# Day 5: Intro to Linear Regression

Stephen R. Proulx

1/27/2025

## Today's objectives:

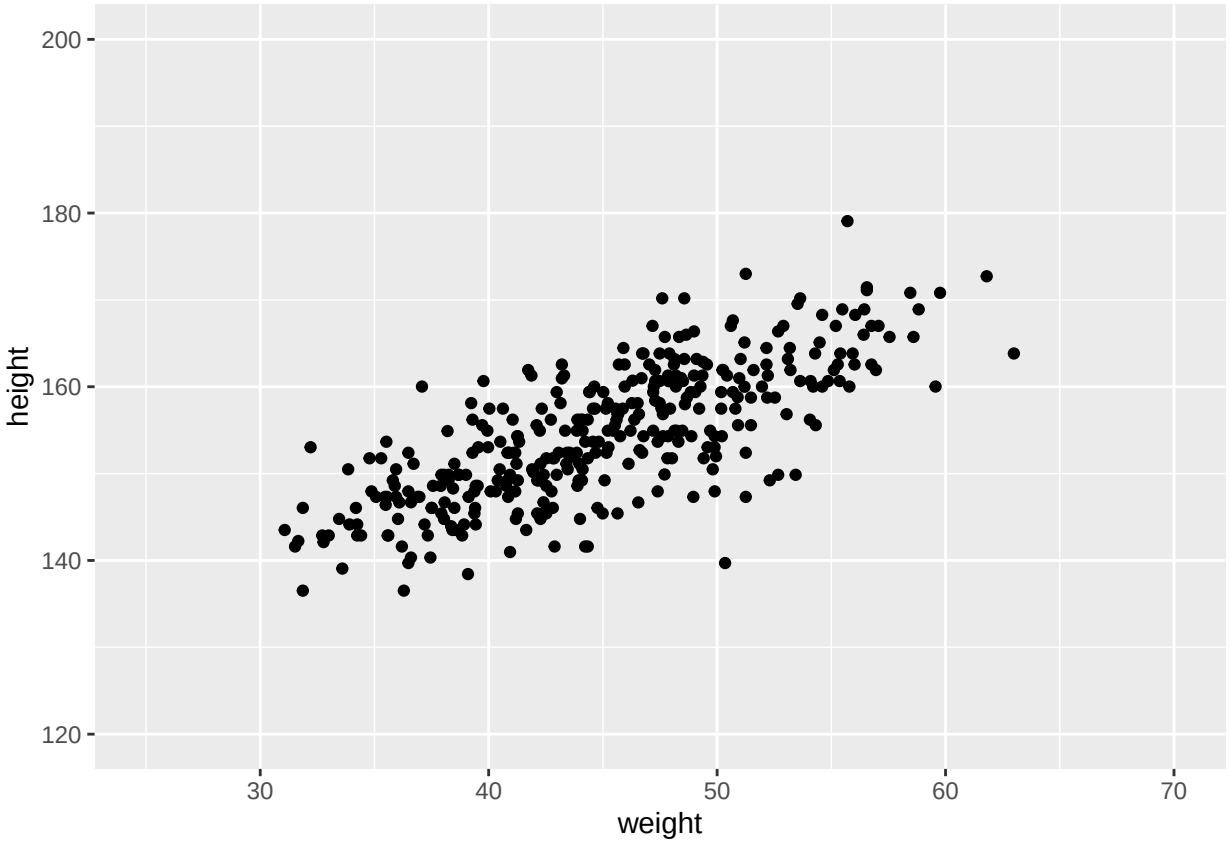
- Learn how to write notation for a linear regression type of model
- Learn how to use the "quadratic approximation" to calculate a posterior distribution
- Do a linear regression
- Plot samples and simulations from the posterior distribution

### 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 `<-` assignment, 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    154.607187 0.4120393 153.948669 155.265706
## sigma   7.732171 0.2914650   7.266354   8.197989
```

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 position 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 ,0.1) ,
        sigma ~ dunif( 0 , 50 )
    ) , data=d2 )
```

Let's run it again with more reasonable priors

```
quap_model_height_bad_prior <- quap(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu ~ dnorm( 120 ,5) ,
        sigma ~ dunif( 0 , 50 )
    ) , data=d2 )
```

### Extracting samples

The quap command returns a posterior distribution which is represented by a formula for a quadratic function, but we want to have samples from that posterior that we can use. Fortunately, the rethinking command `extract.samples` does exactly what we need.

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

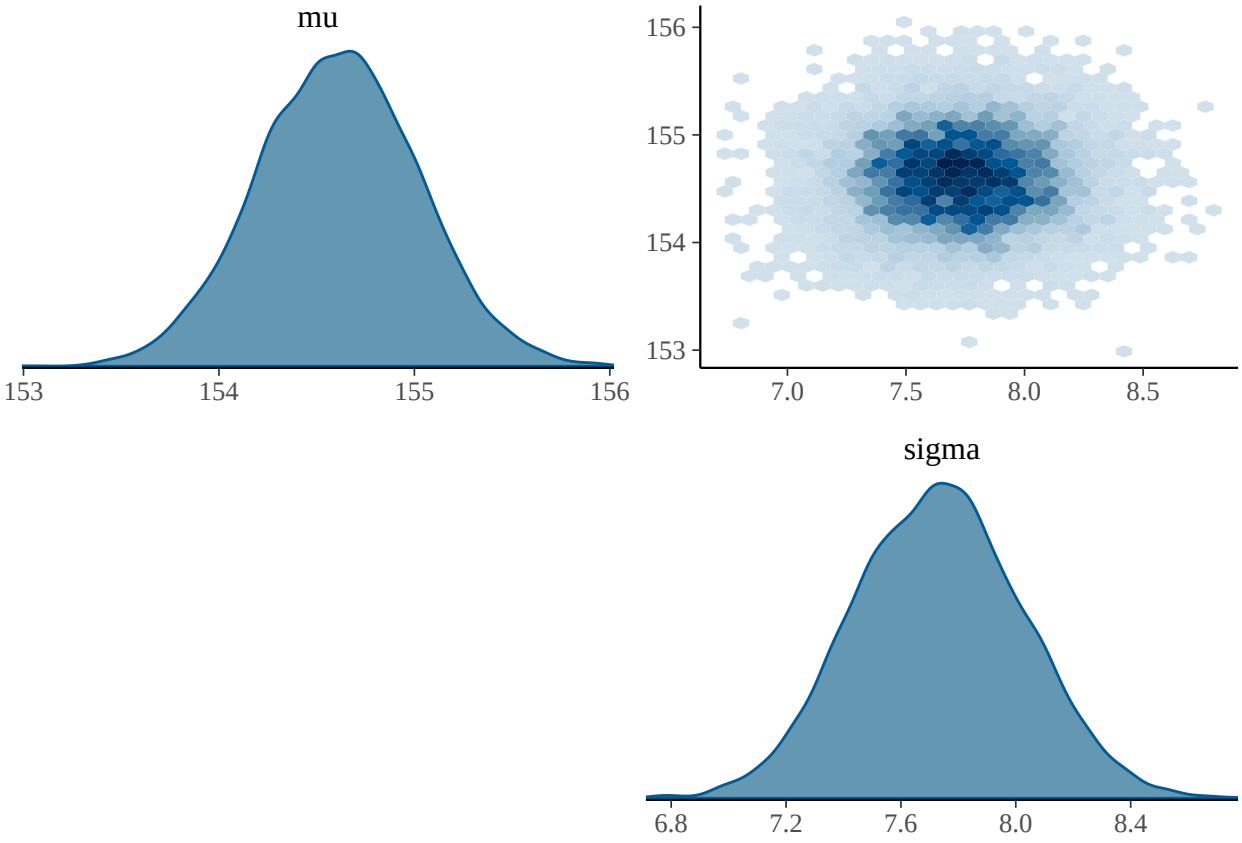
```
## # A tibble: 10,000 x 2
##       mu sigma
##    <dbl> <dbl>
```

```
##  1  154.  8.16
##  2  155.  7.31
##  3  154.  7.85
##  4  155.  7.39
##  5  155.  7.51
##  6  155.  8.05
##  7  155.  7.88
##  8  155.  8.05
##  9  154.  7.47
## 10  154.  7.71
## # i 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 )
```

common errors:

No sampling statement, we used = instead of ~ by mistake

```
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 )
```

Using rnorm instead of dnorm

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

parameter b has no prior:

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

using = instead of <- in the transformation block

```
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 )
```

forgot to pass along the data:

```
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 )
    ) )
```

**go back to the correct quap:**

```
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)
```

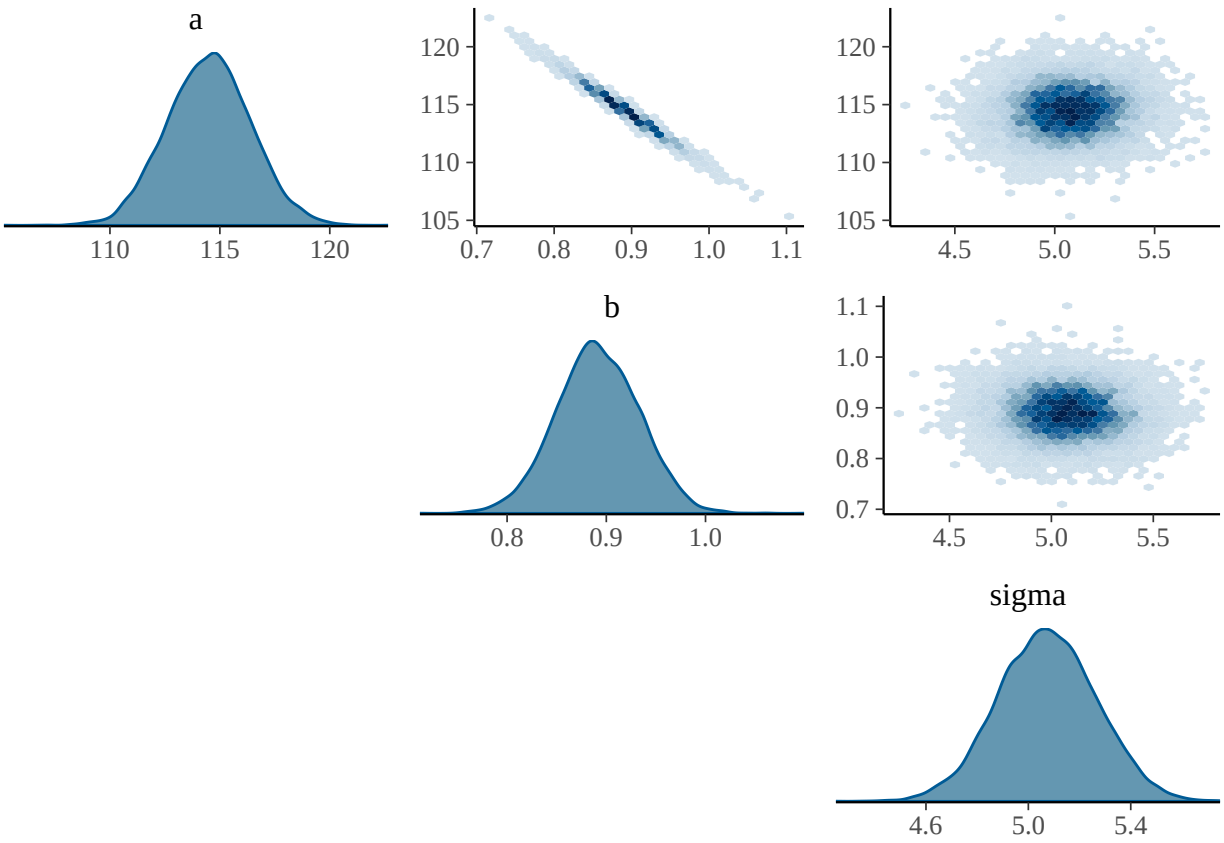
```
##              mean         sd        5.5%       94.5%
## a     114.5201767 1.89777752 111.4871617 117.5531917
## b       0.8910609 0.04175862   0.8243226   0.9577993
## sigma   5.0728705 0.19126257   4.7671960   5.3785451
```

```
(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  115. 0.887  5.06
##  2  112. 0.949  5.01
##  3  116. 0.857  4.93
##  4  114. 0.884  5.11
##  5  118. 0.814  5.34
##  6  114. 0.893  4.84
##  7  115. 0.887  5.21
##  8  114. 0.907  5.20
##  9  112. 0.934  5.15
## 10  114. 0.893  4.99
## # i 9,990 more rows
```

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

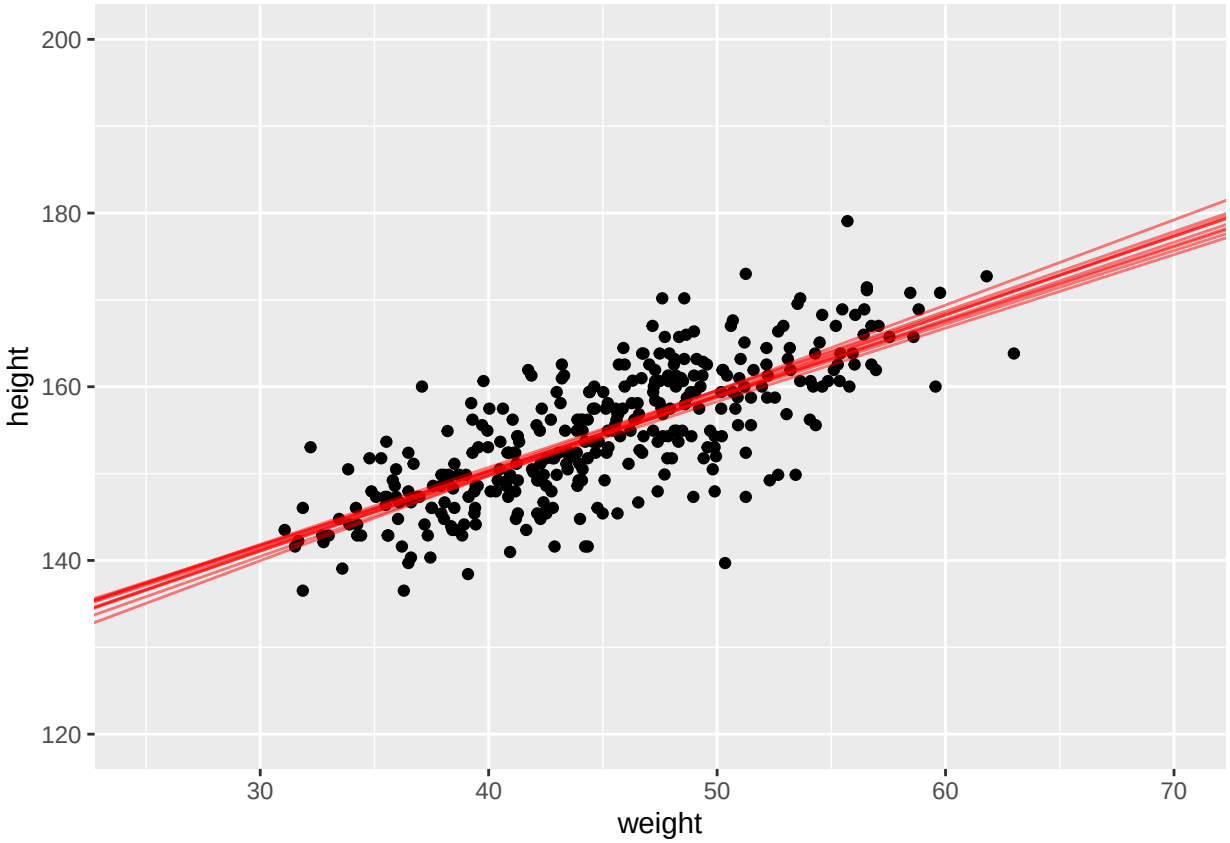
```
## Warning: Only one chain in 'x'. This plot is more useful with multiple chains.
```




**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 a set of input values. Here we look at weights from 25 to 70.

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

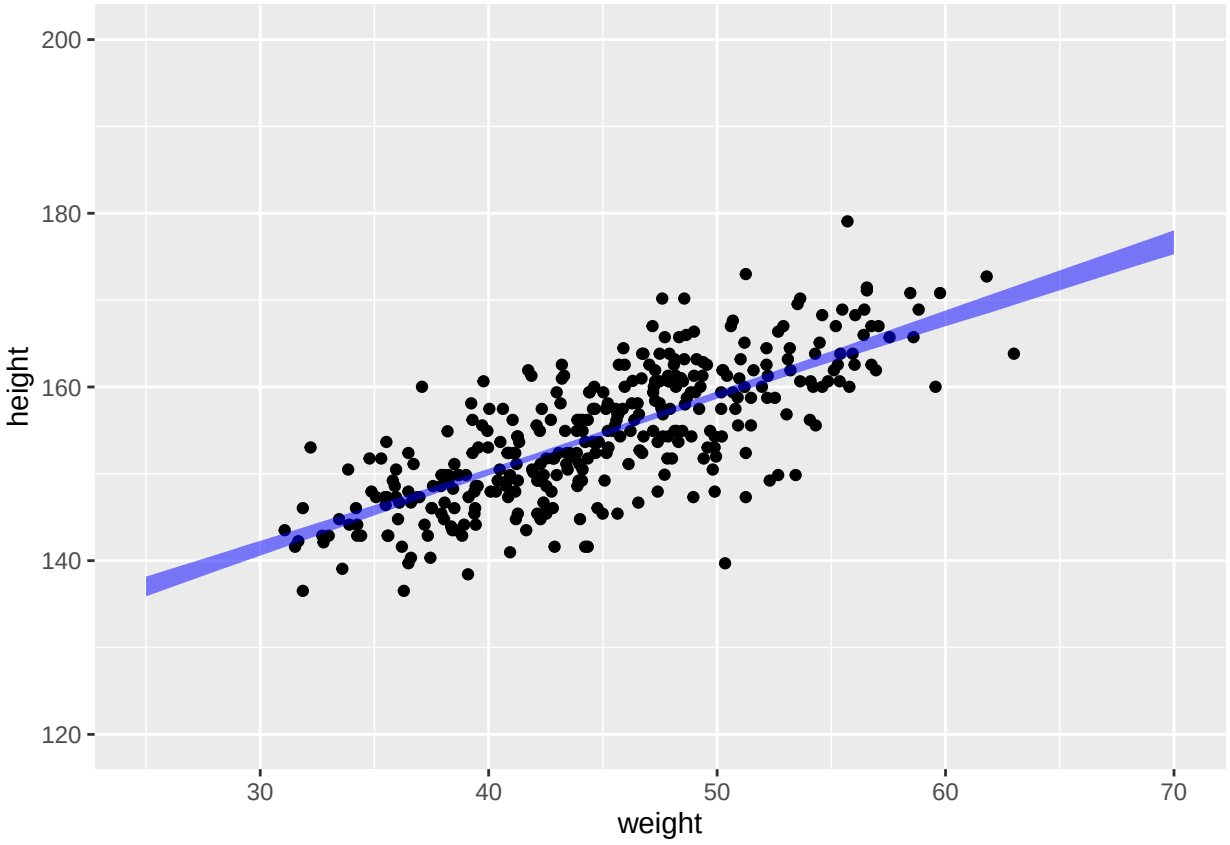
```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
## `.name_repair` is omitted as of tibble 2.0.0.
## i Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning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

```
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),
```



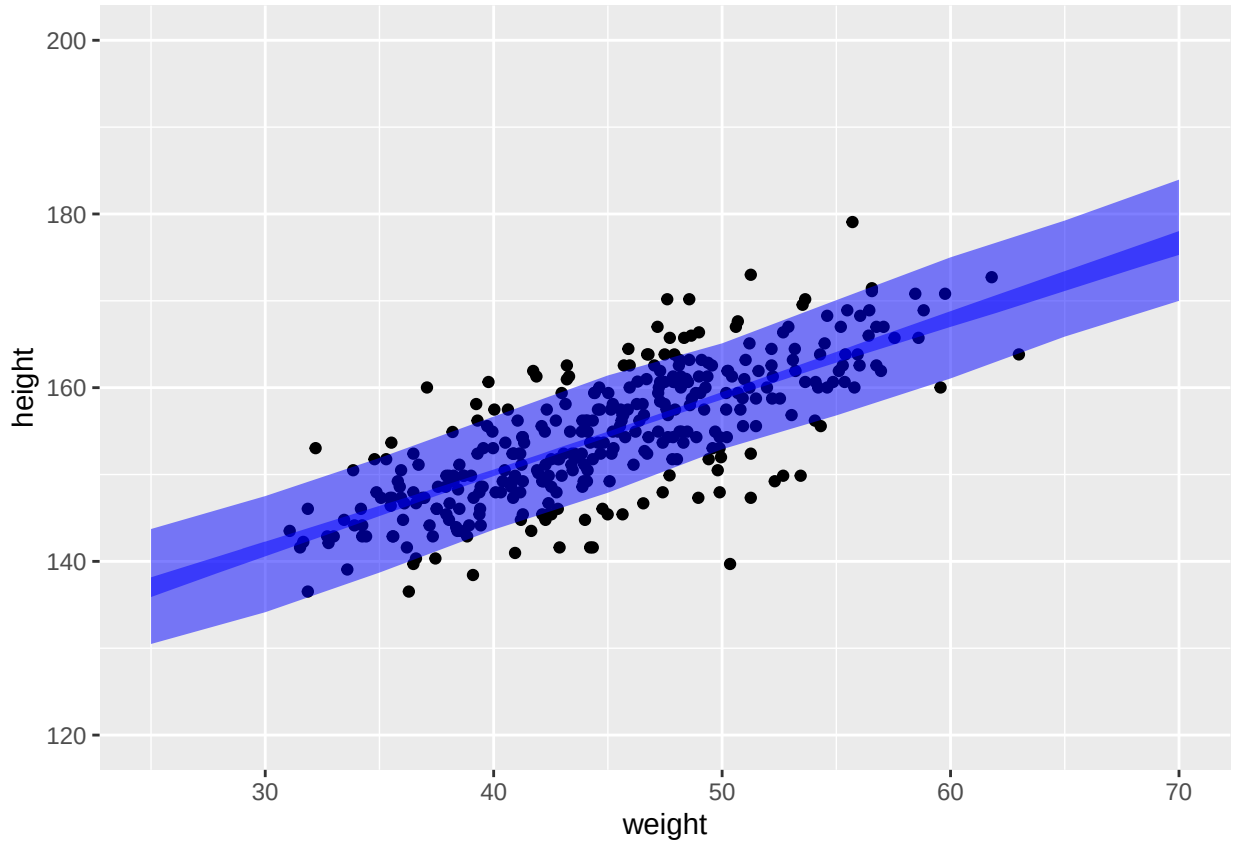

Using sim: `sim_df` is similar to `link_df`, but outputs simulated data instead of the sampled parameters.

```
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_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     154.6013676 0.27030765 154.1693638 155.0333714
## b       0.9032809 0.04192363   0.8362788   0.9702829
## sigma   5.0718806 0.19115476   4.7663784   5.3773828
```

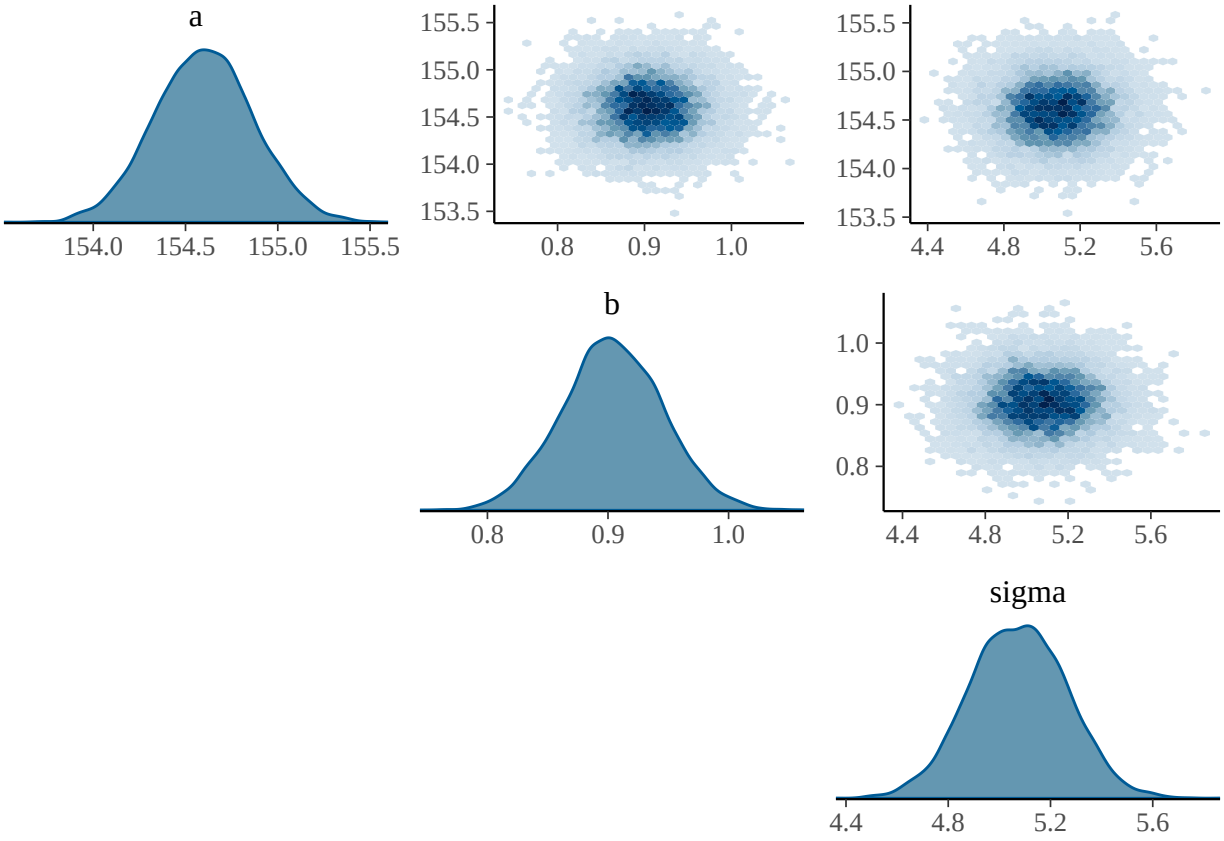
Now sample from the posterior to get a list of parameter values

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

Create the paired plot and compare to what we saw for the un-centered data.

```
bayesplot::mcmc_pairs(samples_height_weight_centered_model,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

First just create the background for showing the linear fits:

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

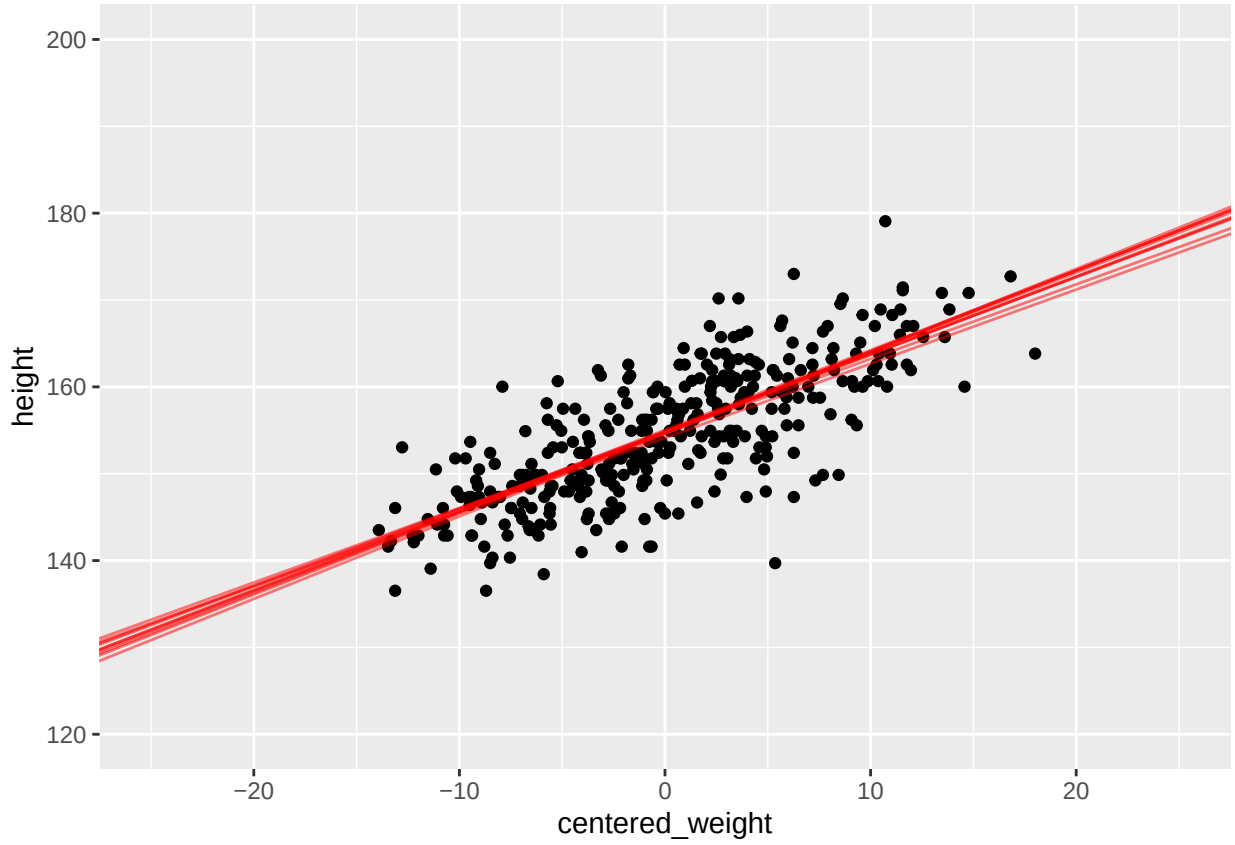
Use `link_df` and `sim_df` to create samples of the mean and to simulate from the posterior, and create a figure putting them all together.

We are using the `centered_weight` now, so this goes from -25 to +25. Create a tibble of weights and use `link_df` to find the posterior distribution for the `mu` parameter

Now get the 10 samples from the posterior and add them to the plot

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

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




Now that you have the samples for `mu`, we can find the 0.1 and 0.9 quantiles for each value of `centered_weight` in our posterior sample dataset. e are using the `centered_weight` now, so this goes from -25 to +25. Create a tibble of weights and use `link_df` to find the posterior distribution for the `mu` parameter

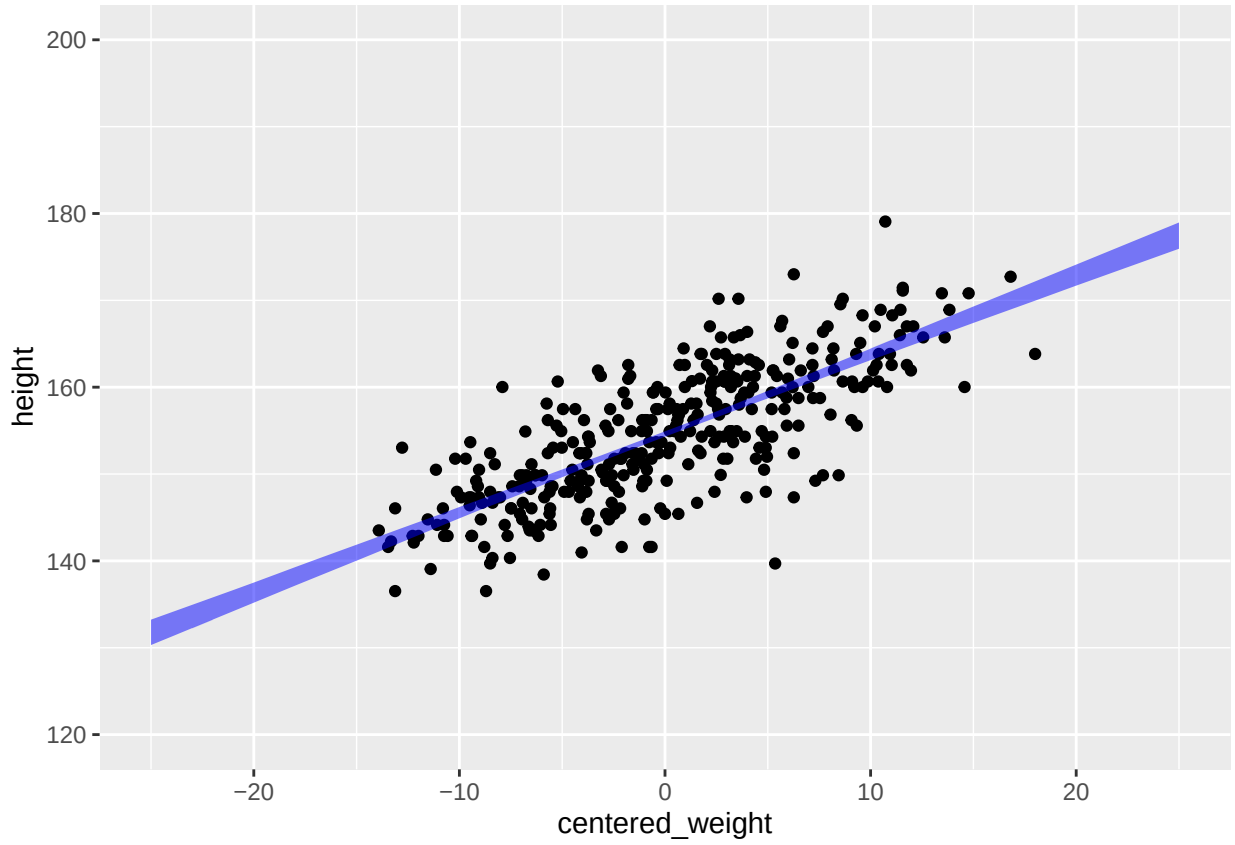
```
weights<-tibble(centered_weight=seq(from=-25, to=25, by=1),age=40)
samples_height_weight_mu_centered <- link_df(quap_model_height_weight_centered,data=weights , n=100)
```

Now that you have the samples for `mu`, we can find the 0.1 and 0.9 quantiles for each value of `centered_weight` in our posterior sample dataset.

```
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()
```

And now plot the data long with a `geom_ribbon` for the quantiles.

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

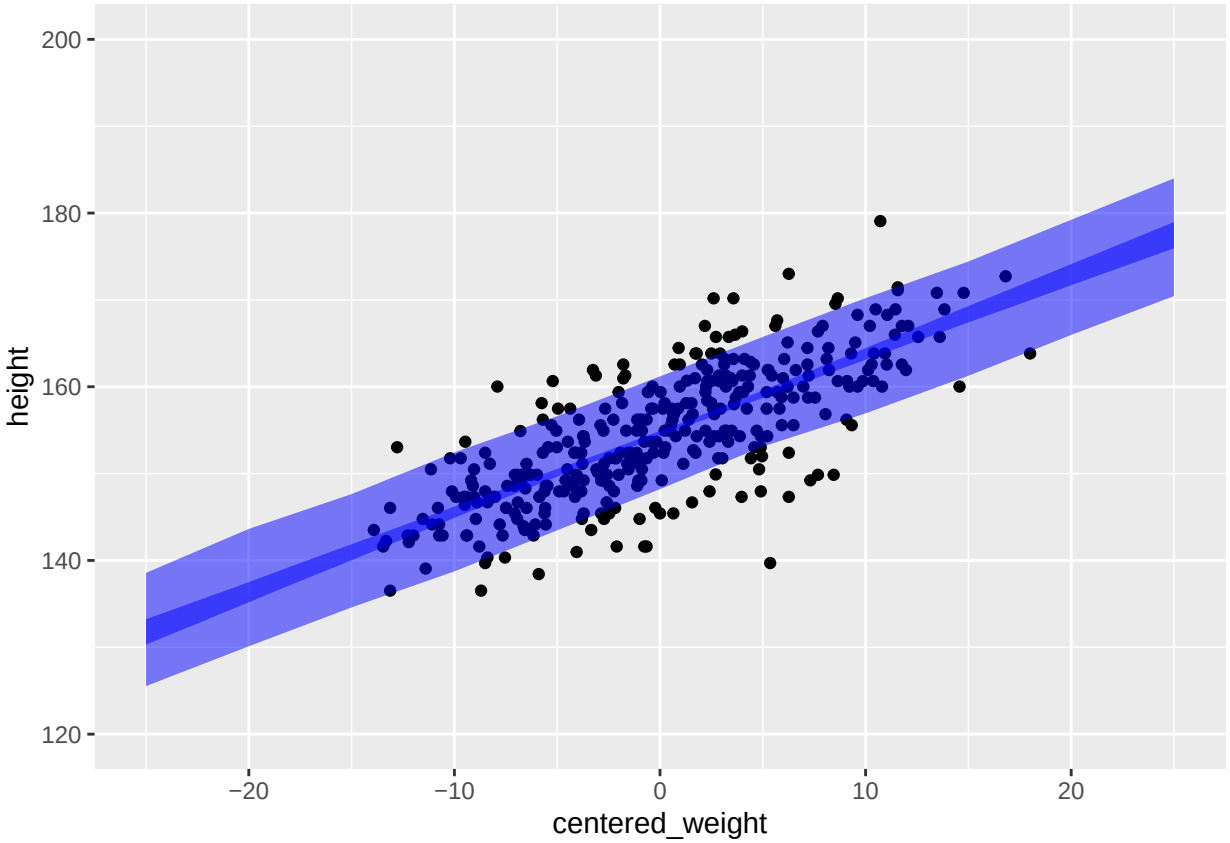



Now do the same but using `sim_df` to simulate data:

```
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. The average value for height, for each weight, is contained in the dataframe samples_summarized for the un-centered model and samples_summarized_centered for the centered model. In order to plot these two results on the same graph we need to shift their x-axis values. Try and produce a graph with the mean heights as a function of weight for both models- you should see that they are virtually identical.

Produce the summarized version of the posterior for the centered model, and add a column to record which model this is from.

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

We already have the summarized table for the regular model. We need to add a column to tell us which model it is from.

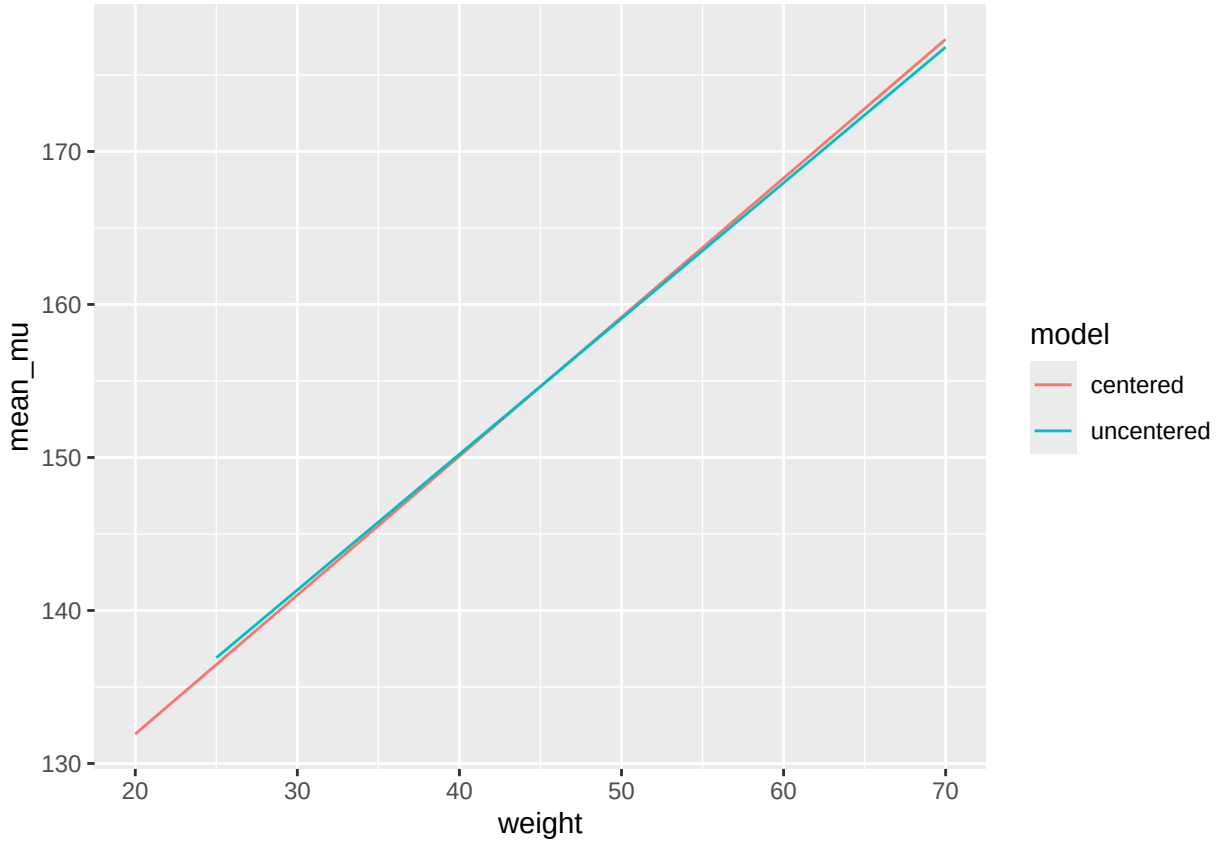
```
samples_summarized <- mutate(samples_summarized,model="uncentered")
```

Combine the two datasets by using all of the rows from each dataframe.

```
combinded_summary <- bind_rows(samples_summarized_centered,samples_summarized)
```

Plot them on the same graph, using color to distinguish between the two models.

```
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
```