Day 5: Intro to Linear Regression

Stephen R. Proulx

1/10/2021

Today's objectives:

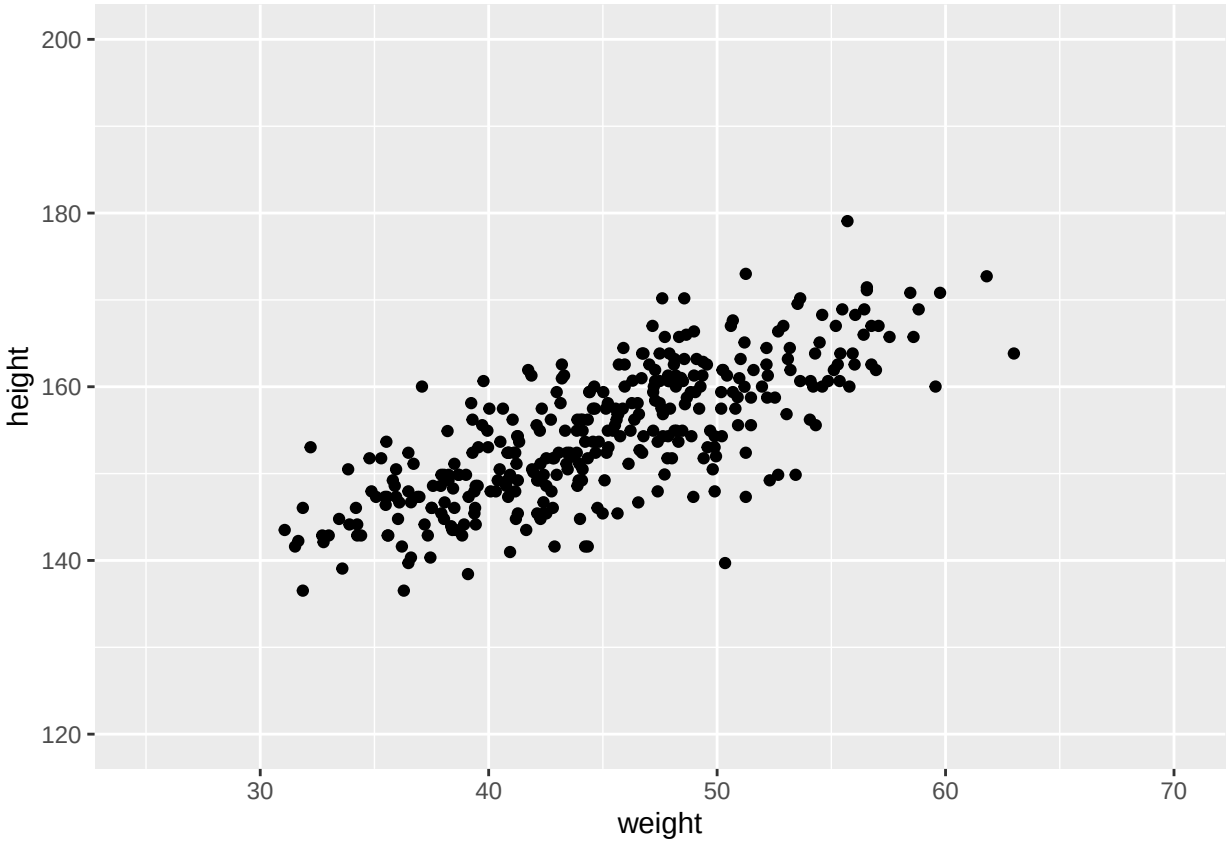
- Learn how to write notation for a linear regression type of model
- Learn how to use the "quadratic approximation" to calculate a posterior distribution
- Do a linear regression
- Plot samples and simulations from the posterior distribution

Height data, mean and variance

Here we will go through the example in the book that fits human height data using a normal likelihood function. Because normal distributions have both a mean and standard deviation, this is a two parameter model, so a grid approximation will really be a grid this time.

```
data("Howell1")
d<-Howell1
d2<- as_tibble(d)%>% filter(age>=18)


(data_plot <- ggplot(data=d2, aes(x=weight,y=height)) +
  geom_point() +
  xlim(25,70)+
  ylim(120,200))
```



Introduciton to quap: Quadratic Approximation

We're going to treat this mostly as a black box that we will use on our way to generating samples from a posterior distribution with MCMC methods. The quadratic approximation requires that we be able to calculated the posterior probability for specific values of our parameters, and then searches for the top of the posterior probability hill. Once it gets there, it just figures out how curved the hill top is and then approximates the entire posterior distribution from that.

For our purposes we need:
1. To learn the syntax for calling quap
2. To recognize error messages
3. To create a dataframe with samples of parameters from the quap approximation.

quap syntax

Here we recreate the model that we used for grid approximation of the height data. The syntax involves making an `alist` that is our model description, and defining the data with `data=`. Note that each line in the alist must have a `,` to end with.

Possible entries in the `alist` include the likelihood:

$$data \sim \text{likelihood}(parameters)$$

Transformations among parameters. These use the R `<-` assignmnet, not `=` :

$$parameter_1 < -\text{transformation}(parameter_2)$$

And priors:

$$parameter_i \sim \text{prior}$$

The order of the equations in the model matters. You need to list the likelihood, then the transformations, and then the priors.

So our height model is:

```
quap_model_height <- quap(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu ~ dnorm( 178 , 20 ) ,
        sigma ~ dunif( 0 , 50 )
    ) , data=d2 )
```

It runs and returns no errors. We can see the summary with

```
precis(quap_model_height)
```

```
##             mean        sd      5.5%      94.5%
## mu    154.607388 0.4119727 153.94898 155.265800
## sigma   7.730919 0.2913470   7.26529   8.196548
```

Which should look pretty familiar.

quap errors

quap does a simple hill climbing procedure, and like anyone looking closely at a topo map, it can easily wander off and get lost. Errors usually have to do with the random starting positon on the map leading off to nowhere, or that it takes too many steps without getting to a flat area and gives up (this is by design, so you don't have a function that never stops running).

Here I've set a unreasonable prior that causes the likelihood to be so small and cause numerical errors.

```
quap_model_height_bad_prior <- quap(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu ~ dnorm( 100 ,2) ,
        sigma ~ dunif( 0 , 50 )
    ) , data=d2 )
```

Extracting samples

The rethinking command `extract.samples` does exactly what we need.

```
(samples_height_model <- extract.samples(quap_model_height,n=1e4) %>%
  as_tibble() )
```

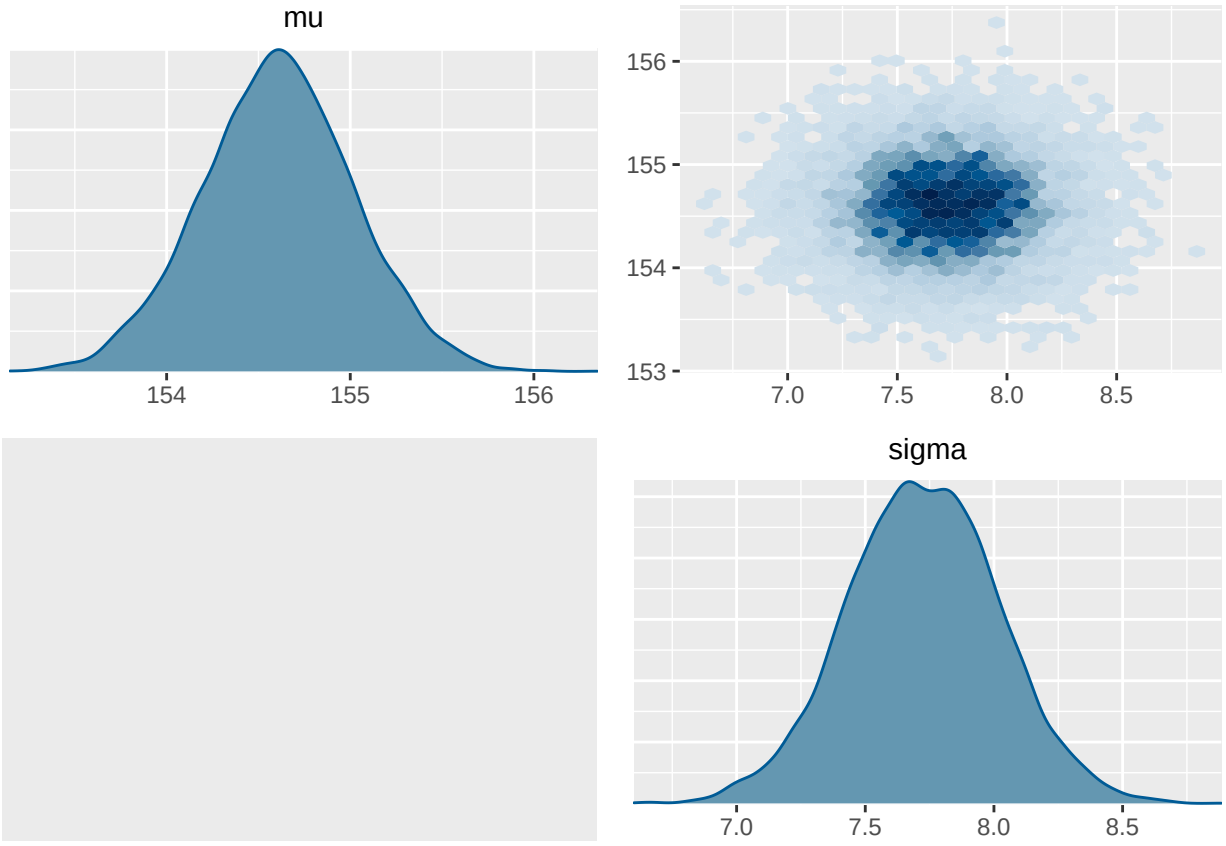
```
## # A tibble: 10,000 x 2
##       mu sigma
##    <dbl> <dbl>
##  1  155.  7.96
```

```
##  2  155.  8.27
##  3  155.  7.25
##  4  155.  7.64
##  5  154.  8.02
##  6  153.  8.15
##  7  155.  7.32
##  8  155.  7.28
##  9  154.  7.99
## 10  154.  7.78
## # ... with 9,990 more rows
```

And these samples can now be worked with as before, and give the same results (up to random noise).

```
bayesplot::mcmc_pairs(samples_height_model,diag_fun = "dens",
  off_diag_fun =  "hex")
```

```
## Warning: Only one chain in 'x'. This plot is more useful with multiple
## chains.
```



A linear regression model

Now we are going to do a linear regression

$$y_i \sim \text{Normal}(\mu, \sigma) \mu_i = a + b * weight_i a \sim \text{Normal}(178, 20) b \sim \text{Normal}(178, 20) \sigma \sim \text{Uniform}(0, 50)$$

```
quap_model_height_weight <- quap(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu <- a + b*( weight  ) ,
        a ~ dnorm( 178 , 20 ) ,
        b ~ dlnorm( 0 , 1 ) ,
        sigma ~ dunif( 0 , 50 )
    ) , data=d2 )
```

```
precis(quap_model_height_weight)
```

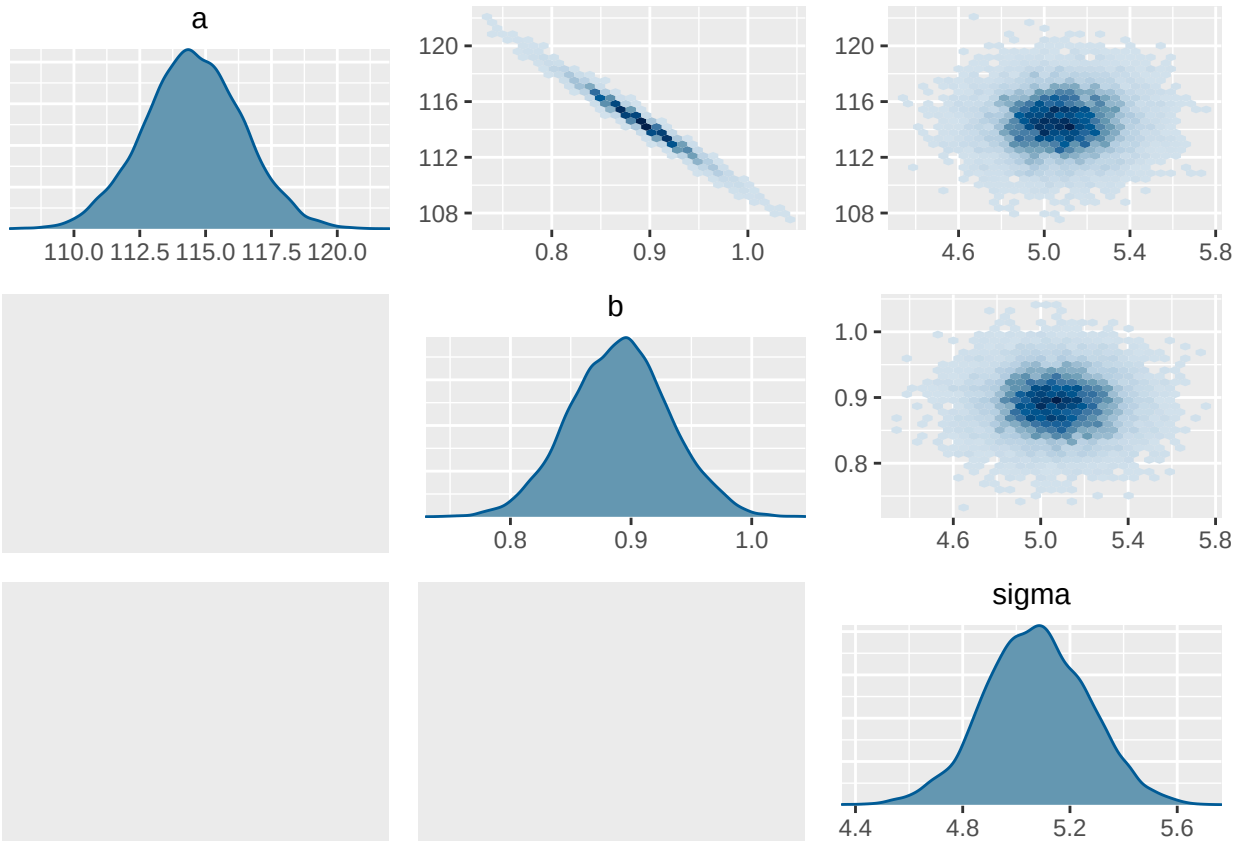
```
##             mean         sd        5.5%       94.5%
## a     114.534136 1.89774721 111.5011698 117.5671029
## b       0.890734 0.04175799   0.8239966   0.9574713
## sigma   5.072720 0.19124901   4.7670669   5.3783726
```

```
(samples_height_weight_model <- extract.samples(quap_model_height_weight,n=1e4) %>%
  as_tibble() )
```

```
## # A tibble: 10,000 x 3
##        a     b sigma
##    <dbl> <dbl> <dbl>
##  1  112. 0.945  5.24
##  2  111. 0.956  5.08
##  3  113. 0.921  4.94
##  4  115. 0.876  4.89
##  5  113. 0.926  5.08
##  6  118. 0.831  4.95
##  7  115. 0.888  5.22
##  8  115. 0.885  4.92
##  9  112. 0.933  5.17
## 10  113. 0.938  4.91
## # ... with 9,990 more rows
```

```
bayesplot::mcmc_pairs(samples_height_weight_model,diag_fun = "dens",
  off_diag_fun =  "hex")
```

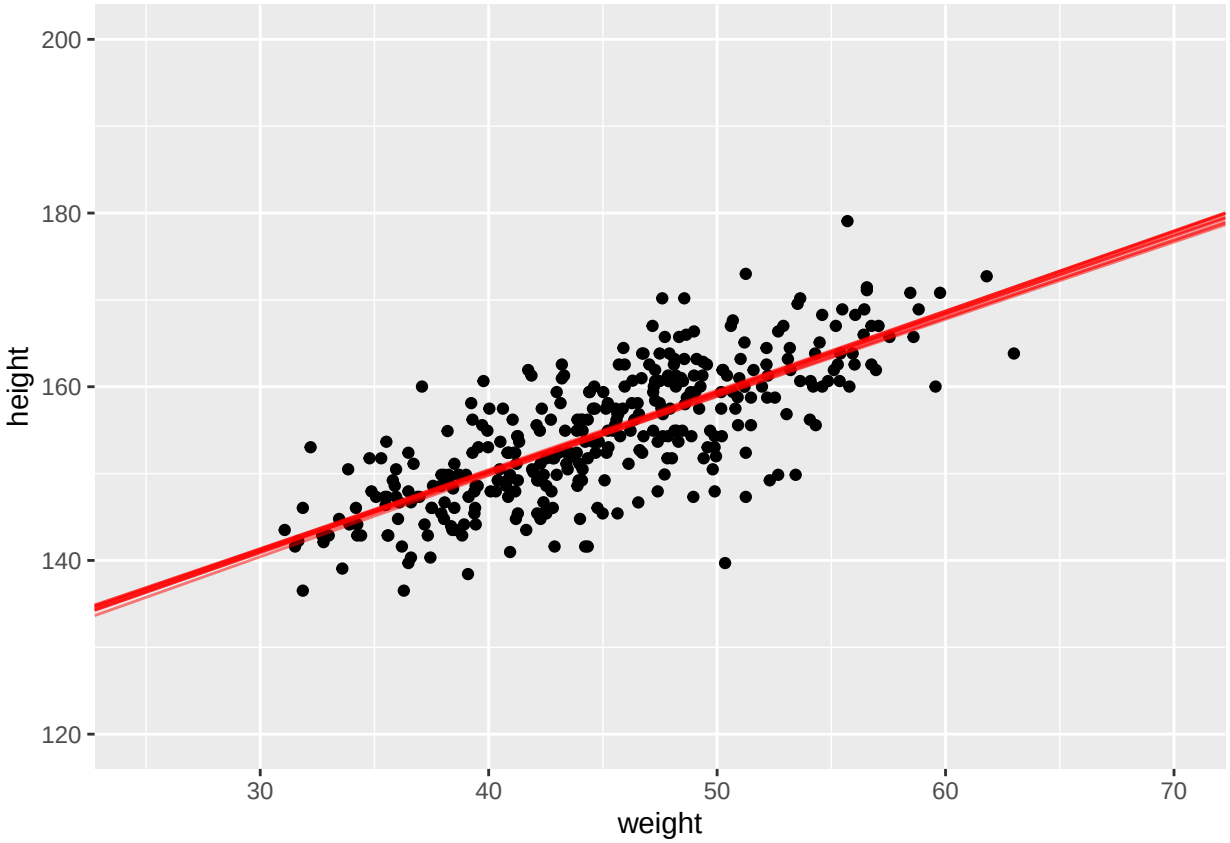
```
## Warning: Only one chain in 'x'. This plot is more useful with multiple
## chains.
```



Plotting the linear fits

```
subsamples <-sample_n(samples_height_weight_model,size=10)

data_plot +
  geom_abline(intercept = subsamples$a, slope = subsamples$b, alpha=0.5, color="red")
```



Using the rethinking package functions `link` and `sim`

We'll actually be using the functions `link_df` and `sim_df`. These use the rethinking package functions `link` and `sim`, but do some extra work to return a neat dataframe that you can then plot or summarise. I wrote the wrapper function, it is included in the file "helper.R". Since I wrote these they have not been extensively tested, and this means they may fail if you use them in an unanticipated way.

using link: When we generate samples from `quap` they are samples of the parameters, which in the case are the intercept `a` and slope, `b`. We often want to reverse our transformation formula to get results in terms of the parameters that go into the likelihood themselves. In this case, that means `mu`. So using `link_df` will give us samples in terms of `mu`.

We need to supply `link_df` with a quap model and with

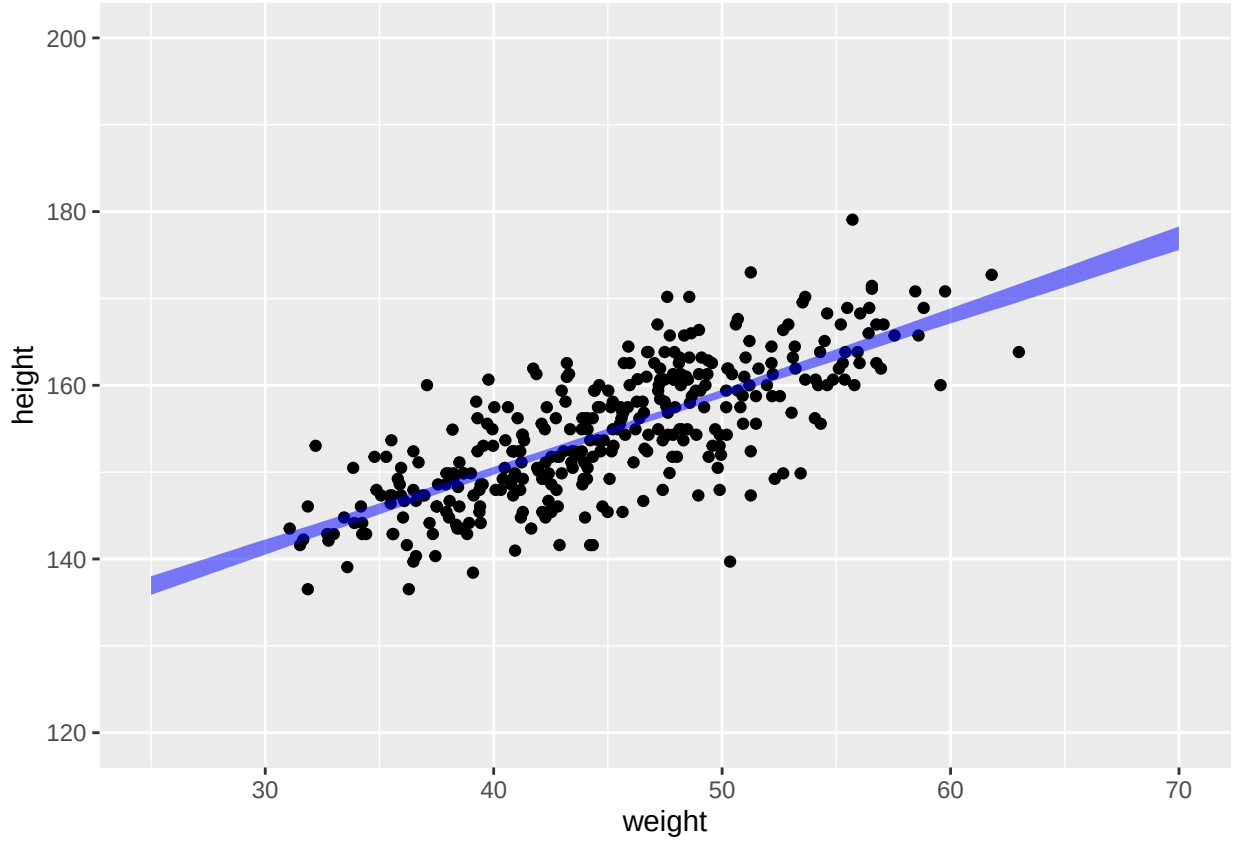
```
weights<-tibble(weight=seq(from=25, to=70, by=1),age=40)
samples_height_weight_mu <- link_df(quap_model_height_weight,data=weights , n=100)
```

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

## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(i)‘ instead of ‘i‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
samples_summarized <- samples_height_weight_mu %>%
  group_by(weight)%>%
  summarise(mean_mu=mean(mu),
            lower_mu=quantile(mu,0.1),
            upper_mu=quantile(mu,0.9))%>%
  ungroup()

(data_plot_means=data_plot +
  geom_ribbon(data=samples_summarized, inherit.aes = FALSE, aes(x=weight, ymin=lower_mu,ymax=upper_mu),
```

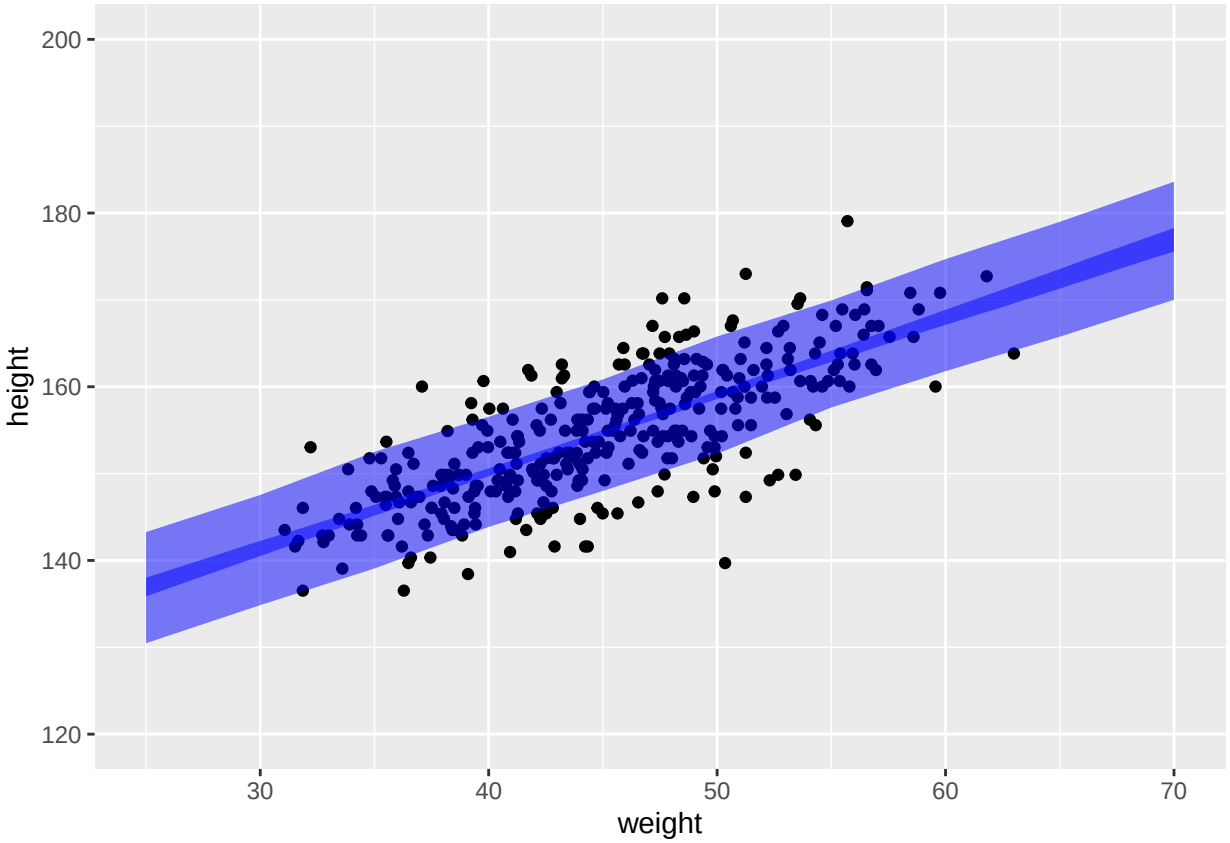


Using sim: `sim_df` is similar to `link_df`, but outputs simulated data instead of the sampled parameters.

```
weights=tibble(weight=seq(from=25,to=70,by=5))
simulated_height_weight <- sim_df(quap_model_height_weight,data=weights)
```

```
simulations_summarized <- simulated_height_weight %>%
  group_by(weight)%>%
  summarise(mean_height=mean(height),
            lower_height=quantile(height,0.1),
            upper_height=quantile(height,0.9))%>%
  ungroup()

data_plot_means +
  geom_ribbon(data=simulations_summarized, inherit.aes = FALSE, aes(x=weight, ymin=lower_height,ymax=up
```



Centered predictors for the linear regression model

Now we will re-do the same analysis, but with a "centered" description of the weight data. This helps make the parameters of the linear model make more sense.

To start, add a column to your dataframe which is the weight of the observed individual minus the mean weight in the population as a whole.

```
d2 <- mutate(d2, centered_weight = weight - mean(weight))
```

Now perform a `quap` model of the data, but using your new centered weight column as the linear model predictor. Give the new model a new name.

```
quap_model_height_weight_centered <- quap(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu <- a + b*( centered_weight  ) ,
        a ~ dnorm( 178 , 20 ) ,
        b ~ dlnorm( 0 , 1 ) ,
        sigma ~ dunif( 0 , 50 )
    ) , data=d2 )
```

```
precis(quap_model_height_weight_centered)
```

```
##              mean         sd        5.5%       94.5%
## a     154.6013672 0.27030753 154.1693635 155.0333708
## b       0.9032807 0.04192361   0.8362787   0.9702827
## sigma   5.0718784 0.19115455   4.7663765   5.3773803
```

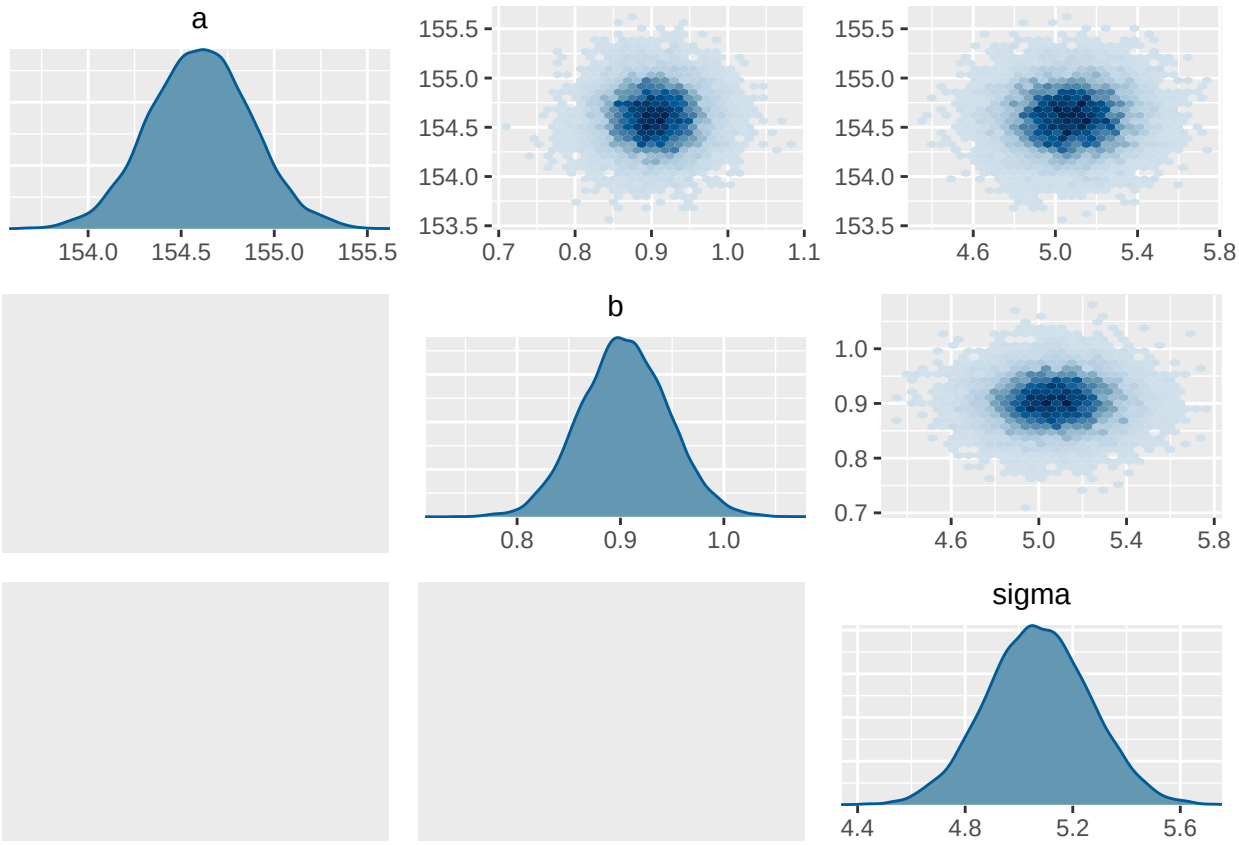
Now sample from the posterior to get a list of parameter values

```
(samples_height_weight_model_centered <- extract.samples(quap_model_height_weight_centered,n=1e4) %>%
  as_tibble() )
```

```
## # A tibble: 10,000 x 3
##        a     b sigma
##    <dbl> <dbl> <dbl>
##  1  155. 0.867  4.76
##  2  154. 0.920  5.15
##  3  155. 0.860  5.20
##  4  155. 0.909  5.09
##  5  155. 0.829  5.14
##  6  154. 0.914  5.53
##  7  155. 0.950  4.97
##  8  155. 0.954  5.21
##  9  155. 0.947  5.10
## 10  155. 0.876  4.91
## # ... with 9,990 more rows
```

```
bayesplot::mcmc_pairs(samples_height_weight_model_centered,diag_fun = "dens",
  off_diag_fun =  "hex")
```

```
## Warning: Only one chain in 'x'. This plot is more useful with multiple
## chains.
```



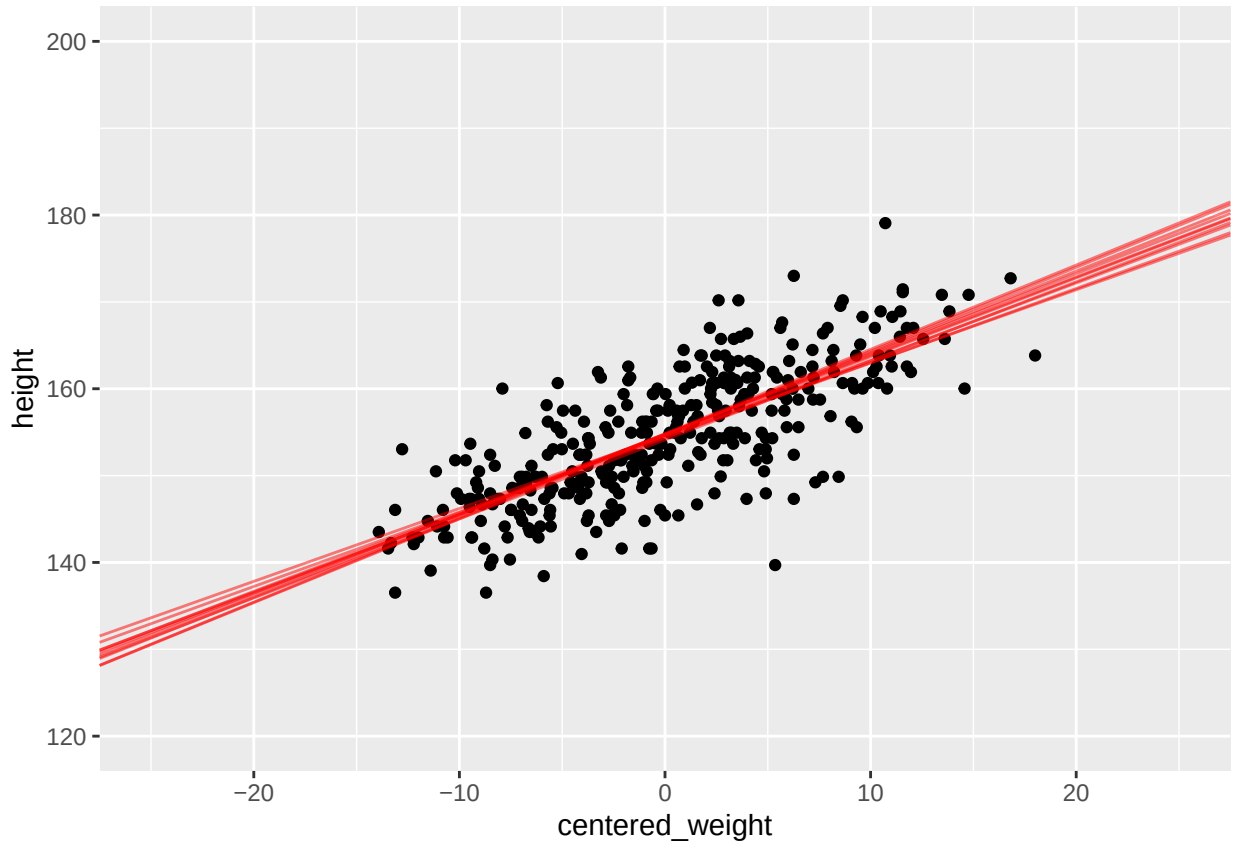
Plot some of the lines that are part of your sample. Be sure to plot the data with the centered weight as the x axis to compare the data to the prediction lines.

Plotting the linear fits

```
data_plot_centered <- ggplot(data=d2, aes(x=centered_weight,y=height)) +
  geom_point() +
  xlim(-25,25)+
  ylim(120,200)


subsamples <-sample_n(samples_height_weight_model_centered,size=10)

data_plot_centered +
  geom_abline(intercept = subsamples$a, slope = subsamples$b, alpha=0.5, color="red")
```



Use `link_df` and `sim_df` to create samples of the mean and to simulate from the posterior, and create a figure putting them all together.

```
weights<-tibble(centered_weight=seq(from=-25, to=25, by=1),age=40)
samples_height_weight_mu_centered <- link_df(quap_model_height_weight_centered,data=weights , n=100)
```

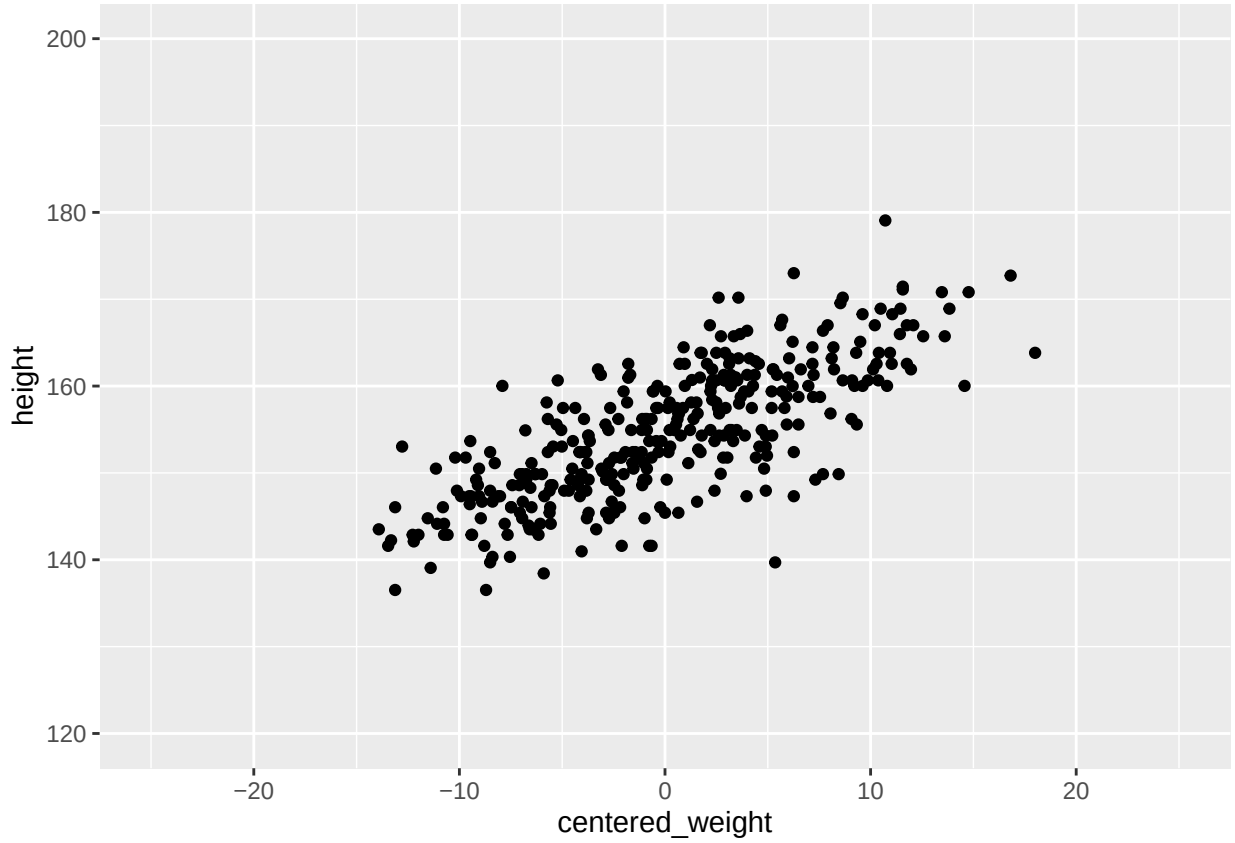
```
samples_summarized_centered <- samples_height_weight_mu_centered %>%
  group_by(centered_weight)%>%
  summarise(mean_mu=mean(mu),
            lower_mu=quantile(mu,0.1),
            upper_mu=quantile(mu,0.9))%>%
  ungroup()

samples_summarized_centered
```

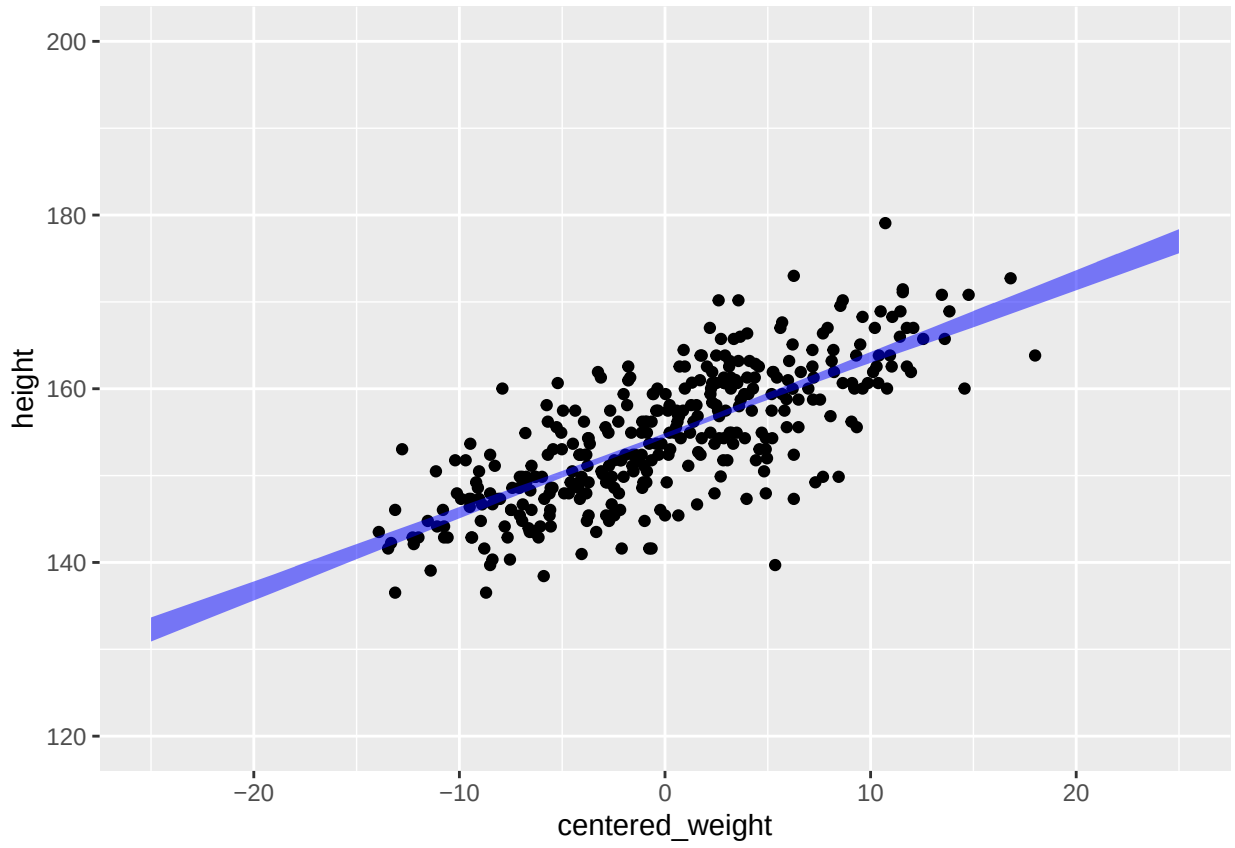
```
## # A tibble: 51 x 4
##    centered_weight mean_mu lower_mu upper_mu
##              <dbl>   <dbl>    <dbl>    <dbl>
##  1             -25    132.     131.     134.
##  2             -24    133.     132.     134.
##  3             -23    134.     133.     135.
##  4             -22    135.     134.     136.
##  5             -21    136.     135.     137.
##  6             -20    137.     136.     138.
##  7             -19    138.     137.     139.
```

```
##  8             -18    138.     138.     140.
##  9             -17    139.     138.     140.
## 10             -16    140.     139.     141.
## # ... with 41 more rows
```

```
data_plot_centered
```



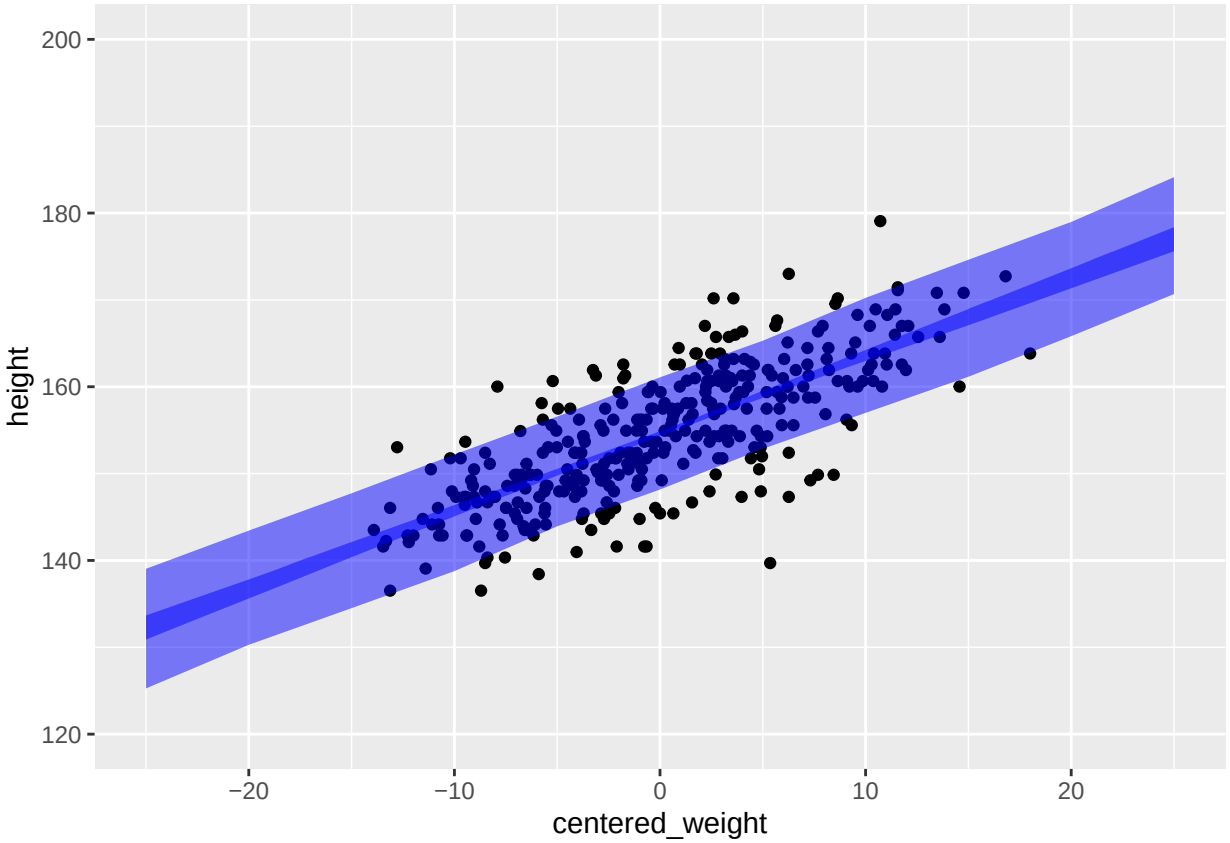

```
(data_plot_means_centered=data_plot_centered +
  geom_ribbon(data=samples_summarized_centered, inherit.aes = FALSE, aes(x=centered_weight, ymin=lower_
```




```
weights<-tibble(centered_weight=seq(from=-25, to=25, by=5),age=40)
simulated_height_weight_centered <- sim_df(quap_model_height_weight_centered,data=weights )
```

```
simulations_summarized_centered <- simulated_height_weight_centered %>%
  group_by(centered_weight)%>%
  summarise(mean_height=mean(height),
            lower_height=quantile(height,0.1),
            upper_height=quantile(height,0.9))%>%
  ungroup()

data_plot_means_centered +
  geom_ribbon(data=simulations_summarized_centered, inherit.aes = FALSE, aes(x=centered_weight, ymin=lo
```



Exercise

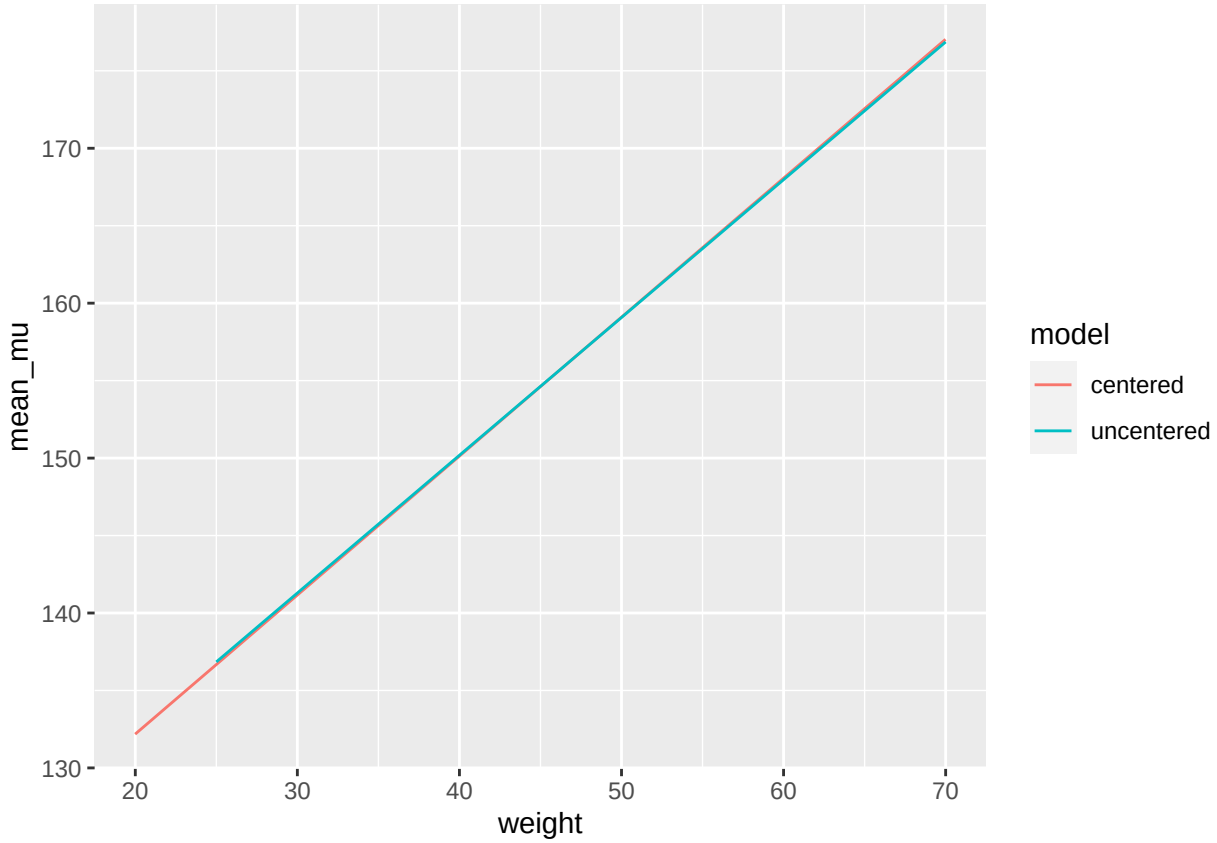
compare the sampled value of mu for the un-centered and centered models

```
samples_summarized_centered <- mutate(samples_summarized_centered,weight=centered_weight + mean(d2$weigh

samples_summarized <- mutate(samples_summarized,model="uncentered")


combinded_summary <- bind_rows(samples_summarized_centered,samples_summarized)

ggplot(combinded_summary, aes(x=weight,y=mean_mu,color=model))+geom_line()
```



alterntive method for plotting uncertainty in the esitmates and posterior simulations

You do not need to use this section, I just put it in to show you how to generate the figures without using the `link` and `sim` functions.

First add the weights to our dataframe of parameters sampled from the posterior. We make a list of the weights we want repeated a bunch of times so that each sample is paired with a random weight from our list of weights to include. We then use mutate to calculate the `mu` value for each sample and weight using the linear model based on the parameters `a` and `b`. Then we draw a normal variable with mean `mu` and sd `sigma`.

```
weights <- tibble(weight=rep(seq(from=25 , to = 70 , by=5),1e5))


samples_height_weight_model_alt <-bind_cols(samples_height_weight_model,slice(weights,1:(nrow(samples_h
  mutate(mu=a+b*weight,
         height_ran = rnorm(n(),mu,sigma))
```

Next we summarize the `mu` and `height_ran` values

```
samples_summarized <- samples_height_weight_model_alt %>%
  group_by(weight)%>%
  summarise(mean_mu=mean(mu),
```

```
            lower_mu=quantile(mu,0.1),
            upper_mu=quantile(mu,0.9),
            mean_draw=mean(height_ran),
            lower_draw=quantile(height_ran,0.1),
            upper_draw=quantile(height_ran,0.9))%>%
  ungroup()
```

And use ribbon plot as before.

```
data_plot +
  geom_ribbon(data=samples_summarized, inherit.aes = FALSE, aes(x=weight, ymin=lower_mu,ymax=upper_mu),
  geom_ribbon(data=samples_summarized, inherit.aes = FALSE, aes(x=weight, ymin=lower_draw,ymax=upper_dra
```