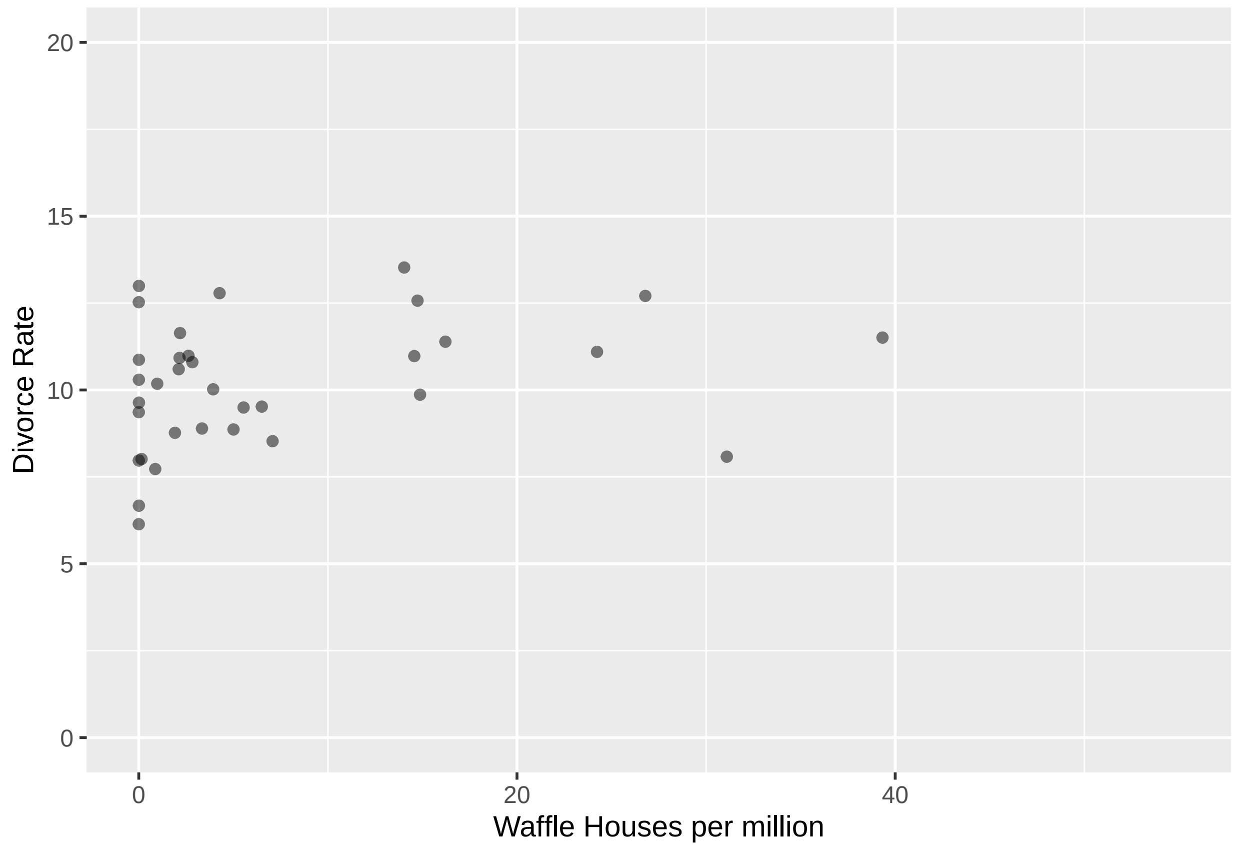
Load the data on divorce rates by state.

```
data(WaffleDivorce)
d <- WaffleDivorce
```

This is the correlation between waffle houses and divorce rate. It isn't super informative, but is none the less a statistically detectable effect.

```
  ggplot(d, aes(x = WaffleHouses/Population, y = Divorce))+
  geom_jitter(alpha=0.5)+
  scale_x_continuous("Waffle Houses per million", limits = c(0, 55)) +
  scale_y_continuous("Divorce Rate", limits = c(0, 20))
```

```
## Warning: Removed 16 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

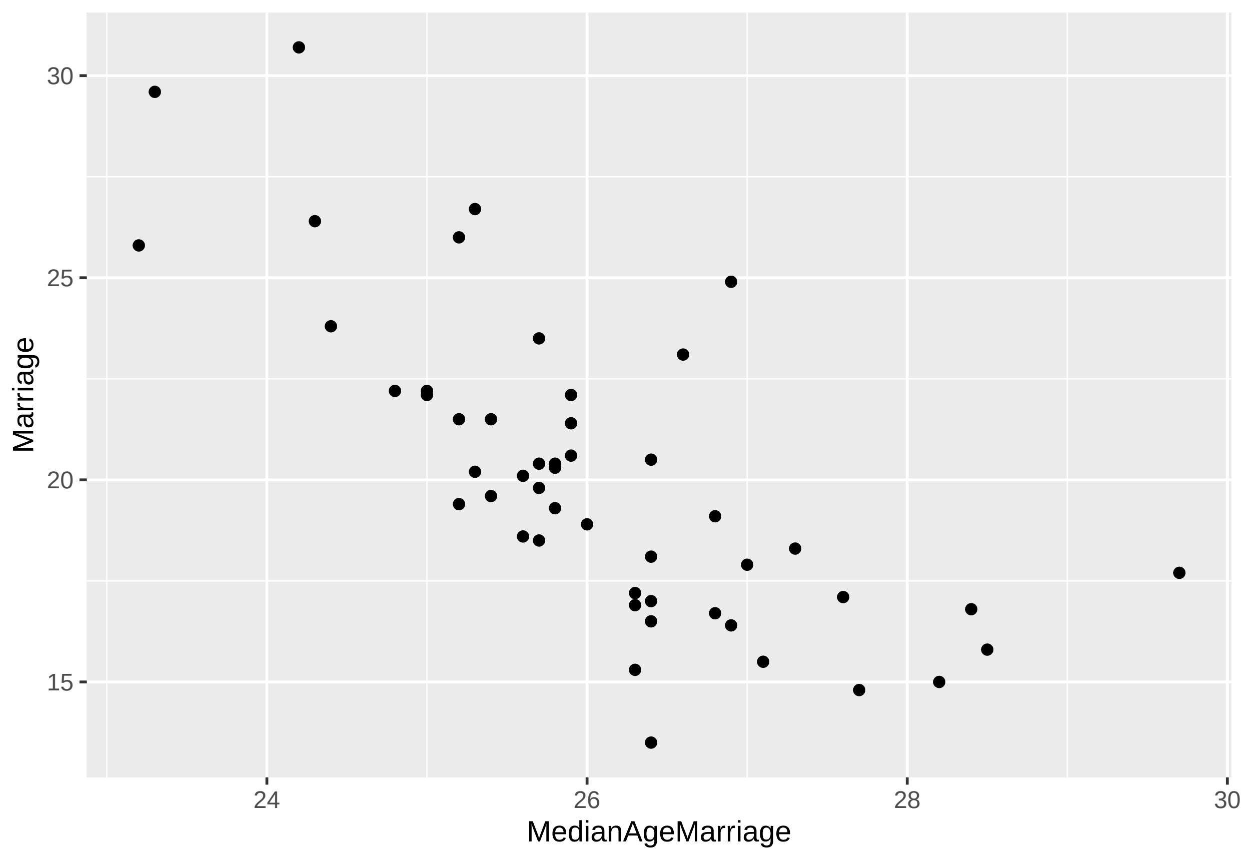


Prepare the data

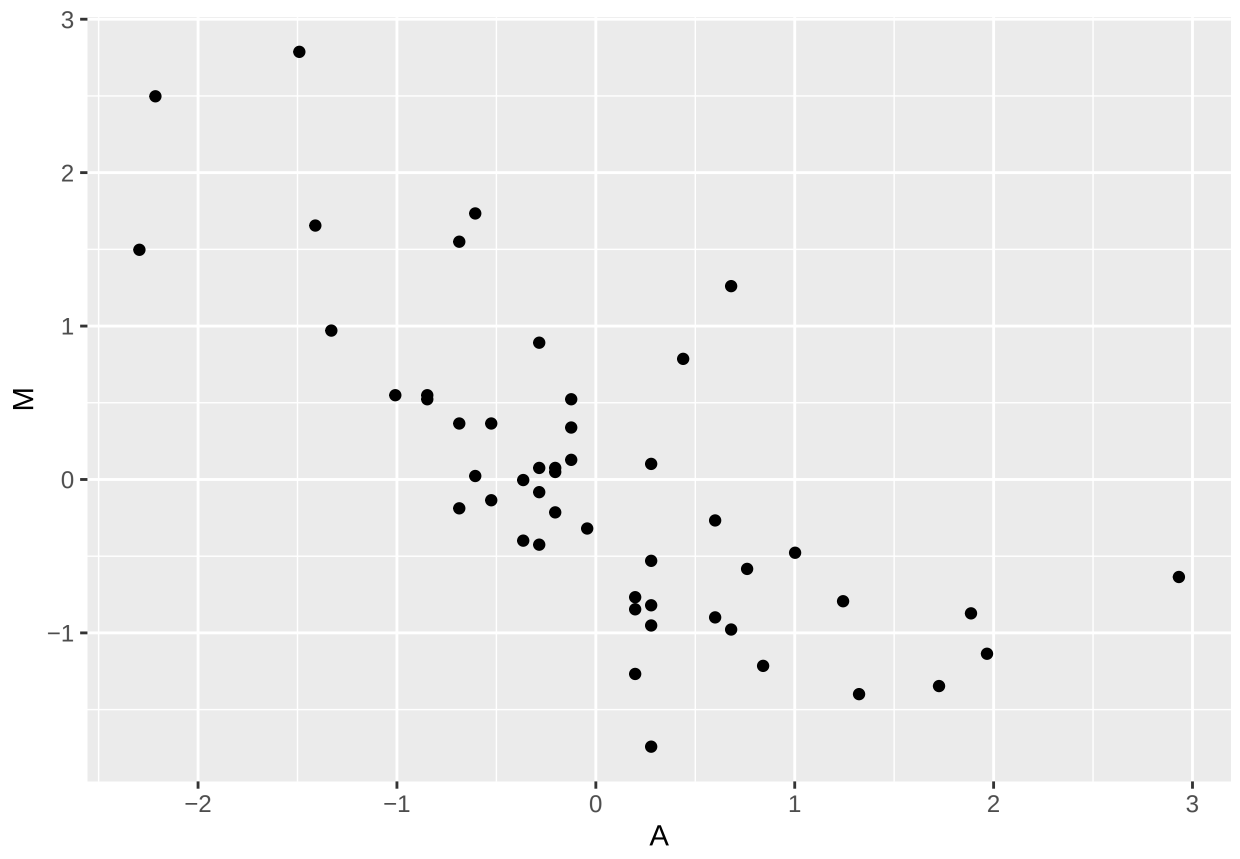
We'll work with standardized transformations of the data. The function `standardize` does the work for us, and we'll use new columns that are single letter names, as in the book.

```r
d <-
  d %>%
  mutate(A = standardize(MedianAgeMarriage),
         D = standardize(Divorce),
         M = standardize(Marriage))
```

```r
ggplot(data=d, aes(x=MedianAgeMarriage,y=Marriage)) + geom_point()
```



```r
ggplot(data=d, aes(x=A,y=M)) + geom_point()
```



First model, median age matters

Here's the description of the model

$$D_i \sim \text{Normal}(\mu_i, \sigma)\mu_i = a + bA * A_i a \sim \text{Normal}(0, 0.2) bA \sim \text{Normal}(0, 0.5)\sigma \sim \text{Exponential}(1)$$

Write the quap model for this- you need to fill in the "alist" part:

```
M5.1 <- quap( alist(
  D ~ dnorm(mu,sigma),
  mu <- a+  bA * A,
  a ~ dnorm(0,0.2),
  bA ~ dnorm(0,0.5),
  sigma ~ dexp(1)),
  data=d)
```

Have a look at the means and likely intervals of the parameters. `precis` is a good method for this.

```
precis(M5.1)
```

```
##                mean         sd       5.5%      94.5%
## a      4.321436e-08 0.09737874 -0.1556300  0.1556301
## bA    -5.684034e-01 0.10999977 -0.7442042 -0.3926025
## sigma  7.883254e-01 0.07801127  0.6636483  0.9130025
```

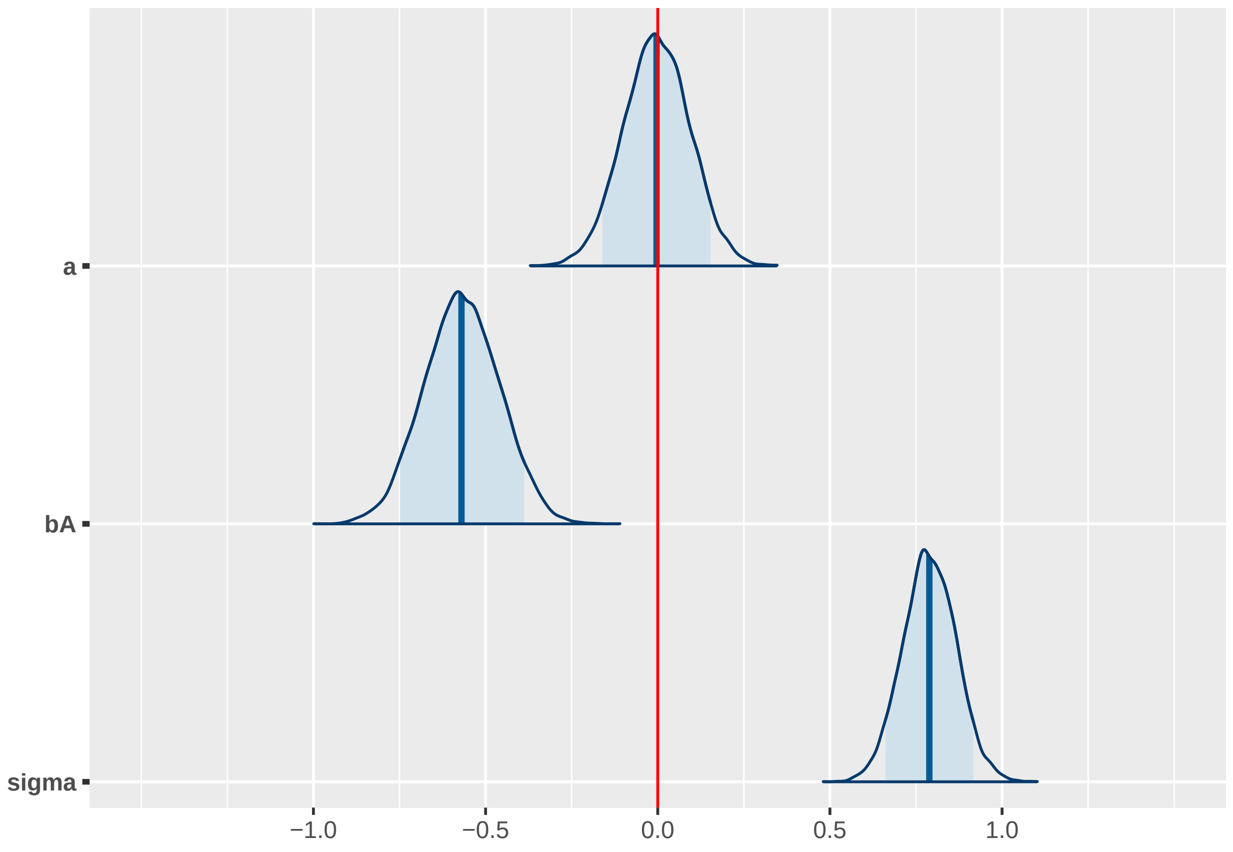
Extract the samples:

```r
samples_M5.1 <- extract.samples(M5.1)
```

Use this `mcmc_areas` to get the parameter distributions plotted in a way that we can visualize their limits.

```r
bayesplot::mcmc_areas(samples_M5.1,prob=0.9 , area_method="equal height") +
  geom_vline(xintercept = 0 , color="red")+
  scale_x_continuous(limits=c(-1.5,1.5),
                     breaks = c(-1,-0.5,0,0.5,1))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



```r
test_d <- tibble(A=seq(from=-2,to=2,by=0.1))
```

And run this back through `link_df`

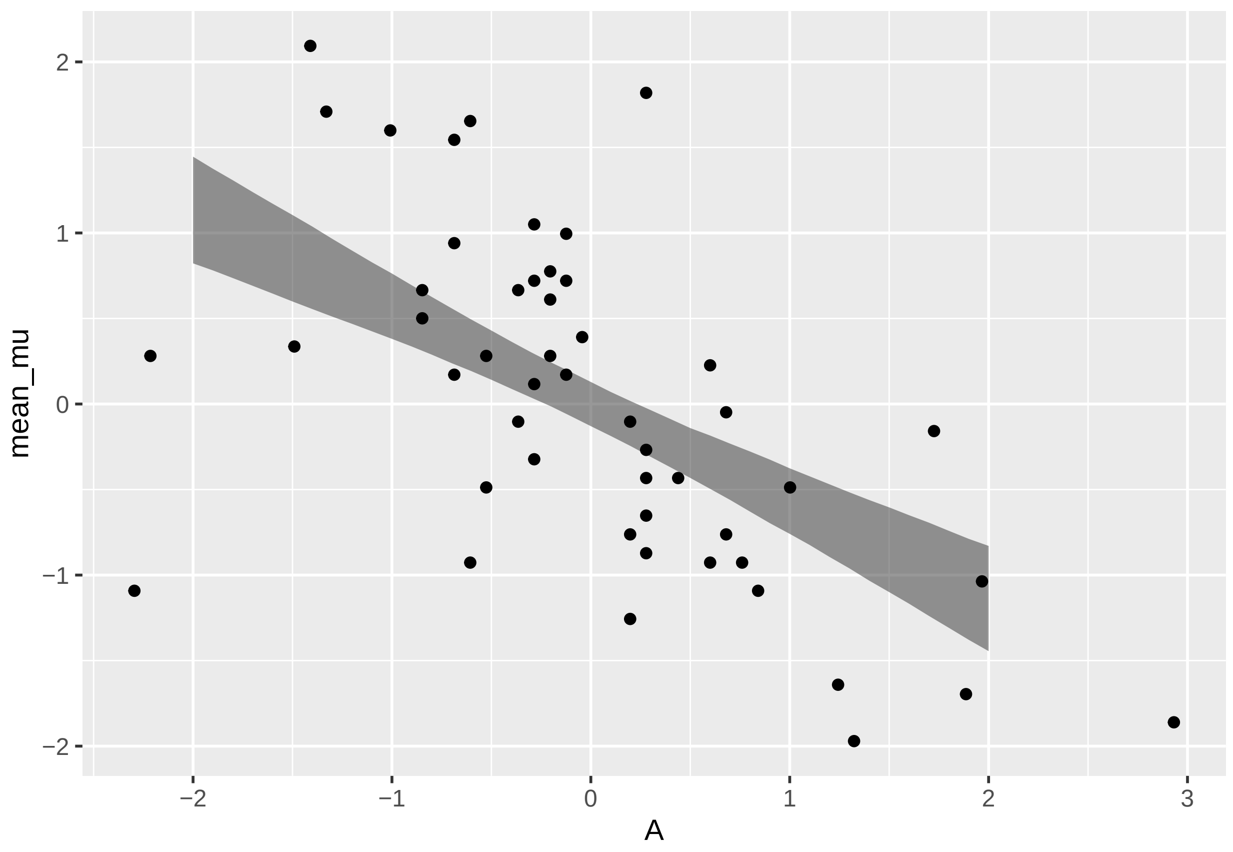
```r
samples_Aonly <- link_df(M5.1,test_d)
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
## `.name_repair` is omitted as of tibble 2.0.0.
## i Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use ‘all_of()‘ or ‘any_of()‘ instead.
##   # Was:
##   data %>% select(i)
##
##   # Now:
##   data %>% select(all_of(i))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
summarised_samples_Aonly<-group_by(samples_Aonly,A) %>%
  summarise(mean_mu=mean(mu),
            lower_mu=quantile(mu,0.1),
            upper_mu=quantile(mu,0.9))%>%
  ungroup()

ggplot(summarised_samples_Aonly,aes(x = A, y =mean_mu)) +
  geom_ribbon(aes(ymin=lower_mu,ymax=upper_mu),alpha=0.5)+
  geom_point(data=d,aes(x=A,y=D))
```



Second model, marriage rate matters

Here's the description of the model

$$D_i \sim \text{Normal}(\mu_i, \sigma)\mu_i = a + bM * M_i a \sim \text{Normal}(0, 0.2)bM \sim \text{Normal}(0, 0.5)\sigma \sim \text{Exponential}(1)$$

Write the quap model for this- you need to fill in the `alist` part:

```
M5.2 <- quap( alist(
  D ~ dnorm(mu,sigma),
  mu <- a+  bM * M,
  a ~ dnorm(0,0.2),
  bM ~ dnorm(0,1),
  sigma ~ dexp(1)),
  data=d)
```

Have a look at the means and likely intervals of the parameters. `precis` is a good method for this.

```
precis(M5.2)
```

```
##                  mean         sd       5.5%     94.5%
## a     -4.588311e-08 0.10822353 -0.1729621 0.1729621
## bM     3.675204e-01 0.12891991  0.1614815 0.5735593
## sigma  9.099932e-01 0.08978569  0.7664983 1.0534880
```

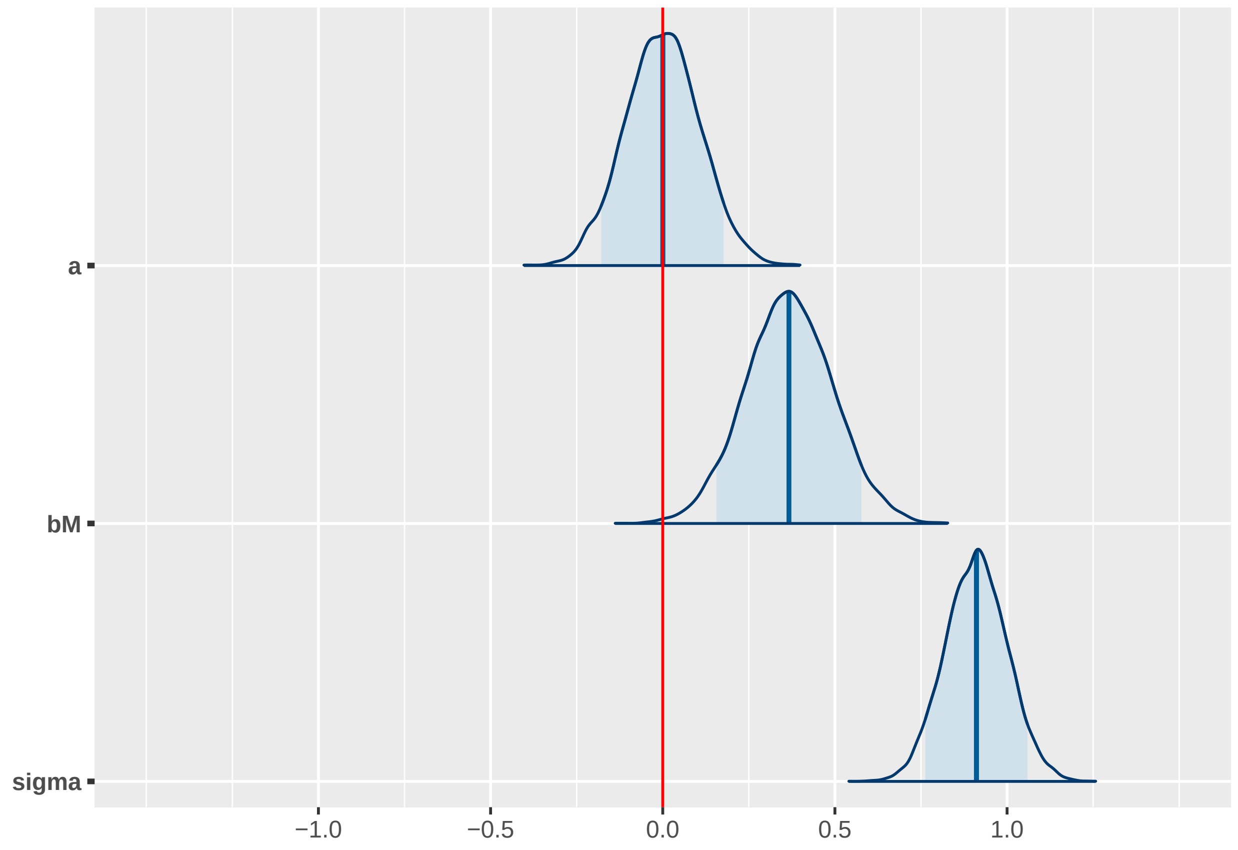
Extract the samples

```
samples_M5.2 <- extract.samples(M5.2)
```

Use this `mcmc_areas` to get the parameter distributions plotted in a way that we can visualize their limits.

```
bayesplot::mcmc_areas(samples_M5.2,prob=0.9 , area_method="equal height") +
  geom_vline(xintercept = 0 , color="red")+
  scale_x_continuous(limits=c(-1.5,1.5),
                     breaks = c(-1,-0.5,0,0.5,1))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```

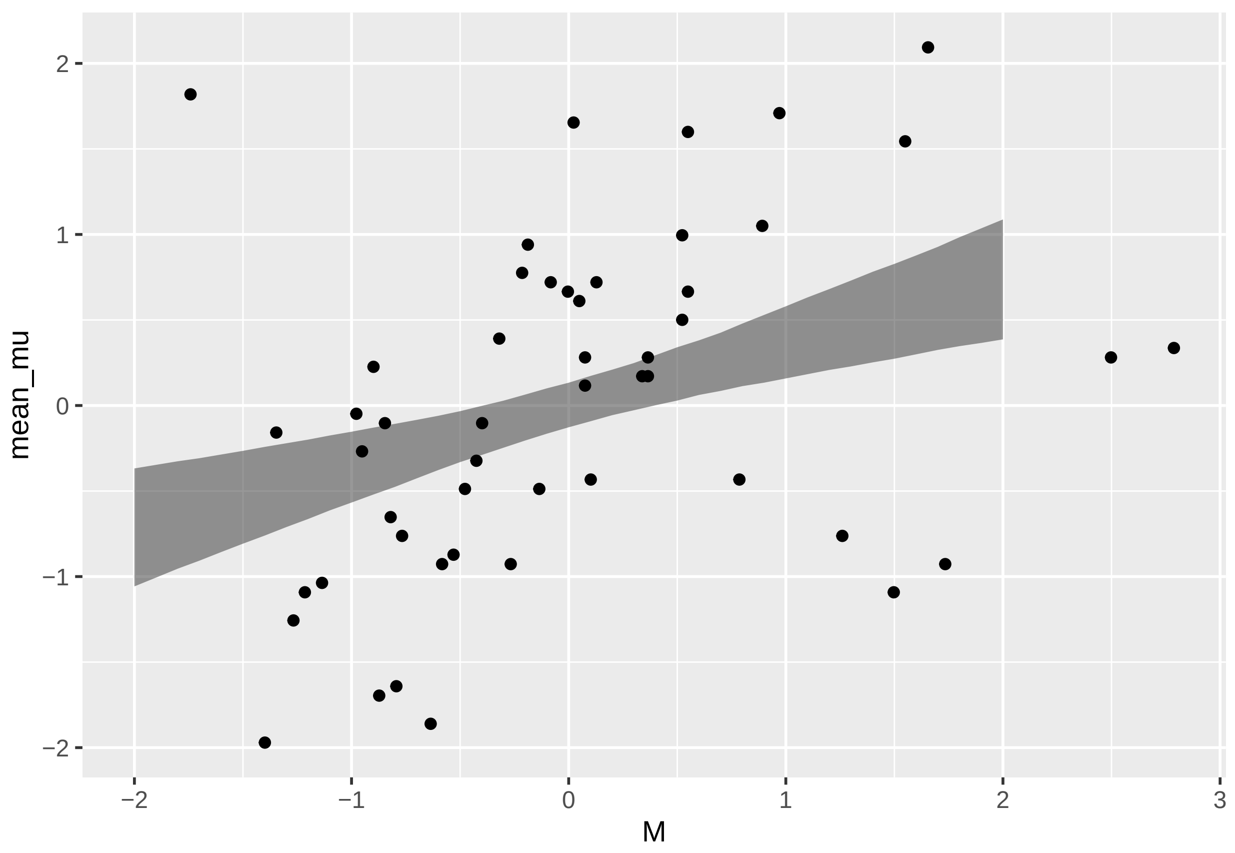


```
test_d <- tibble(M=seq(from=-2,to=2,by=0.1))
```

And run this back through `link_df`

```
samples_Monly <- link_df(M5.2,test_d)
summarised_samples_Aonly<-group_by(samples_Monly,M) %>%
  summarise(mean_mu=mean(mu),
            lower_mu=quantile(mu,0.1),
            upper_mu=quantile(mu,0.9))%>%
  ungroup()

ggplot(summarised_samples_Aonly,aes(x =M, y =mean_mu)) +
  geom_ribbon(aes(ymin=lower_mu,ymax=upper_mu),alpha=0.5)+
  geom_point(data=d,aes(x=M,y=D))
```



Third model, both marriage age and marriage rate matters

$D_i \sim \text{Normal}(\mu_i, \sigma) \mu_i = a + bA * A_i + bM * M_i a \sim \text{Normal}(0, 0.2) bA \sim \text{Normal}(0, 0.5) bM \sim \text{Normal}(0, 0.5) \sigma \sim \text{Exponential}(1)$

Note that `A` and `M` appear in a completely interchangeable way here. So we could say that we are interested in the effect of `A` after controlling for `M`, or the effect of `M` after controlling for `A`, the model won't know the difference!

Write the quap model for this- you need to fill in the "alist" part:

Have a look at the means and likely intervals of the parameters. `precis` is a good method for this.

```
##                 mean         sd       5.5%      94.5%
## a     -1.634688e-05 0.09707779 -0.1551654  0.1551327
## bA    -6.134346e-01 0.15098797 -0.8547426 -0.3721267
## bM    -6.552359e-02 0.15077462 -0.3064906  0.1754434
## sigma  7.851367e-01 0.07784815  0.6607203  0.9095530
```

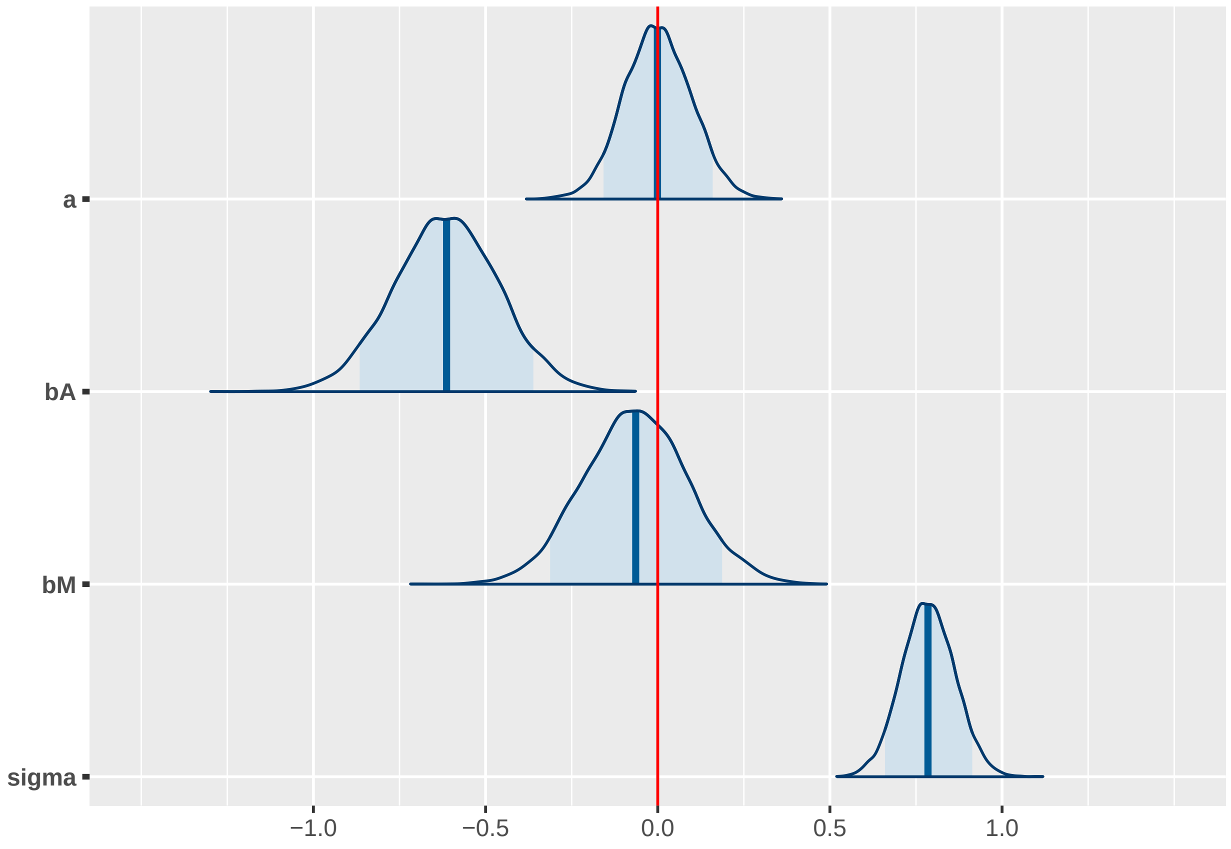
Extract the samples

```
samples_M5.3 <- extract.samples(M5.3)
```

Use `mcmc_areas` to get the parameter distributions plotted in a way that we can visualize their limits. Here you should interpret what you see, what is this telling us about how information from `A` and `M` combine?

```
bayesplot::mcmc_areas(samples_M5.3,prob=0.9 , area_method="equal height") +
  geom_vline(xintercept = 0 , color="red")+
  scale_x_continuous(limits=c(-1.5,1.5),
                     breaks = c(-1,-0.5,0,0.5,1))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



counterfactual plots and posterior predictive plots

We can use the fitted model in order to ask how the model represents the actually observed predictor variables, or how sets of predictor variables that we did not observe are represented.

The recipe is:

1. Create a dataframe containing the sets of predictor variable measurements you want to include.

2. Use `link_df` or `sim_df` to create a sample of mean values or a sample of simulated values.

3. Create a summary or plot with these sampled values.

For this we'll explore how a sequence of values of one of the predictor variables affects the outcome while the other variable is left at it's mean valeu.

```r
test_d <- tibble(M=seq(from=-2,to=2,by=0.1),A=0)
```

And run this back through `link_df`

```r
samples_M5.3 <- link_df(M5.3,test_d)
summarised_samples_M5.3<-group_by(samples_M5.3,index) %>%
  summarise(mean_mu=mean(mu),
            lower_mu=quantile(mu,0.1),
            upper_mu=quantile(mu,0.9),
            M=M)%>%
  ungroup()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'index'. You can override using the
## `.groups` argument.
```

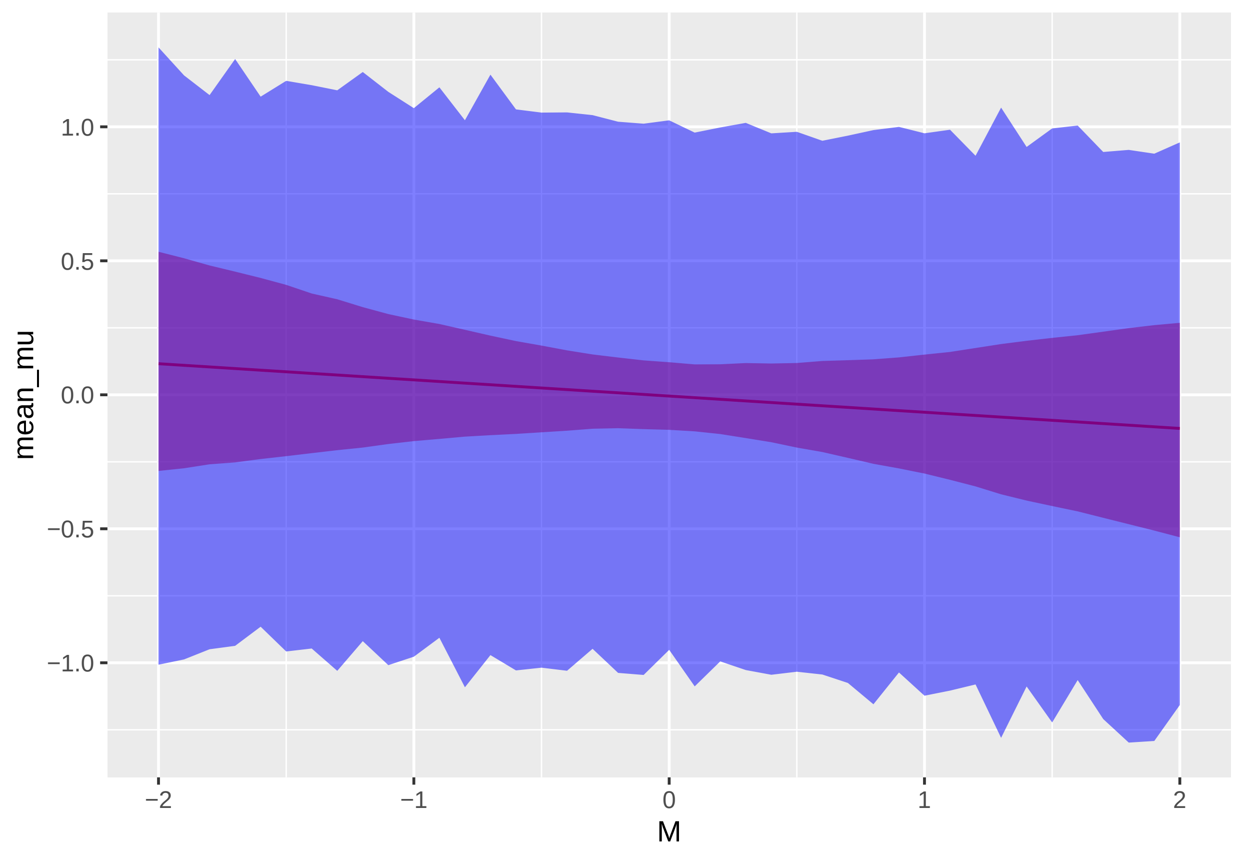
We'll run it through `sim_df`, too. This way we can look at both variance in the parameters and expected variance in observations themselves.

```r
simulations_M5.3 <- sim_df(M5.3,test_d)
summarised_simulations_M5.3<-group_by(simulations_M5.3,index) %>%
  summarise(mean_D=mean(D),
            lower_D=quantile(D,0.1),
            upper_D=quantile(D,0.9),
            M=M)%>%
  ungroup()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'index'. You can override using the
## `.groups` argument.
```

```r
ggplot(summarised_samples_M5.3,aes(x = M, y =mean_mu)) +
  geom_line(color="red")+
  geom_ribbon(aes(ymin=lower_mu,ymax=upper_mu),alpha=0.5,fill="red")+
  geom_ribbon(data=summarised_simulations_M5.3,inherit.aes = FALSE,
              aes(x=M,ymin=lower_D,ymax=upper_D),alpha=0.5,fill="blue")
```



Now you make the complementary figure where $M = 0$ and A is varied.

```
test_d <- tibble(A=seq(from=-2,to=2,by=0.1),M=0)
```

```
test_d
```

```
## # A tibble: 41 x 2
##         A     M
##     <dbl> <dbl>
##  1  -2       0
##  2  -1.9     0
##  3  -1.8     0
##  4  -1.7     0
##  5  -1.6     0
##  6  -1.5     0
##  7  -1.4     0
##  8  -1.3     0
##  9  -1.2     0
## 10  -1.1     0
## # i 31 more rows
```

And run this back through `link_df`

```
samples_M5.3 <- link_df(M5.3,test_d)
summarised_samples_M5.3<-group_by(samples_M5.3,index) %>%
```

```r
  summarise(mean_mu=mean(mu),
            lower_mu=quantile(mu,0.1),
            upper_mu=quantile(mu,0.9),
            A=A)%>%
  ungroup()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'index'. You can override using the
## `.groups` argument.
```

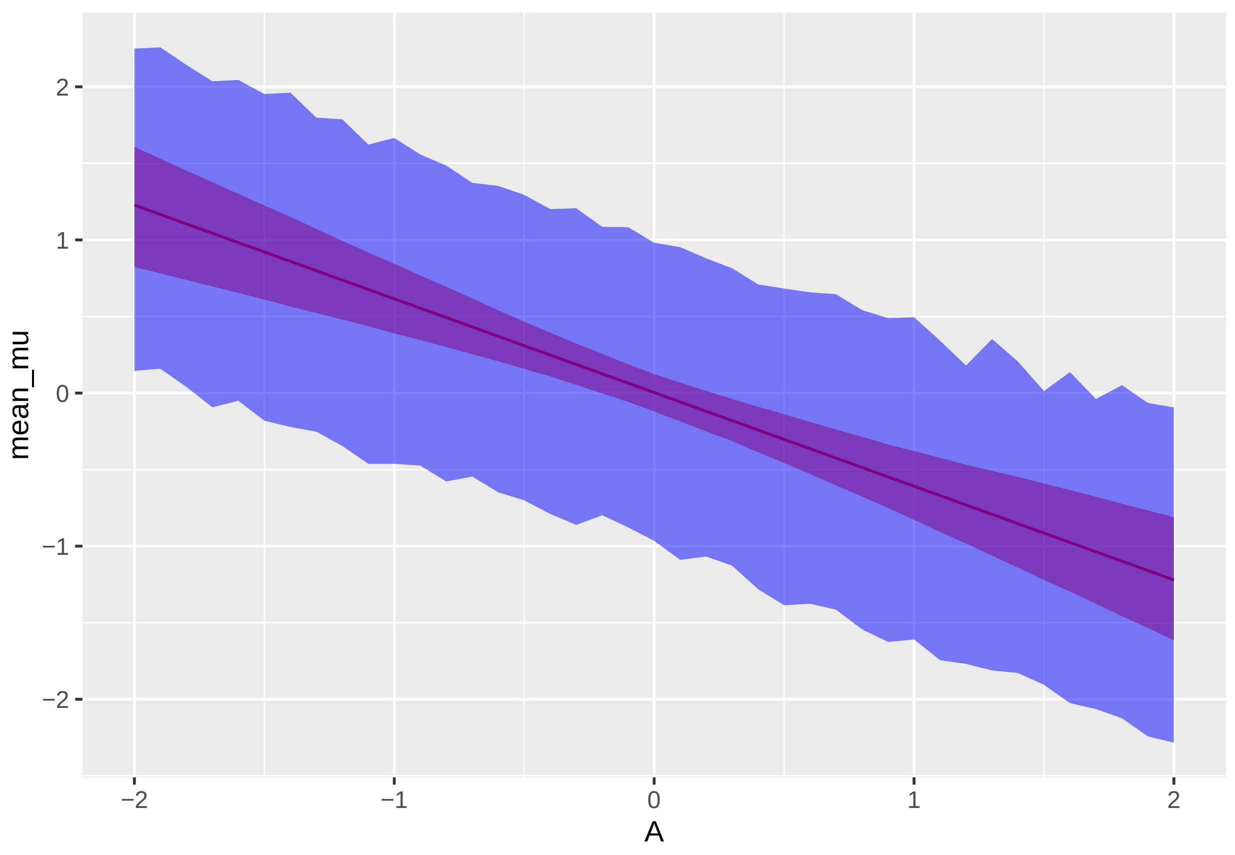
We'll run it through `sim_df`, too. This way we can look at both variance in the parameters and expected variance in observations themselves.

```r
simulations_M5.3 <- sim_df(M5.3,test_d)
summarised_simulations_M5.3<-group_by(simulations_M5.3,index) %>%
  summarise(mean_D=mean(D),
            lower_D=quantile(D,0.1),
            upper_D=quantile(D,0.9),
            A=A)%>%
  ungroup()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'index'. You can override using the
## `.groups` argument.
```

```r
ggplot(summarised_samples_M5.3,aes(x = A, y =mean_mu)) +
  geom_line(color="red")+
  geom_ribbon(aes(ymin=lower_mu,ymax=upper_mu),alpha=0.5,fill="red")+
  geom_ribbon(data=summarised_simulations_M5.3,inherit.aes = FALSE,
            aes(x=A,ymin=lower_D,ymax=upper_D),alpha=0.5,fill="blue")
```


mean_mu
2
1
0
−1
−2
−2
−1
0
1
2
A