Regression Models for Epidemiology

Ezra Morrison

2023-07-06

Table of contents

Preface

This web-book is derived from my lecture slides for the Spring 2023 session of Epidemiology 204: "Quantitative Epidemiology III: Statistical Models", at UC Davis.

I have drawn these materials from many sources, including but not limited to:

- https://www.taylorfrancis.com/books/mono/10.1201/9781315182780/introduction-generalized-linear-models-adrian-barnett-annette-dobson
- https://dmrocke.ucdavis.edu/Class/EPI204-Spring-2021/EPI204-Spring-2021.html
- https://link.springer.com/book/10.1007/978-1-4614-1353-0

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

2 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

3 Linear (Gaussian) Models

[This content is adapted from Dobson & Barnett, An Introduction to Generalized Linear Models, 4th edition, Chapters 2-6]

Functions from these packages will be used throughout this document:

```r
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(dplyr) # manipulate data
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(conflicted) # check for conflicting function definitions
```

4 Understanding Gaussian Linear Regression Models

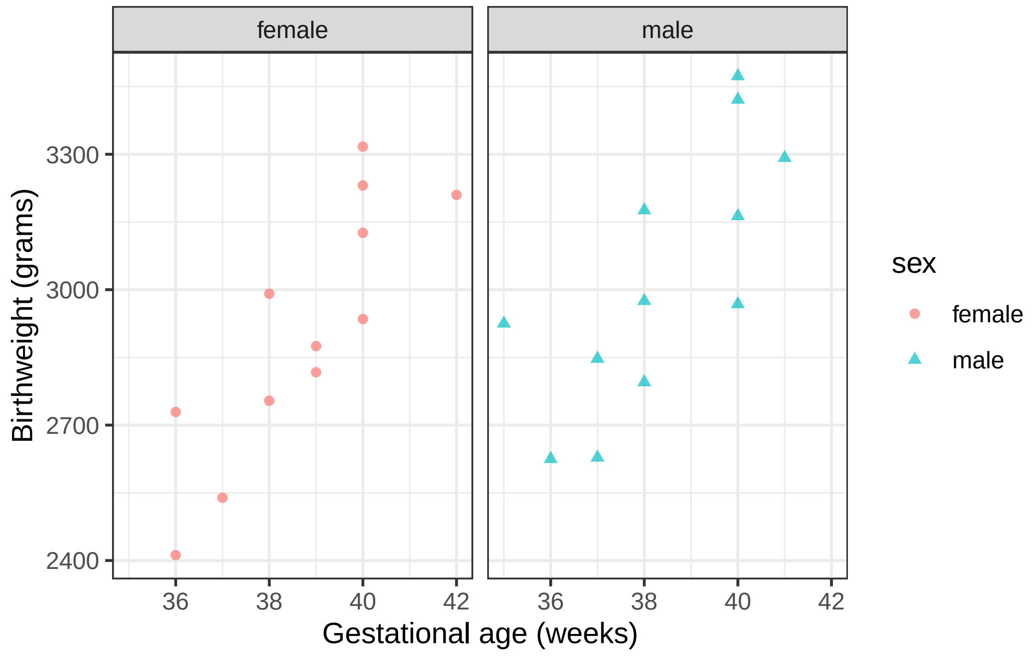
4.1 Motivating example: birthweights and gestational age

Suppose we want to learn about the distributions of birthweights for (human) babies born at different gestational ages and with different chromosomal sexes (Dobson and Barnett, Example 2.2.2):

```r
data("birthweight", package = "dobson")

bw =
  birthweight |>
  pivot_longer(
    cols = everything(),
    names_to = c("sex", ".value"),
    names_sep = "s "
  ) |>
  mutate(
    sex = ifelse(sex == "boy", "male", "female"),
    male = (sex == "male") |> as.integer())  |>
  rename(age = `gestational age`)
```

```r
plot1 = bw |>
  ggplot(aes(
    x = age,
    y = weight,
    linetype = sex,
    shape = sex,
    col = sex))  +
  theme_bw() +
  xlab("Gestational age (weeks)") +
  ylab("Birthweight (grams)") +
  # expand_limits(y = 0, x = 0) +
  geom_point(alpha = .7)
```

```
print(plot1 + facet_wrap(~ sex))
```



4.2 Parallel lines regression

We don't have enough data to model the distribution of birth weight separately for each combination of gestational age and sex, so let's instead consider a (relatively) simple model for how that distribution varies with gestational age and sex.

4.2.1 Notation

Let:

- Y represent birthweight (measured in grams)
- X_1 represent chromosomal sex:
  - $X_1 = 0$ if female (XX)
  - $X_1 = 1$ if male (XY)
- X_2 represent gestational age at birth (measured in weeks).

Note

Female is the **reference level** for the categorical variable X_1 (chromosomal sex). The choice of a reference level is arbitrary and does not limit what we can do with the resulting model; it only makes it more computationally convenient to make inferences about comparisons involving that reference group.

Now, consider the following model:

$$Y \sim N(\mu(X_1, X_2), \sigma^2)$$

$$\mu(X_1, X_2) \overset{\text{def}}{=} E[Y|X_1, X_2] = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

4.2.2 Implementing the Model in R

Here's how we can implement this model in R:

```r
bw_lm1 = lm(
  formula = weight ~ sex + age,
  data = bw)

bw_lm1 |>
  parameters(show_sigma = TRUE) |>
  print_md()
```

Parameter	Coefficient	SE	95% CI	t(21)	p
(Intercept)	-1773.32	794.59	(-3425.75, -120.89)	-2.23	0.037
sex (male)	163.04	72.81	(11.63, 314.45)	2.24	0.036
age	120.89	20.46	(78.34, 163.45)	5.91	< .001

Here's how this model looks, superimposed on the data:

```r
bw =
  bw |>
  mutate(`E[Y|X=x]` = fitted(bw_lm1)) |>
  arrange(sex, age)

plot2 =
  plot1 %+% bw +
  geom_line(aes(y = `E[Y|X=x]`))
```

```r
print(plot2)
```

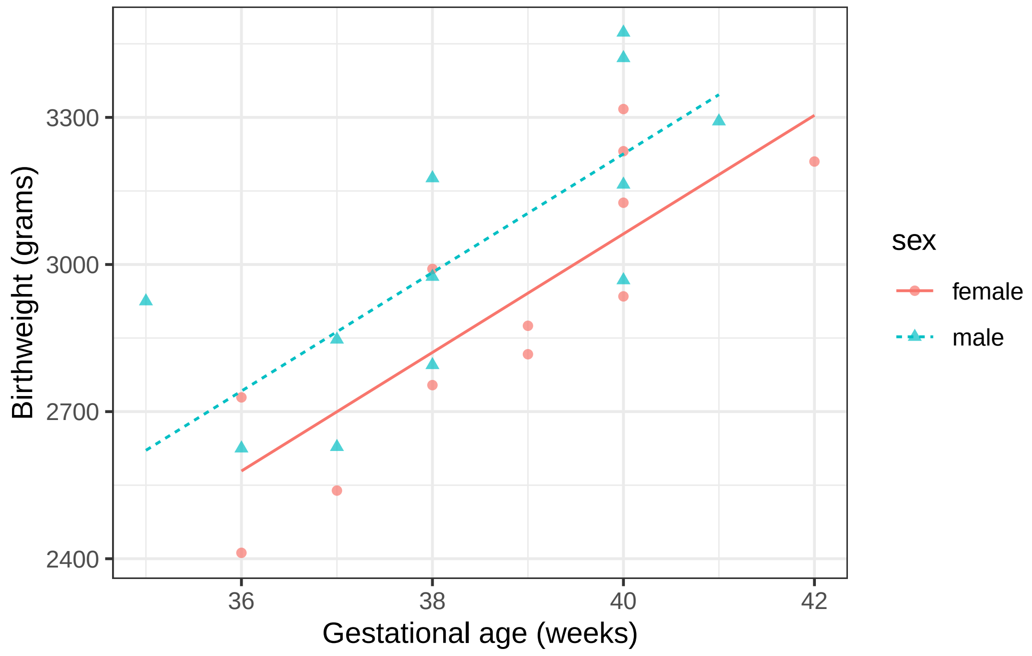


Figure 4.1: Parallel-slopes model of birthweight

4.2.3 Model assumptions and predictions

To learn what this model is assuming, let's plug in a few values.

According to this model, what's the mean birthweight for a female born at 36 weeks?

```r
pred_female = coef(bw_lm1)["(Intercept)"] + coef(bw_lm1)["age"]*36

# print(pred_female)
## built-in prediction:
# predict(bw_lm1, newdata = tibble(sex = "female", age = 36))
```

$$E[Y|X_1 = 0, X_2 = 36] = \beta_0 + \beta_1 \cdot 0 + \beta_2 \cdot 36 = 2578.8739$$

What's the mean birthweight for a male born at 36 weeks?

```r
pred_male =
  coef(bw_lm1)["(Intercept)"] +
  coef(bw_lm1)["sexmale"] +
  coef(bw_lm1)["age"]*36
```

$$E[Y|X_1 = 1, X_2 = 36] = \beta_0 + \beta_1 \cdot 1 + \beta_2 \cdot 36 = 2741.9132$$

What's the difference in mean birthweights between males born at 36 weeks and females born at 36 weeks?

$$\begin{aligned}
&E[Y|X_1 = 1, X_2 = 36] - E[Y|X_1 = 0, X_2 = 36]\\
&= 2741.9132 - 2578.8739\\
&= 163.0393
\end{aligned}$$

Shortcut:

$$\begin{aligned}
&E[Y|X_1 = 1, X_2 = 36] - E[Y|X_1 = 0, X_2 = 36]\\
&= (\beta_0 + \beta_1 \cdot 1 + \beta_2 \cdot 36) - (\beta_0 + \beta_1 \cdot 0 + \beta_2 \cdot 36)\\
&= \beta_1\\
&= 163.0393
\end{aligned}$$

Note that age doesn't show up in this difference: in other words, according to this model, the difference between females and males with the same gestational age is the same for every age.

That's an assumption of the model; it's built-in to the parametric structure, even before we plug in the estimated values of those parameters.

That's why the lines are parallel.

4.3 Interactions

What if we don't like that parallel lines assumption?

Then we need to allow an "interaction" between age and sex:

$$E[Y|X_1, X_2] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 (X_1 \cdot X_2)$$

```r
bw_lm2 = lm(weight ~ sex + age + sex:age, data = bw)
```

Here are the estimated parameters (βs):

```r
bw_lm2 |>
  parameters() |>
  print_md()
```

Parameter	Coefficient	SE	95% CI	t(20)	p
(Intercept)	-2141.67	1163.60	(-4568.90, 285.56)	-1.84	0.081
sex (male)	872.99	1611.33	(-2488.18, 4234.17)	0.54	0.594
age	130.40	30.00	(67.82, 192.98)	4.35	$< .001$
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

Here's another way we could rewrite this model (by collecting terms involving X_2):

$$E[Y|X_1, X_2] = \beta_0 + \beta_1 X_1 + (\beta_2 + \beta_3 X_1) X_2$$

Note

If you want to understand a coefficient in a model with interactions, collect terms for the corresponding variable, and you will see what other variables are interacting with the variable you are interested in.

In this case, the coefficient X_2 is interacting with X_1. So the slope of Y with respect to X_2 depends on the value of X_2.

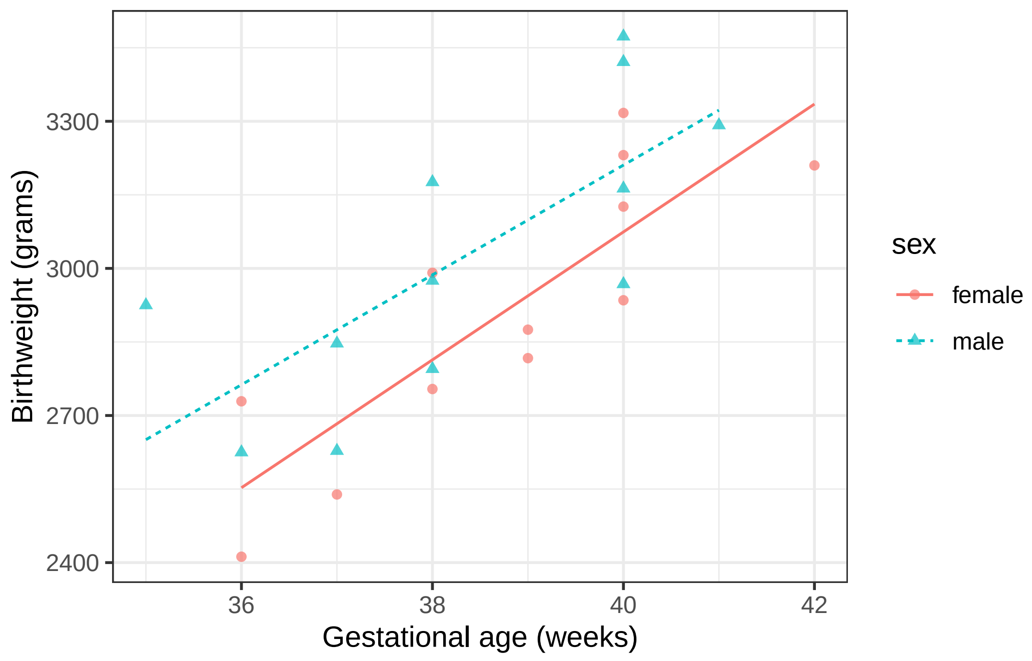
There is no longer a slope; we can't talk about "the slope of birthweight with respect to age". We can only talk about "the slope of birthweight with respect to age among males" and "the slope of birthweight with respect to age among females".

Then: that coefficient is the difference in means per unit change in its corresponding coefficient, when the other collected variables are set to 0.

Here's how this model looks, superimposed on the data:

```r
bw =
  bw |>
  mutate(
    predlm2 = predict(bw_lm2)
  ) |>
  arrange(sex, age)

plot1_interact =
  plot1 %+% bw +
  geom_line(aes(y = predlm2))
```

```
print(plot1_interact)
```



Now we can see that the lines aren't parallel.

To learn what this model is assuming, let's plug in a few values.

According to this model, what's the mean birthweight for a female born at 36 weeks?

```
pred_female = coef(bw_lm2)["(Intercept)"] + coef(bw_lm2)["age"]*36
```

$$E[Y|X_1 = 0, X_2 = 36] = \beta_0 + \beta_1 \cdot 0 + \beta_2 \cdot 36 + \beta_3 \cdot (0 * 36) = 2552.7333$$

What's the mean birthweight for a male born at 36 weeks?

```
pred_male =
  coef(bw_lm2)["(Intercept)"] +
  coef(bw_lm2)["sexmale"] +
  coef(bw_lm2)["age"]*36 +
  coef(bw_lm2)["sexmale:age"] * 36
```

$$E[Y|X_1 = 0, X_2 = 36] = \beta_0 + \beta_1 \cdot 1 + \beta_2 \cdot 36 + \beta_3 \cdot 1 \cdot 36 = 2762.7069$$

What's the difference in mean birthweights between males born at 36 weeks and females born at 36 weeks?

$$\begin{aligned}
&E[Y|X_1 = 1, X_2 = 36] - E[Y|X_1 = 0, X_2 = 36]\\
&= (\beta_0 + \beta_1 \cdot 1 + \beta_2 \cdot 36 + \beta_3 \cdot 1 \cdot 36)\\
&\quad - (\beta_0 + \beta_1 \cdot 0 + \beta_2 \cdot 36 + \beta_3 \cdot 0 \cdot 36)\\
&= \beta_2 + \beta_3 \cdot 36\\
&= 209.9736
\end{aligned}$$

Note that age now does show up in the difference: in other words, according to this model, the difference in mean birthweights between females and males with the same gestational age can vary by gestational age.

That's how the lines in the graph ended up non-parallel.

4.4 Stratified regression

We could re-write the interaction model as a stratified model, with a slope and intercept for each sex:

```
bw_lm_strat =
  bw |>
  lm(
    formula = weight ~ sex + sex:age - 1,
    data = _)

bw_lm_strat |>
  parameters() |>
  print_md()
```

Parameter	Coefficient	SE	95% CI	t(20)	p
sex (female)	-2141.67	1163.60	(-4568.90, 285.56)	-1.84	0.081
sex (male)	-1268.67	1114.64	(-3593.77, 1056.42)	-1.14	0.268
sex (female) × age	130.40	30.00	(67.82, 192.98)	4.35	$< .001$
sex (male) × age	111.98	29.05	(51.39, 172.57)	3.86	$< .001$

4.5 Curved-line regression

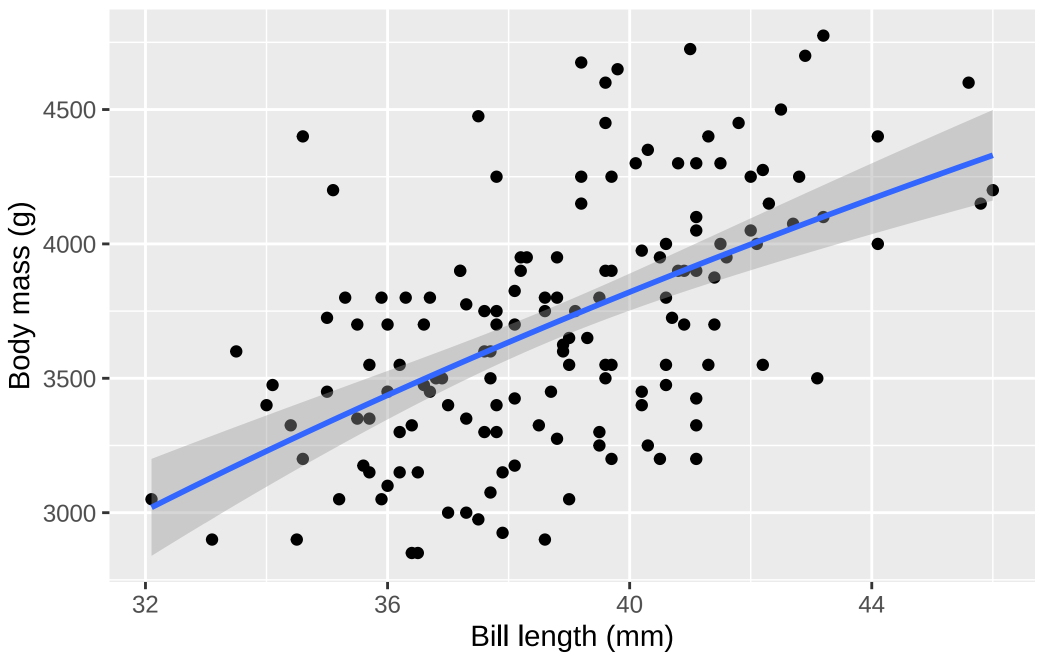
If we transform some of our covariates (Xs) and plot the resulting model on the original covariate scale, we end up with curved regression lines:

```r
bw_lm3 = lm(weight ~ sex:log(age) - 1, data = bw)
library(palmerpenguins)

ggpenguins <-
  palmerpenguins::penguins |>
  dplyr::filter(species == "Adelie") |>
  ggplot(
    aes(x = bill_length_mm , y = body_mass_g)) +
  geom_point() +
  xlab("Bill length (mm)") +
  ylab("Body mass (g)")

ggpenguins2 = ggpenguins +
  stat_smooth(
    method = "lm",
              formula = y ~ log(x),
              geom = "smooth") +
  xlab("Bill length (mm)") +
  ylab("Body mass (g)")
```

```r
ggpenguins2 |> print()
```

```
Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
```

```
Warning: Removed 1 rows containing missing values (`geom_point()`).
```


Body mass (g)
4500
4000
3500
3000
32
36
40
44
Bill length (mm)

5 Estimating Linear Models via Maximum Likelihood

5.1 Review of one-sample inference

Previously, we learned how to fit outcome-only models of the form $p(X = x|\theta)$ to iid data $\mathbf{x} = (x_1, ..., x_n)$ using maximum likelihood estimation:

$$\mathcal{L}(\mathbf{x}|\theta) = p(X_1 = x_1, ..., X_n = x_n|\theta) = \prod_{i=1}^{n} p(X = x_i|\theta)$$

$$\ell(x|\theta) = \log \mathcal{L}(x|\theta)$$

$$\hat{\theta}_{ML} = \arg\max_{\theta} \ell(x|\theta)$$

We learned how to quantify our uncertainty about these maximum likelihood estimates; with sufficient sample size, $\hat{\theta}_{ML}$ has the approximate distribution:

$$\hat{\theta}_{ML} \dot{\sim} N(\theta, \mathcal{I}(\theta)^{-1})$$

For models in the "exponential family" of distributions, which includes the Gaussian, Poisson, Bernoulli, Binomial, Exponential, and Gamma distributions, $\mathcal{I}(\theta) = \text{-}E[l''(X|\theta)]$, so we estimated $\mathcal{I}(\theta)$ using either $\mathcal{I}(\theta)|_{\theta=\hat{\theta}_{ML}}$ or $l''(\mathbf{x}|\theta)|_{\theta=\hat{\theta}_{ML}}$.

Then an asymptotic approximation of a 95% confidence interval for θ_k is

$$\hat{\theta}_{ML} \pm z_{0.975} \times \left[\left(\hat{\mathcal{I}}(\hat{\theta}_{ML})\right)^{-1}\right]_{kk}$$

where z_β the β quantile of the standard Gaussian distribution.

5.2 MLEs for Linear Regression

Let's use maximum likelihood again:

$$\mathcal{L}(\mathbf{y}|\mathbf{x},\beta,\sigma^2) = \prod_{i=1}^{n}(2\pi\sigma^2)^{-1/2}\exp\left\{-\frac{1}{2\sigma^2}(y_i - x_i'\beta)^2\right\}$$

$$\ell(\mathbf{y}|\mathbf{x},\beta,\sigma^2) \propto -\frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - x_i'\beta)^2$$

$$\ell'(\mathbf{y}|\mathbf{x},\beta,\sigma^2) \propto -\frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\frac{d}{d\beta}\left(\sum_{i=1}^{n}(y_i - x_i'\beta)^2\right)$$

A few tools from linear algebra will make this analysis go easier (see the recommended text by Fieller, Section 7.2 for details).

$$f_\beta(\mathbf{x}) = (f_\beta(x_1), f_\beta(x_2), ..., f_\beta(x_n))^\top$$

Let $\mathbf{x}$ and β be vectors of length p, or in other words, matrices of length $p \times 1$:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \; \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$$

Then

$$x' \equiv x^\top \equiv [x_1, x_2, ..., x_p]$$

and

$$x'\beta = [x_1, x_2, ..., x_p]\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} = x_1\beta_1 + x_2\beta_2 + ... + x_p\beta_p$$

If $f(\beta)$ is a function that takes β as input and outputs a scalar, such as $f(\beta) = x'\beta$, then:

$$\frac{d}{d\beta}f(\beta) = \begin{bmatrix} \frac{d}{d\beta_1}f(\beta) \\ \frac{d}{d\beta_2}f(\beta) \\ \vdots \\ \frac{d}{d\beta_p}f(\beta) \end{bmatrix}$$

In particular, if $f(\beta) = x'\beta$, then:

$$\frac{d}{d\beta}x'\beta = \begin{bmatrix} \frac{d}{d\beta_1}(x_1\beta_1 + x_2\beta_2 + ... + x_p\beta_p) \\ \frac{d}{d\beta_2}(x_1\beta_1 + x_2\beta_2 + ... + x_p\beta_p) \\ \vdots \\ \frac{d}{d\beta_p}(x_1\beta_1 + x_2\beta_2 + ... + x_p\beta_p) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \mathbf{x}$$

In general:

$$\frac{d}{d\beta}x'\beta = x$$

This looks a lot like non-vector calculus, except that you have to transpose the coefficient.

Similarly,

$$\frac{d}{d\beta}\beta'\beta = 2\beta$$

This is like taking the derivative of x^2.

And finally, if S is a $p \times p$ matrix, then:

$$\frac{d}{d\beta}\beta' S\beta = 2S\beta$$

Again, this is like taking the derivative of cx^2 with respect to c in non-vector calculus.

Thus:

$$\sum_{i=1}^{n}(y_i - f_\beta(x_i))^2 = (\mathbf{y} - X\beta)'(\mathbf{y} - X\beta)$$

$$(\mathbf{y} - X\beta)' = (\mathbf{y}' - (X\beta)') = (\mathbf{y}' - \beta' X')$$

So

$$\begin{aligned}(\mathbf{y}-X\beta)'(\mathbf{y}-X\beta) &= (\mathbf{y}'-\beta'X')(\mathbf{y}-X\beta)\\ &= y'y-\beta'X'y-y'X\beta+\beta'X'X\beta\\ &= y'y-2y'X\beta+\beta'X'X\beta\end{aligned}$$

So

$$\begin{aligned}\frac{d}{d\beta}\left(\sum_{i=1}^{n}(y_i-x_i'\beta)^2\right) &= \frac{d}{d\beta}(\mathbf{y}-X\beta)'(\mathbf{y}-X\beta)\\ &= \frac{d}{d\beta}(y'y-2y'X\beta+\beta'X'X\beta)\\ &= (-2X'y+2X'X\beta)\end{aligned}$$

So if $\ell(\beta,\sigma^2)=0$, then

$$\begin{aligned}0 &= (-2X'y+2X'X\beta)\\ 2X'y &= 2X'X\beta\\ X'y &= X'X\beta\\ (X'X)^{-1}X'y &= \beta\end{aligned}$$

The second derivative matrix $\ell''_{\beta,\beta'}(\beta,\sigma^2;\mathbf{X},\mathbf{y})$ is negative definite at $\beta=(X'X)^{-1}X'y$, so $\hat{\beta}_{ML}=(X'X)^{-1}X'y$ is the MLE for β.

Similarly (not shown):

$$\hat{\sigma}^2_{ML}=\frac{1}{n}(Y-X\hat{\beta})'(Y-X\hat{\beta})$$

And

$$\begin{aligned}\mathcal{I}_\beta &= E[-\ell''_{\beta,\beta'}(Y|X,\beta,\sigma^2)]\\ &= \frac{1}{\sigma^2}X'X\end{aligned}$$

So:

$$Var(\hat{\beta})\approx(\mathcal{I}_\beta)^{-1}=\sigma^2(X'X)^{-1}$$

and

$$\hat{\beta} \dot{\sim} N(\beta, \mathcal{I}_{\beta}^{-1})$$

These are all results you have hopefully seen before, and in the Gaussian linear regression case they are exact, not just approximate.

In our model 2 above, this matrix is:

```r
bw_lm2 |> vcov()
```

```
            (Intercept)  sexmale      age sexmale:age
(Intercept)     1353968 -1353968 -34871.0     34871.0
sexmale        -1353968  2596387  34871.0    -67211.0
age              -34871    34871    899.9      -899.9
sexmale:age       34871   -67211   -899.9      1743.5
```

Note that if we take the square roots of the diagonals, we get the standard errors listed in the model output:

```r
bw_lm2 |> vcov() |> diag() |> sqrt()
```

```
(Intercept)     sexmale         age sexmale:age
    1163.60     1611.33       30.00       41.76
```

```r
bw_lm2 |> summary() |> coef() |> pander()
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-2142	1164	-1.841	0.08057
sexmale	873	1611	0.5418	0.594
age	130.4	30	4.347	0.0003127
sexmale:age	-18.42	41.76	-0.4411	0.6639

So we can do confidence intervals, hypothesis tests, and p-values exactly as in the one-variable case we looked at previously.

6 Inference about Gaussian Linear Regression Models

Research question: is there really an interaction between sex and age?

$H_0 : \beta_3 = 0$

$H_A : \beta_3 \neq 0$

$P(|\hat{\beta}_3| > |-18.4172| \mid H_0) = ?$

6.1 Wald tests and CIs

R can give you Wald tests for single coefficients and corresponding CIs:

```
bw_lm2 |>
  parameters(ci_method = "wald") |>
  print_md()
```

Parameter	Coefficient	SE	95% CI	t(20)	p
(Intercept)	-2141.67	1163.60	(-4568.90, 285.56)	-1.84	0.081
sex (male)	872.99	1611.33	(-2488.18, 4234.17)	0.54	0.594
age	130.40	30.00	(67.82, 192.98)	4.35	$< .001$
sex (male) $\times$ age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

6.2 One-parameter inference by hand

To understand what's happening, let's replicate these results by hand for the interaction term.

6.2.1 P-value

```r
beta_hat = coef(summary(bw_lm2))["sexmale:age", "Estimate"]
se_hat = coef(summary(bw_lm2))["sexmale:age", "Std. Error"]
dfresid = bw_lm2$df.residual
t_stat = abs(beta_hat)/se_hat
pval_t = pt(abs(t_stat), df = dfresid, lower = FALSE) * 2
```

$$
\begin{aligned}
P\left(|\hat{\beta}_3| > |-18.4172| \middle| H_0\right) &= P\left(\left|\frac{\hat{\beta}_3}{\hat{SE}(\hat{\beta}_3)}\right| > \left|\frac{-18.4172}{41.7558}\right| \middle| H_0\right) \\
= P\left(|T_{20}| > 0.4411 | H_0\right) \\
= 0.6639
\end{aligned}
$$

This matches the result in the table above.

6.2.2 Confidence interval

```r
confint_radius_t = se_hat * qt(p = 0.975, df = dfresid, lower = TRUE)
confint_t = beta_hat + c(-1,1) * confint_radius_t
print(confint_t)
```

```
[1] -105.52   68.68
```

This also matches.

6.3 Gaussian approximations

Here are the asymptotic (Gaussian approximation) equivalents:

6.3.1 P-value

```r
pval_z = pnorm(abs(t_stat), lower = FALSE) * 2

print(pval_z)
```

```
[1] 0.6592
```

6.3.2 Confidence interval

```r
confint_radius_z = se_hat * qnorm(0.975, lower = TRUE)
confint_z =
  beta_hat + c(-1,1) * confint_radius_z
print(confint_z)
```

```
[1] -100.26   63.42
```

6.4 Likelihood ratio statistics

```r
logLik(bw_lm2)
```

```
'log Lik.' -156.6 (df=5)
```

```r
logLik(bw_lm1)
```

```
'log Lik.' -156.7 (df=4)
```

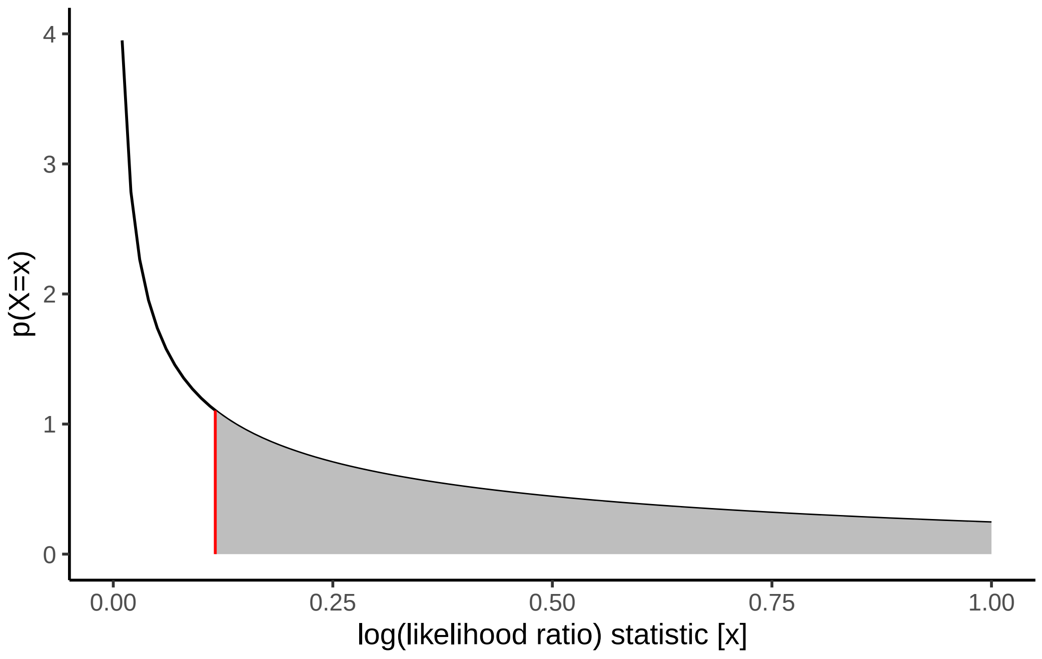
```r
lLR = (logLik(bw_lm2) - logLik(bw_lm1)) |> as.numeric()
delta_df = (bw_lm1$df.residual - df.residual(bw_lm2))

d_lLR = function(x, df = delta_df) dchisq(x, df = df)

x_max = 1

chisq_plot =
  ggplot() +
  geom_function(fun = d_lLR) +
  stat_function( fun = d_lLR, xlim = c(lLR, x_max), geom = "area", fill = "gray") +
  geom_segment(aes(x = lLR, xend = lLR, y = 0, yend = d_lLR(lLR)), col = "red") +
  xlim(0.0001,x_max) +
  ylim(0,4) +
  ylab("p(X=x)") +
  xlab("log(likelihood ratio) statistic [x]") +
  theme_classic()
```

```
pchisq(
  q = 2*lLR,
  df = delta_df,
  lower = FALSE)
```

```
[1] 0.6298
```

```
chisq_plot |> print()
```

```
Warning: Removed 1 row containing missing values (`geom_function()`).
```



Now we can get the p-value:

```
pchisq(2*lLR, df = delta_df, lower = FALSE)
```

```
[1] 0.6298
```

6.5

In practice you don't have to do this by hand; there are functions to do it for you:

```
# built in
library(lmtest)
lrtest(bw_lm2, bw_lm1)
```

```
Likelihood ratio test

Model 1: weight ~ sex + age + sex:age
Model 2: weight ~ sex + age
  #Df LogLik Df Chisq Pr(>Chisq)
1   5   -157
2   4   -157 -1  0.23       0.63
```

7 Goodness of fit

7.1 AIC and BIC

When we use likelihood ratio tests, we are comparing how well different models fit the data.

Likelihood ratio tests require "nested" models: one must be a special case of the other.

If we have non-nested models, we can instead use the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC):

- $\text{AIC} = -2 * \ell(\hat{\theta}) + 2 * p$
- $\text{BIC} = -2 * \ell(\hat{\theta}) + p * \log(n)$

where ℓ is the log-likelihood of the data evaluated using the parameter estimates $\hat{\theta}$, p is the number of estimated parameters in the model (including $\hat{\sigma}^2$), and n is the number of observations.

You can calculate these criteria using the `logLik()` function, or use the built-in R functions:

7.1.1 AIC in R

```r
-2 * logLik(bw_lm2) |> as.numeric() +
  2*(length(coef(bw_lm2))+1) # sigma counts as a parameter here
```

```
[1] 323.2
```

```r
AIC(bw_lm2)
```

```
[1] 323.2
```

7.1.2 BIC in R

```r
-2 * logLik(bw_lm2) |> as.numeric() +
  (length(coef(bw_lm2))+1) * log(nobs(bw_lm2))
```

```
[1] 329
```

```r
BIC(bw_lm2)
```

```
[1] 329
```

Large values of AIC and BIC are worse than small values. There are no hypothesis tests or p-values associated with these criteria.

7.2 (Residual) Deviance

Let q be the number of distinct covariate combinations in a data set.

```r
bw.X.unique =
  bw |>
  count(sex, age)

n_unique.bw  = nrow(bw.X.unique)
```

For example, in the `birthweight` data, there are $q = 12$ unique patterns:

```r
bw.X.unique |>
  pander(
    row.names = rownames(bw.X.unique))
```

	sex	age	n
1	female	36	2
2	female	37	1
3	female	38	2
4	female	39	2
5	female	40	4
6	female	42	1

	sex	age	n
7	male	35	1
8	male	36	1
9	male	37	2
10	male	38	3
11	male	40	4
12	male	41	1

Note

If a given covariate pattern has more than one observation in a dataset, those observations are called **replicates**.

Then the most complicated model we could fit would have one parameter (a mean) for each of these combinations, plus a variance parameter:

```
lm_max =
  bw |>
  mutate(age = factor(age)) |>
  lm(
    formula = weight ~ sex:age - 1,
    data = _)

lm_max |>
  parameters() |>
  print_md()
```

Parameter	Coefficient	SE	95% CI	t(12)	p
sex (male) × age35	2925.00	187.92	(2515.55, 3334.45)	15.56	< .001
sex (female) × age36	2570.50	132.88	(2280.98, 2860.02)	19.34	< .001
sex (male) × age36	2625.00	187.92	(2215.55, 3034.45)	13.97	< .001
sex (female) × age37	2539.00	187.92	(2129.55, 2948.45)	13.51	< .001
sex (male) × age37	2737.50	132.88	(2447.98, 3027.02)	20.60	< .001
sex (female) × age38	2872.50	132.88	(2582.98, 3162.02)	21.62	< .001
sex (male) × age38	2982.00	108.50	(2745.60, 3218.40)	27.48	< .001
sex (female) × age39	2846.00	132.88	(2556.48, 3135.52)	21.42	< .001
sex (female) × age40	3152.25	93.96	(2947.52, 3356.98)	33.55	< .001
sex (male) × age40	3256.25	93.96	(3051.52, 3460.98)	34.66	< .001
sex (male) × age41	3292.00	187.92	(2882.55, 3701.45)	17.52	< .001
sex (female) × age42	3210.00	187.92	(2800.55, 3619.45)	17.08	< .001

We call this model the **full**, **maximal**, or **saturated** model for this dataset.

We can calculate the log-likelihood of this model as usual:

```r
logLik(lm_max)
```

```
'log Lik.' -151.4 (df=13)
```

We can compare this model to our other models using chi-square tests, as usual:

```r
lrtest(lm_max, bw_lm2)
```

```
Likelihood ratio test

Model 1: weight ~ sex:age - 1
Model 2: weight ~ sex + age + sex:age
  #Df LogLik Df Chisq Pr(>Chisq)
1  13   -151
2   5   -157 -8  10.4       0.24
```

The likelihood ratio statistic for this test is

$$\lambda = 2 * (\ell_{\text{full}} - \ell) = 10.3554$$

where:

- ℓ_{max} is the log-likelihood of the full model: -151.4016
- ℓ is the log-likelihood of our comparison model (two slopes, two intercepts): -156.5793

This statistic is called the **deviance** or **residual deviance** for our two-slopes and two-intercepts model; it tells us how much the likelihood of that model deviates from the likelihood of the maximal model.

The corresponding p-value tells us whether there we have enough evidence to detect that our two-slopes, two-intercepts model is a worse fit for the data than the maximal model; in other words, it tells us if there's evidence that we missed any important patterns. (Remember, a nonsignificant p-value could mean that we didn't miss anything and a more complicated model is unnecessary, or it could mean we just don't have enough data to tell the difference between these models.)

7.3 Null Deviance

Similarly, the *least* complicated model we could fit would have only one mean parameter, an intercept:

$$\mathrm{E}[Y|X = x] = \beta_0$$

We can fit this model in R like so:

```r
lm0 = lm(weight ~ 1, data = bw)

lm0 |> parameters() |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(23)	p
(Intercept)	2967.67	57.58	(2848.56, 3086.77)	51.54	< .001

This model also has a likelihood:

```r
logLik(lm0)
```

```
'log Lik.' -169 (df=2)
```

And we can compare it to more complicated models using a likelihood ratio test:

```r
lrtest(bw_lm2, lm0)
```

```
Likelihood ratio test

Model 1: weight ~ sex + age + sex:age
Model 2: weight ~ 1
  #Df LogLik Df Chisq Pr(>Chisq)
1   5   -157
2   2   -169 -3  24.8    1.7e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The likelihood ratio statistic for the test comparing the null model to the maximal model is

$$\lambda = 2 * (\ell_{\text{full}} - \ell_0) = 35.1067$$

where:

- ℓ_0 is the log-likelihood of the null model: -168.955
- ℓ_{full} is the log-likelihood of the maximal model: -151.4016

In R, this test is:

```
lrtest(lm_max, lm0)
```

```
Likelihood ratio test

Model 1: weight ~ sex:age - 1
Model 2: weight ~ 1
  #Df LogLik  Df Chisq Pr(>Chisq)
1  13   -151
2   2   -169 -11  35.1    0.00024 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This log-likelihood ratio statistic is called the **null deviance**. It tells us whether we have enough data to detect a difference between the null and full models.

Caution

This definition is a bit different from what I said in class (4/25); then, I said the null deviance was twice the difference in log-likelihood between the model of interest and the null model. The one I have written above is the standard one, and we'll stick to the standard definition going forward.

8 Rescaling

8.1 Rescale age

```r
bw =
  bw |>
  mutate(
    `age - mean` = age - mean(age),
    `age - 36wks` = age - 36
  )

lm1c = lm(weight ~ sex + `age - 36wks`, data = bw)

lm2c = lm(weight ~ sex + `age - 36wks` + sex:`age - 36wks`, data = bw)

parameters(lm2c, ci_method = "wald") |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(20)	p
(Intercept)	2552.73	97.59	(2349.16, 2756.30)	26.16	< .001
sex (male)	209.97	129.75	(-60.68, 480.63)	1.62	0.121
age - 36wks	130.40	30.00	(67.82, 192.98)	4.35	< .001
sex (male) × age - 36wks	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

Compare with what we got without rescaling:

```r
parameters(bw_lm2, ci_method = "wald") |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(20)	p
(Intercept)	-2141.67	1163.60	(-4568.90, 285.56)	-1.84	0.081
sex (male)	872.99	1611.33	(-2488.18, 4234.17)	0.54	0.594
age	130.40	30.00	(67.82, 192.98)	4.35	< .001
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

9 Prediction

9.1 Prediction for linear models

$$\begin{aligned}\hat{Y} &= \hat{E}[Y|X=x]\\ &= x'\hat{\beta}\\ &= \hat{\beta}_0 \cdot 1 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_p x_p\end{aligned}$$

9.2 prediction in R

```
X = model.matrix(bw_lm1)
X |> as_tibble() |> print()
```

```
# A tibble: 24 x 3
   `(Intercept)` sexmale   age
           <dbl>   <dbl> <dbl>
 1             1       1    40
 2             1       0    40
 3             1       1    38
 4             1       0    36
 5             1       1    40
 6             1       0    40
 7             1       1    35
 8             1       0    38
 9             1       1    36
10             1       0    42
# i 14 more rows
```

```
print(X[1,])
```

```
(Intercept)     sexmale         age 
          1           1          40 
```

```r
print(coef(bw_lm1))
```

```
(Intercept)     sexmale         age 
    -1773.3       163.0       120.9 
```

```r
print(X[1,] * coef(bw_lm1))
```

```
(Intercept)     sexmale         age 
      -1773         163        4836 
```

```r
{X[1,] * coef(bw_lm1)} |> sum() |> print()
```

```
[1] 3225
```

```r
X %*% coef(bw_lm1) |> as.vector()
```

```
 [1] 3225 3062 2984 2579 3225 3062 2621 2821 2742 3304 2863 2942 3346 3062 3225
[16] 2700 2863 2579 2984 2821 3225 2942 2984 3062
```

```r
predict(bw_lm1)
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16 
3225 3062 2984 2579 3225 3062 2621 2821 2742 3304 2863 2942 3346 3062 3225 2700 
  17   18   19   20   21   22   23   24 
2863 2579 2984 2821 3225 2942 2984 3062 
```

```r
predict(bw_lm1, se.fit = TRUE)
```

```
$fit
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16 
3225 3062 2984 2579 3225 3062 2621 2821 2742 3304 2863 2942 3346 3062 3225 2700 
  17   18   19   20   21   22   23   24 
2863 2579 2984 2821 3225 2942 2984 3062 

$se.fit
```

```
 [1] 61.46 57.17 51.58 76.03 61.46 57.17 85.25 53.38 69.96 83.89 57.95 51.38
[13] 74.78 57.17 61.46 62.42 57.95 76.03 51.58 53.38 61.46 51.38 51.58 57.17

$df
[1] 21

$residual.scale
[1] 177.1
```

```
predict(bw_lm1, se.fit = TRUE)$fit
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16 
3225 3062 2984 2579 3225 3062 2621 2821 2742 3304 2863 2942 3346 3062 3225 2700 
  17   18   19   20   21   22   23   24 
2863 2579 2984 2821 3225 2942 2984 3062 
```

```
predict(bw_lm1, se.fit = TRUE)$se.fit
```

```
 [1] 61.46 57.17 51.58 76.03 61.46 57.17 85.25 53.38 69.96 83.89 57.95 51.38
[13] 74.78 57.17 61.46 62.42 57.95 76.03 51.58 53.38 61.46 51.38 51.58 57.17
```

```
predict(bw_lm1, se.fit = TRUE, interval = "confidence")$fit |> as_tibble()
```

```
# A tibble: 24 x 3
     fit   lwr   upr
   <dbl> <dbl> <dbl>
 1 3225. 3098. 3353.
 2 3062. 2944. 3181.
 3 2984. 2876. 3091.
 4 2579. 2421. 2737.
 5 3225. 3098. 3353.
 6 3062. 2944. 3181.
 7 2621. 2444. 2798.
 8 2821. 2710. 2932.
 9 2742. 2596. 2887.
10 3304. 3130. 3479.
# i 14 more rows
```

```
predict(bw_lm1, se.fit = TRUE, interval = "predict")$fit |> as_tibble()
```

```
Warning in predict.lm(bw_lm1, se.fit = TRUE, interval = "predict"): predictions on current da
```

```
# A tibble: 24 x 3
     fit   lwr   upr
   <dbl> <dbl> <dbl>
 1 3225. 2836. 3615.
 2 3062. 2675. 3449.
 3 2984. 2600. 3367.
 4 2579. 2178. 2980.
 5 3225. 2836. 3615.
 6 3062. 2675. 3449.
 7 2621. 2212. 3030.
 8 2821. 2436. 3205.
 9 2742. 2346. 3138.
10 3304. 2897. 3712.
# i 14 more rows
```

The warning from the last command is: "predictions on current data refer to *future* responses" (since you already know what happened to the current data, and thus don't need to predict it). You could also supply `newdata` to get predictions for new combinations of predictors that you didn't see in your original dataset. See `?predict.lm` for more.

10 Diagnostics

10.1 Residuals

10.1.1 Definitions of residuals

- Residuals:
$$e_i = y_i - \hat{E}[Y|X = x]$$
- For Gaussian models, e_i can be seen as an estimate of
$$\epsilon_i = Y_i - \mathrm{E}[Y|X = x_i]$$
- Standardized residuals:
$$r_i = \frac{e_i}{\hat{SD}(e_i)}$$
- For Gaussian models:
$$\hat{SD}(e_i) \approx \frac{e_i}{\hat{\sigma}}$$

10.1.2 Characteristics of residuals

With enough data and a correct model, the residuals will be approximately Guassian distributed, with variance σ^2, which we can estimate using $\hat{\sigma}^2$: that is:

$$e_i \dot{\sim}_{iid} N(0, \hat{\sigma}^2)$$

Hence, with enough data and a correct model, the standardized residuals will be approximately standard Gaussian; that is,

$$r_i \dot{\sim}_{iid} N(0, 1)$$

10.2 Marginal distributions of residuals

To look for problems with our model, we can check whether the residuals e_i and standardized residuals r_i look like they have the distributions that they are supposed to have, according to the model.

First, we need to compute the residuals. R makes this easy:

10.2.1 (non-standardized) residuals in R

```
resid(bw_lm2)
```

```
      1       2       3       4       5       6       7       8       9      10 
 176.27 -140.73 -144.13  -59.53  177.47 -126.93  -68.93  242.67 -139.33   51.67 
     11      12      13      14      15      16      17      18      19      20 
 156.67 -125.13  274.28 -137.71  -27.69 -246.69 -191.67  189.33  -11.67 -242.64 
     21      22      23      24 
 -47.64  262.36  210.36  -30.62 
```

```
# check by hand
bw$weight - fitted(bw_lm2)
```

```
      1       2       3       4       5       6       7       8       9      10 
 176.27 -140.73 -144.13  -59.53  177.47 -126.93  -68.93  242.67 -139.33   51.67 
     11      12      13      14      15      16      17      18      19      20 
 156.67 -125.13  274.28 -137.71  -27.69 -246.69 -191.67  189.33  -11.67 -242.64 
     21      22      23      24 
 -47.64  262.36  210.36  -30.62 
```

Success!

10.2.2 Standardized residuals in R

```
rstandard(bw_lm2)
```

```
       1        2        3        4        5        6        7        8 
 1.15982 -0.92601 -0.87479 -0.34723  1.03507 -0.73473 -0.39901  1.43752 
       9       10       11       12       13       14       15       16 
-0.82539  0.30606  0.92807 -0.87616  1.91428 -0.86559 -0.16430 -1.46376 
      17       18       19       20       21       22       23       24 
-1.11016  1.09658 -0.06761 -1.46159 -0.28696  1.58040  1.26717 -0.19805 
```

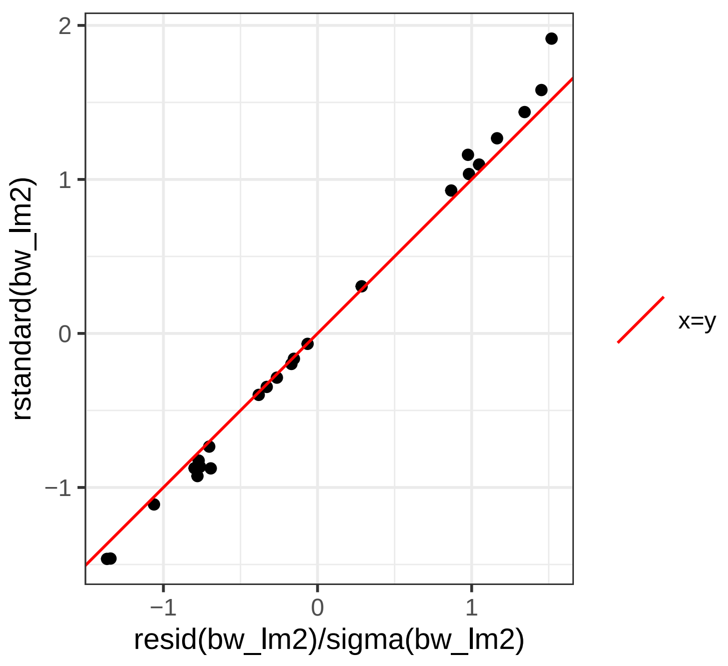
```r
resid(bw_lm2)/sigma(bw_lm2)
```

```
       1        2        3        4        5        6        7        8 
 0.97593 -0.77920 -0.79802 -0.32962  0.98258 -0.70279 -0.38166  1.34357 
       9       10       11       12       13       14       15       16 
-0.77144  0.28606  0.86741 -0.69282  1.51858 -0.76244 -0.15331 -1.36584 
      17       18       19       20       21       22       23       24 
-1.06123  1.04825 -0.06463 -1.34341 -0.26376  1.45262  1.16471 -0.16954 
```

These are not quite the same, because R is doing something more complicated and precise to get the standard errors. Let's not worry about those details for now; the difference is pretty small in this case:

```r
rstandard_compare_plot =
  tibble(
    x = resid(bw_lm2)/sigma(bw_lm2),
    y = rstandard(bw_lm2)) |>
  ggplot(aes(x = x, y = y)) +
  geom_point() +
  theme_bw() +
  coord_equal() +
  xlab("resid(bw_lm2)/sigma(bw_lm2)") +
  ylab("rstandard(bw_lm2)") +
  geom_abline(
    aes(
    intercept = 0,
    slope = 1,
    col = "x=y")) +
  labs(colour="") +
  scale_colour_manual(values="red")
```

```r
print(rstandard_compare_plot)
```



Let's add these residuals to the `tibble` of our dataset:

```
bw =
  bw |>
  mutate(
    fitted_lm2 = fitted(bw_lm2),

    resid_lm2 = resid(bw_lm2),
    # resid_lm2 = weight - fitted_lm2,

    std_resid_lm2 = rstandard(bw_lm2),
    # std_resid_lm2 = resid_lm2 / sigma(bw_lm2)
  )

bw |>
  select(
    sex,
    age,
    weight,
    fitted_lm2,
    resid_lm2,
    std_resid_lm2
  )
```

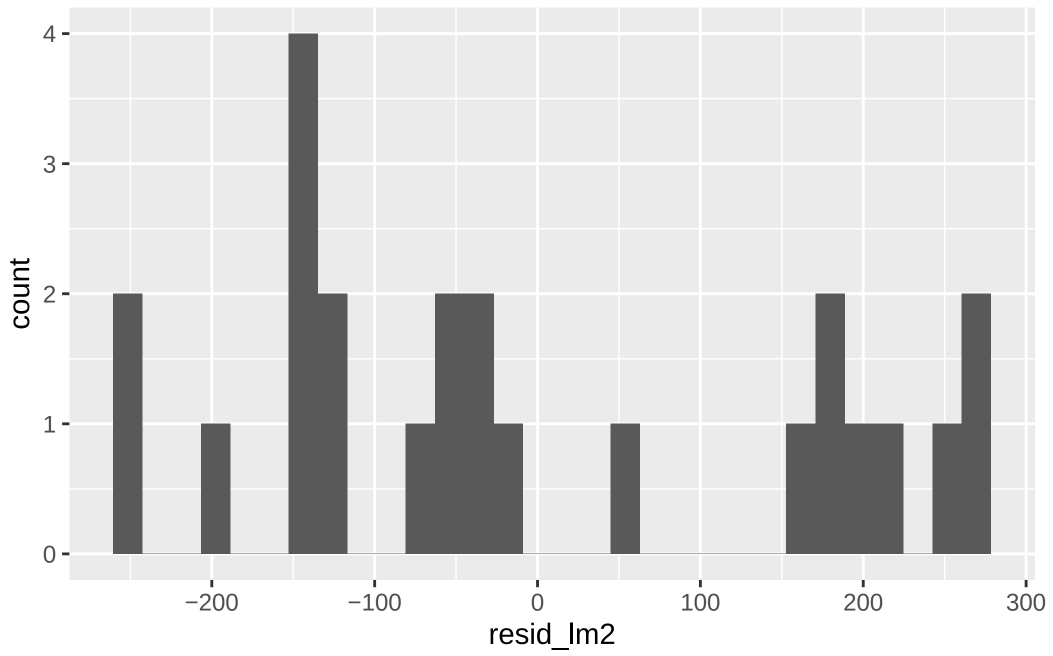
```
# A tibble: 24 x 6
   sex      age weight fitted_lm2 resid_lm2 std_resid_lm2
   <chr>  <dbl>  <dbl>      <dbl>     <dbl>         <dbl>
 1 female    36   2729      2553.     176.           1.16
 2 female    36   2412      2553.    -141.          -0.926
 3 female    37   2539      2683.    -144.          -0.875
 4 female    38   2754      2814.     -59.5         -0.347
 5 female    38   2991      2814.     177.           1.04
 6 female    39   2817      2944.    -127.          -0.735
 7 female    39   2875      2944.     -68.9         -0.399
 8 female    40   3317      3074.     243.           1.44
 9 female    40   2935      3074.    -139.          -0.825
10 female    40   3126      3074.      51.7          0.306
# i 14 more rows
```

Now let's build histograms:

10.2.3 Marginal distribution of (nonstandardized) residuals

```
resid_marginal_hist =
  bw |>
  ggplot(aes(x = resid_lm2)) +
  geom_histogram()
```

```
print(resid_marginal_hist)
```

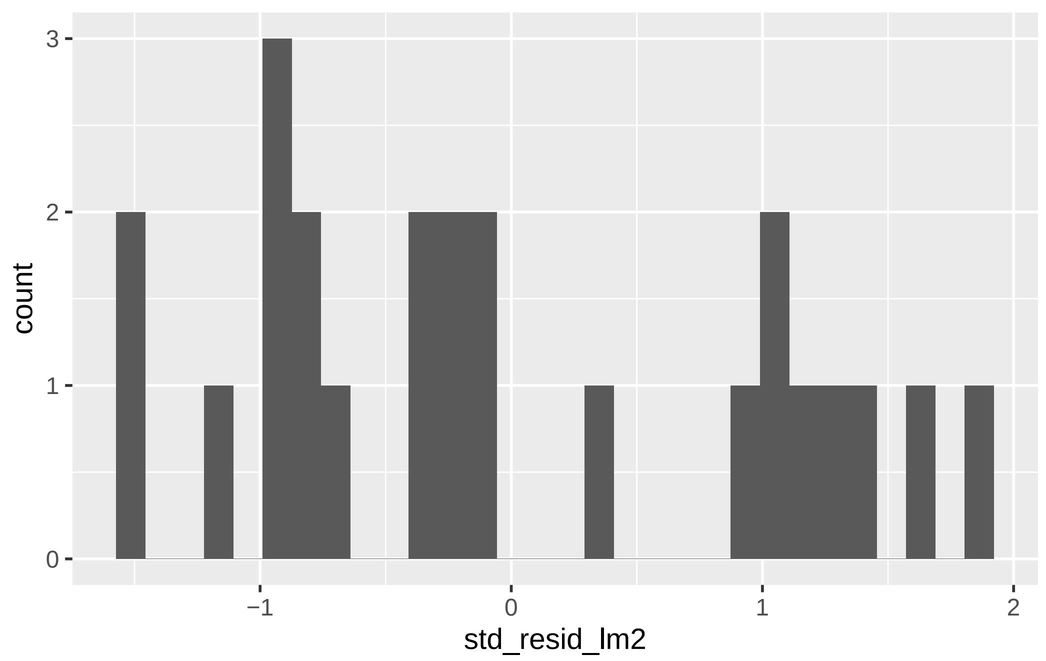


Hard to tell with this small amount of data, but I'm a bit concerned that the histogram doesn't show a bell-curve shape.

10.2.3.1 Marginal distribution of standardized residuals

```r
std_resid_marginal_hist =
  bw |>
  ggplot(aes(x = std_resid_lm2)) +
  geom_histogram()
```

```r
print(std_resid_marginal_hist)
```



This looks similar, although the scale of the x-axis got narrower, because we divided by $\hat{\sigma}$ (roughly speaking).

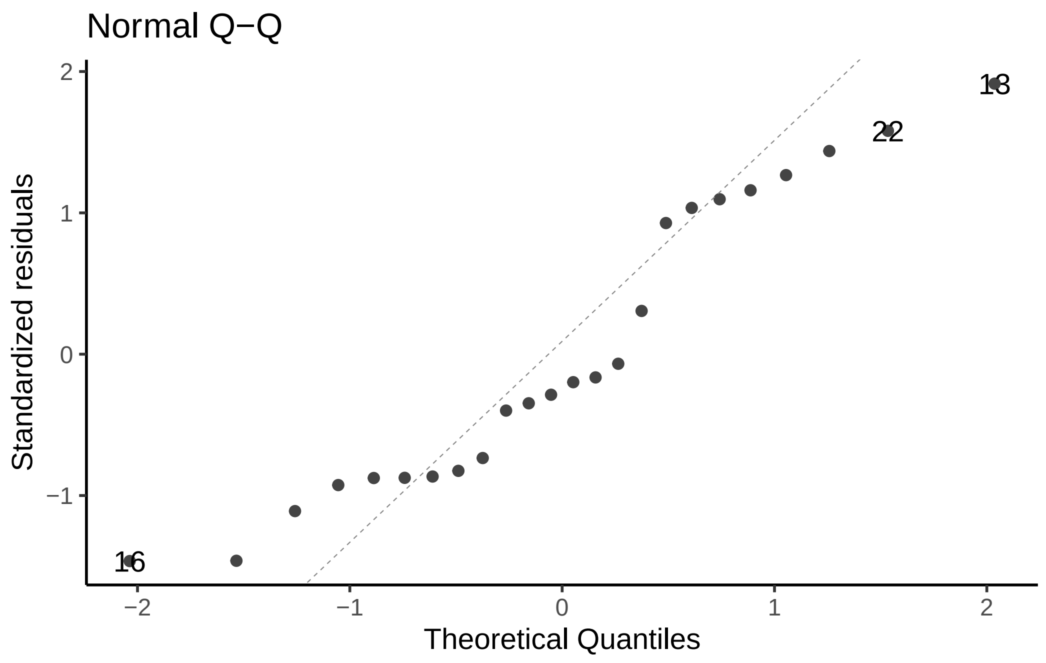
Still hard to tell if the distribution is Gaussian.

10.3 QQ plot of standardized residuals

Another way to assess normality is the QQ plot of the standardized residuals versus normal quantiles:

```r
library(ggfortify)
# needed to make ggplot2::autoplot() work for `lm` objects

qqplot_lm2_auto =
  bw_lm2 |>
  autoplot(
    which = 2, # options are 1:6; can do multiple at once
    ncol = 1) +
  theme_classic()
```

```
print(qqplot_lm2_auto)
```



If the Gaussian model were correct, these points should follow the dotted line.

Note

Fig 2.4 panel (c) in Dobson is a little different; they didn't specify how they produced it, but other statistical analysis systems do things differently from R.

10.3.1 QQ plot - how it's built

Let's construct it by hand:

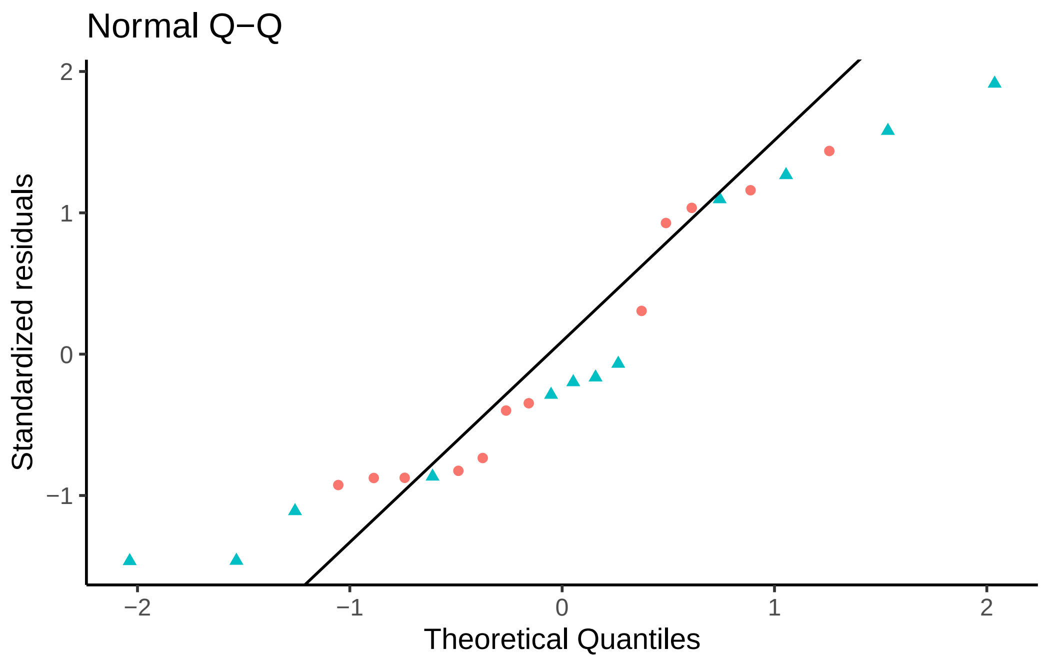
```
bw = bw |>
  mutate(
    p = (rank(std_resid_lm2) - 1/2)/n(), # "Blom's method"
    expected_quantiles_lm2 = qnorm(p)
  )

qqplot_lm2 =
```

```r
  bw |>
  ggplot(
    aes(
      x = expected_quantiles_lm2,
      y = std_resid_lm2,
      col = sex,
      shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  theme(legend.position='none') + # removing the plot legend
  ggtitle("Normal Q-Q") +
  xlab("Theoretical Quantiles") +
  ylab("Standardized residuals")
```

We find the expected line like so:

```r
ps <- c(.25, .75)                  # reference probabilities
a <- quantile(rstandard(bw_lm2), ps)  # empirical quantiles
b <- qnorm(ps)                     # theoretical quantiles

qq_slope = diff(a)/diff(b)
qq_intcpt = a[1] - b[1] * qq_slope

qqplot_lm2 =
  qqplot_lm2 +
  geom_abline(slope = qq_slope, intercept = qq_intcpt)
```

```r
print(qqplot_lm2)
```



10.4 Conditional distributions of residuals

If our Gaussian linear regression model is correct, the residuals e_i and standardized residuals r_i should have:

- an approximately Gaussian distribution, with:
- a mean of 0
- a constant variance

This should be true **regardless** of the value of X.

But if we didn't correctly guess the functional form of linear component of the mean,

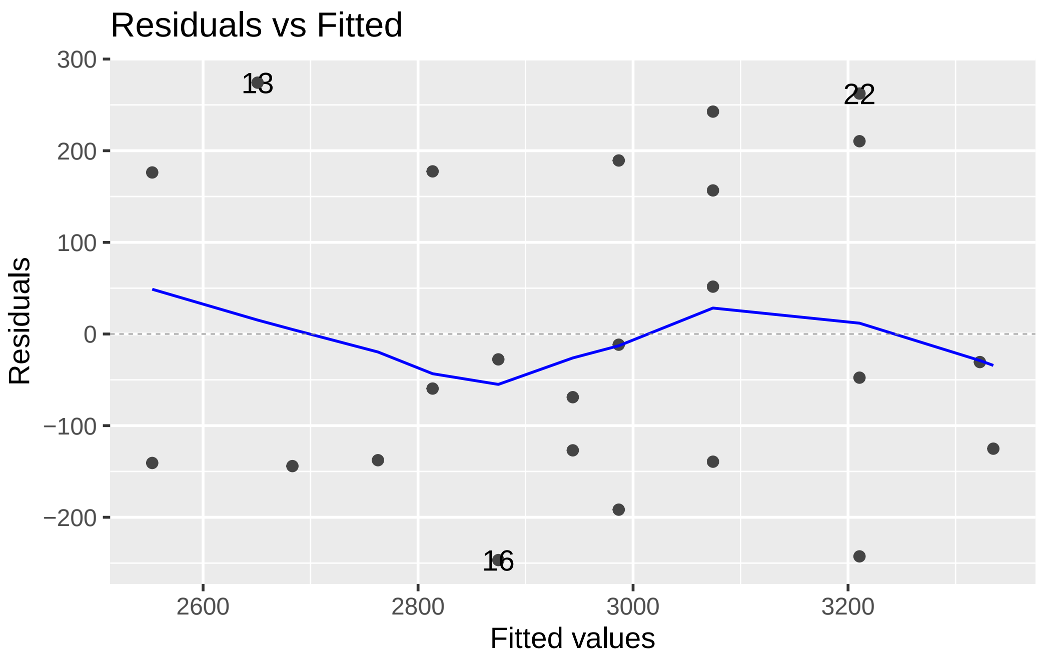
$$\mathrm{E}[Y|X = x] = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$$

Then the the residuals might have nonzero mean or nonconstant variance for some values of x.

10.4.1 Residuals versus fitted values

To look for these issues, we can plot the residuals e_i against the fitted values $\hat{y}_i$:

```r
autoplot(bw_lm2, which = 1, ncol = 1) |> print()
```



If the model is correct, the blue line should stay flat and close to 0, and the cloud of dots should have the same vertical spread regardless of the fitted value.

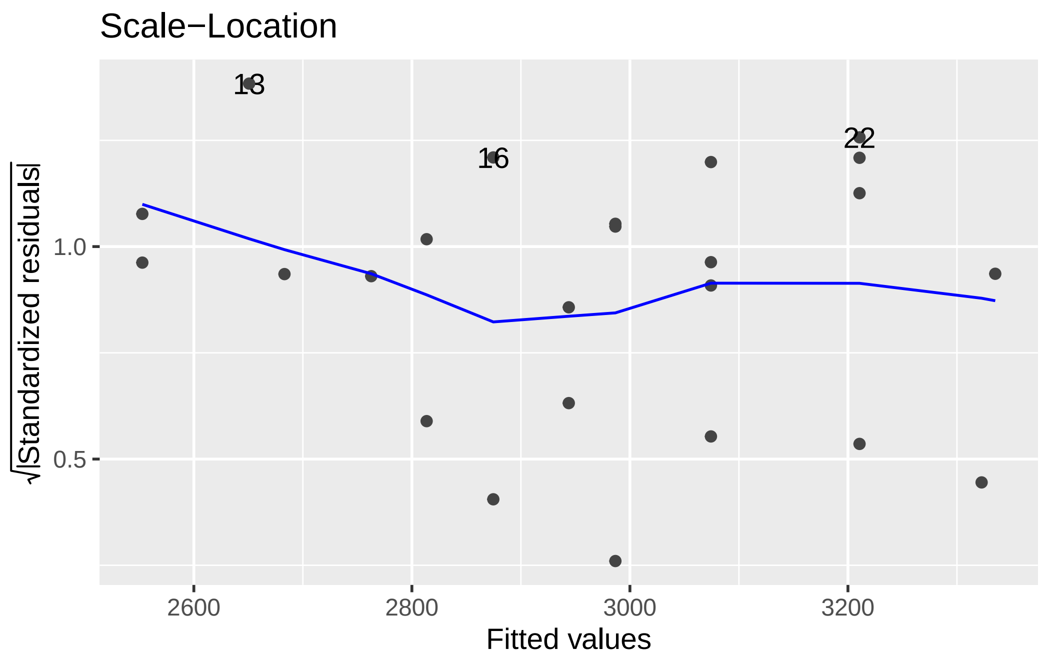
If not, we probably need to change the functional form of linear component of the mean,

$$\mathrm{E}[Y|X=x] = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

10.4.2 Scale-location plot

We can also plot the square roots of the absolute values of the standardized residuals against the fitted values:

```r
autoplot(bw_lm2, which = 3, ncol = 1) |> print()
```

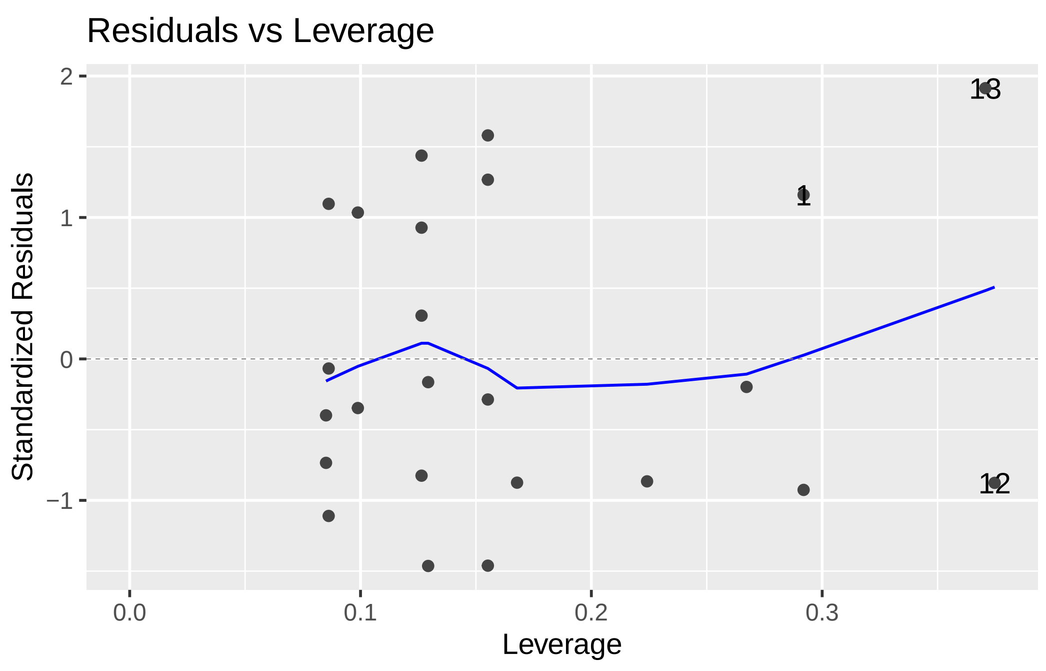


Here, the blue line doesn't need to be near 0, but it should be flat. If not, the residual variance σ^2 might not be constant, and we might need to transform our outcome Y (or use a model that allows non-constant variance).

10.4.3 Residuals versus leverage

We can also plot our standardized residuals against "leverage", which roughly speaking is a measure of how unusual each x_i value is. Very unusual x_i values can have extreme effects on the model fit, so we might want ot remove those observations as outliers, particularly if they have large residuals.

```r
autoplot(bw_lm2, which = 5, ncol = 1) |> print()
```



The blue line should be relatively flat and close to 0 here.

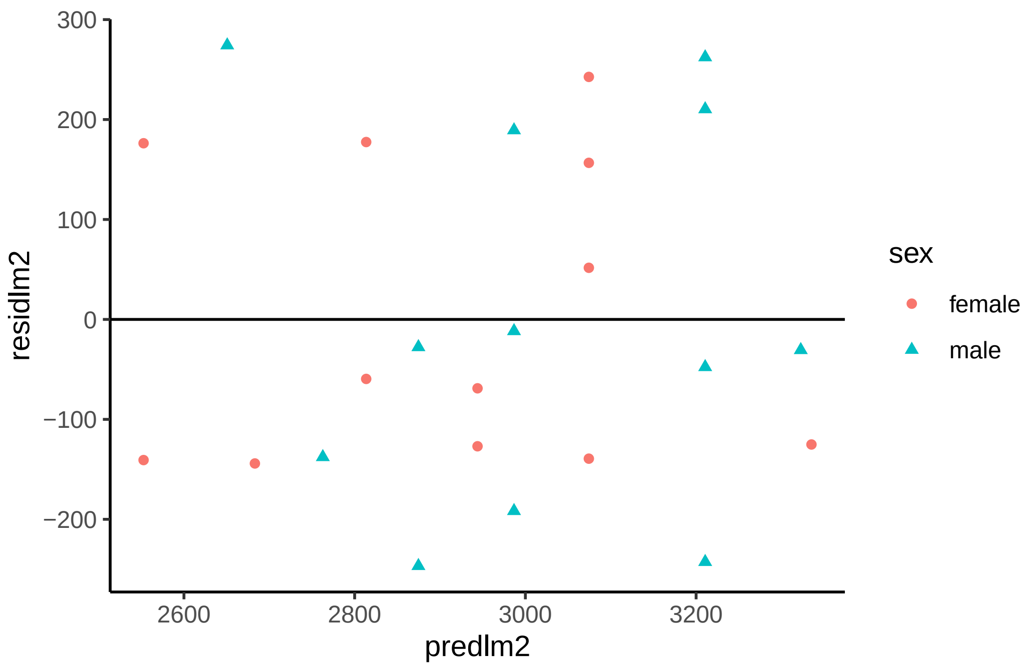
10.5 Diagnostics constructed by hand

```
bw =
  bw |>
  mutate(
    predlm2 = predict(bw_lm2),
    residlm2 = weight - predlm2,
    std_resid = residlm2 / sigma(bw_lm2),
    # std_resid_builtin = rstandard(bw_lm2), # uses leverage
    sqrt_abs_std_resid = std_resid |> abs() |> sqrt()

  )
```

10.5.0.1 Residuals vs fitted

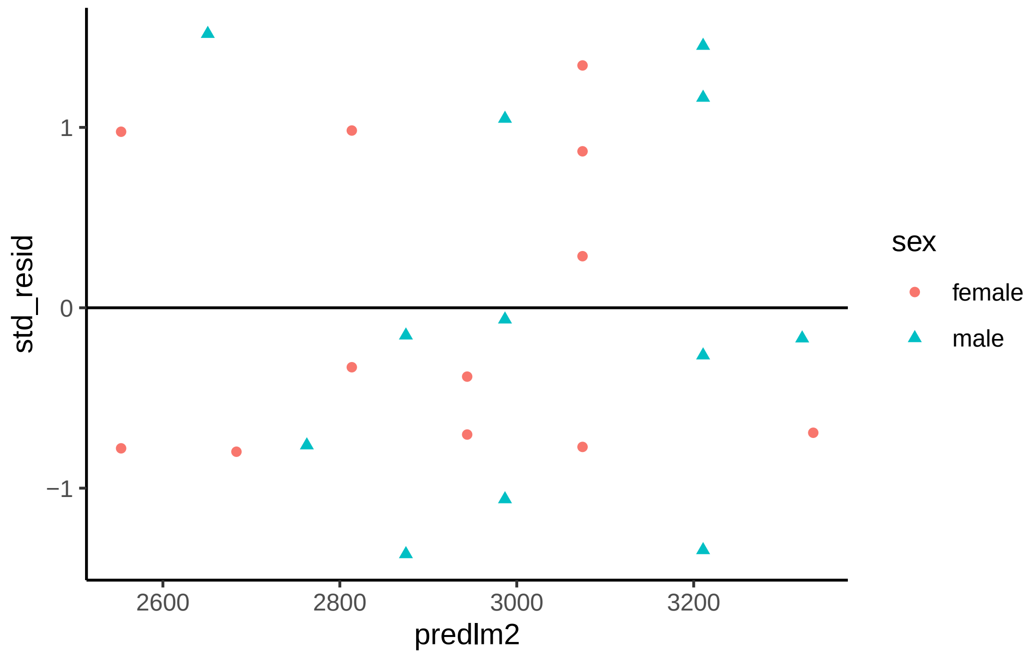
```
resid_vs_fit = bw |>
  ggplot(
    aes(x = predlm2, y = residlm2, col = sex, shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  geom_hline(yintercept = 0)
```

```
print(resid_vs_fit)
```



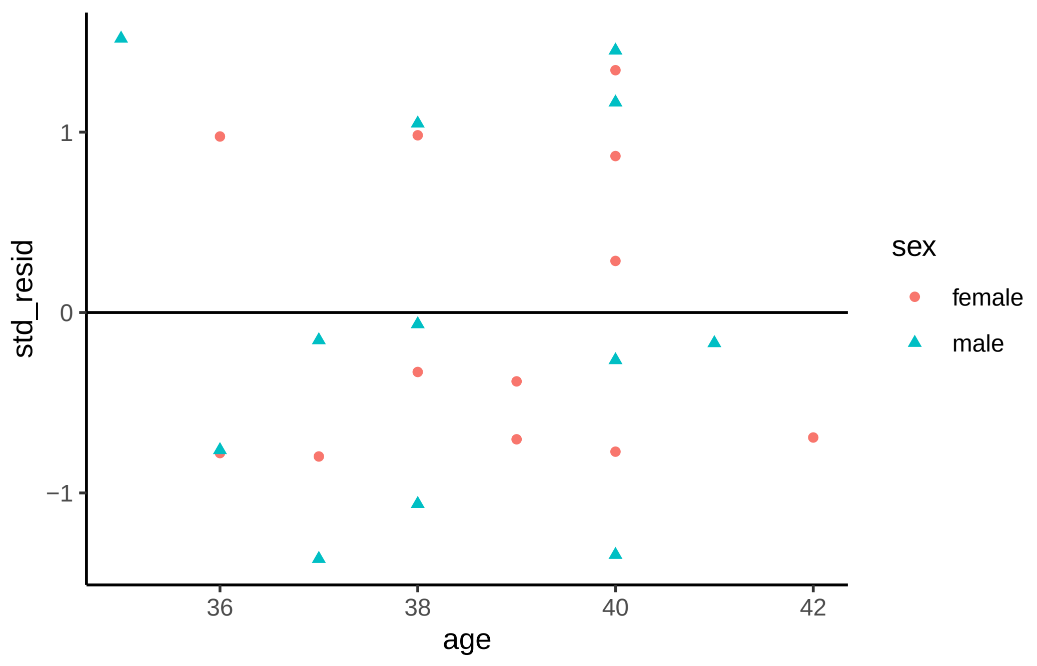
10.5.0.2 Standardized residuals vs fitted

```
bw |>
  ggplot(
    aes(x = predlm2, y = std_resid, col = sex, shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  geom_hline(yintercept = 0)
```



10.5.0.3 Standardized residuals vs gestational age

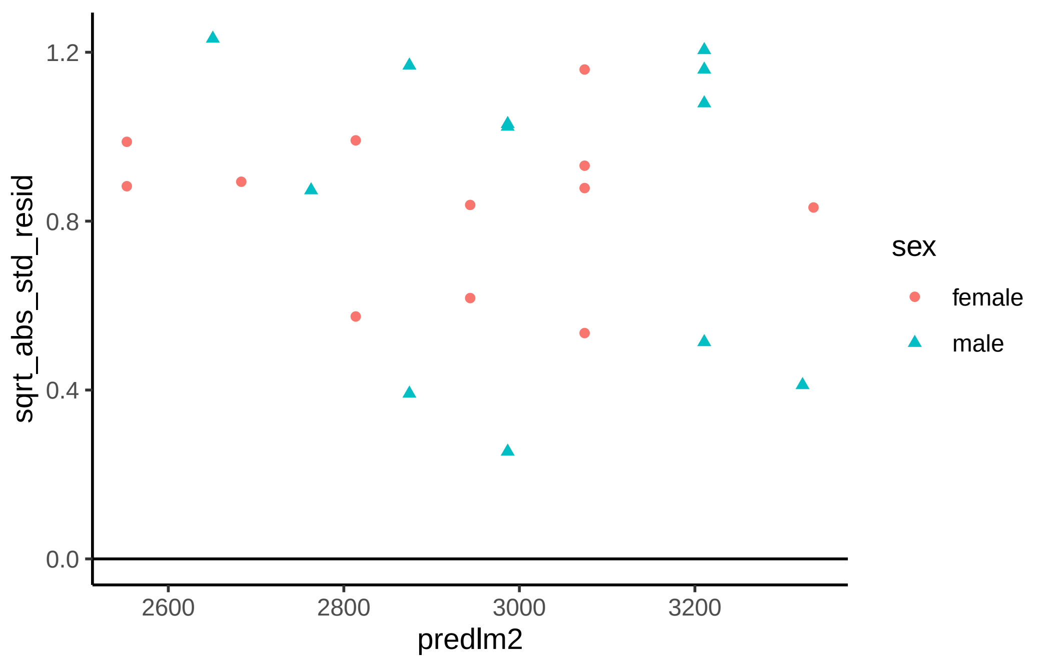
```
bw |>
  ggplot(
    aes(x = age, y = std_resid, col = sex, shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  geom_hline(yintercept = 0)
```



10.5.0.4 `sqrt(abs(rstandard()))` vs fitted

Compare with `autoplot(bw_lm2, 3)`

```
bw |>
  ggplot(
    aes(x = predlm2, y = sqrt_abs_std_resid, col = sex, shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  geom_hline(yintercept = 0)
```


sqrt_abs_std_resid
1.2
0.8
0.4
0.0
2600
2800
3000
3200
predlm2
sex
female
male

11 Model selection

If we have a lot of covariates in our dataset, we might want to choose a small subset to use in our model.

There are a few possible metrics to consider for choosing a "best" model.

11.1 Mean squared error

We might want to minimize the **mean squared error**, $\mathrm{E}[(y-\hat{y})^2]$, for new observations that weren't in our data set when we fit the model.

Unfortunately,

$$\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

gives a biased estimate of $\mathrm{E}[(y-\hat{y})^2]$ for new data. If we want an unbiased estimate, we will have to be clever.

11.2 Cross-validation

```r
data("carbohydrate", package = "dobson")
library(cvTools)
full.model <- lm(carbohydrate ~ ., data = carbohydrate)
temp =
  cvFit(full.model, data = carbohydrate, K = 5, R = 10,
y = carbohydrate$carbohydrate)
```

12 Categorical covariates with more than two levels

12.1

In the birthweight example, the variable `sex` had only two observed values:

```
unique(bw$sex)
```

```
[1] "female" "male"
```

If there are more than two observed values, we can't just use a single variable with 0s and 1s.

12.2

For example, here's the (in)famous `iris` data:

```
iris |> tibble()
```

```
# A tibble: 150 x 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa
 3          4.7         3.2          1.3         0.2 setosa
 4          4.6         3.1          1.5         0.2 setosa
 5          5           3.6          1.4         0.2 setosa
 6          5.4         3.9          1.7         0.4 setosa
 7          4.6         3.4          1.4         0.3 setosa
 8          5           3.4          1.5         0.2 setosa
 9          4.4         2.9          1.4         0.2 setosa
```

```
10          4.9          3.1          1.5         0.1 setosa
# i 140 more rows
```

```r
summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.30   Min.   :2.00   Min.   :1.00   Min.   :0.1   setosa    :50
 1st Qu.:5.10   1st Qu.:2.80   1st Qu.:1.60   1st Qu.:0.3   versicolor:50
 Median :5.80   Median :3.00   Median :4.35   Median :1.3   virginica :50
 Mean   :5.84   Mean   :3.06   Mean   :3.76   Mean   :1.2
 3rd Qu.:6.40   3rd Qu.:3.30   3rd Qu.:5.10   3rd Qu.:1.8
 Max.   :7.90   Max.   :4.40   Max.   :6.90   Max.   :2.5
```

12.3

There are three species:

```r
iris$Species |> unique()
```

```
[1] setosa     versicolor virginica
Levels: setosa versicolor virginica
```

12.4

If we want to model `Sepal.Length` by species, we could create a variable X that represents "setosa" as $X = 1$, "virginica" as $X = 2$, and "versicolor" as $X = 3$.

```r
data(iris) # this step is not always necessary, but ensures you're starting
# from the original version of a dataset stored in a loaded package

iris =
  iris |>
  tibble() |>
  mutate(
    X = case_when(
      Species == "setosa" ~ 1,
      Species == "virginica" ~ 2,
      Species == "versicolor" ~ 3
    )
  )

iris |>
  distinct(Species, X) |>
  print()
```

```
# A tibble: 3 x 2
  Species         X
  <fct>       <dbl>
1 setosa          1
2 versicolor      3
3 virginica       2
```

Then we could fit a model like:

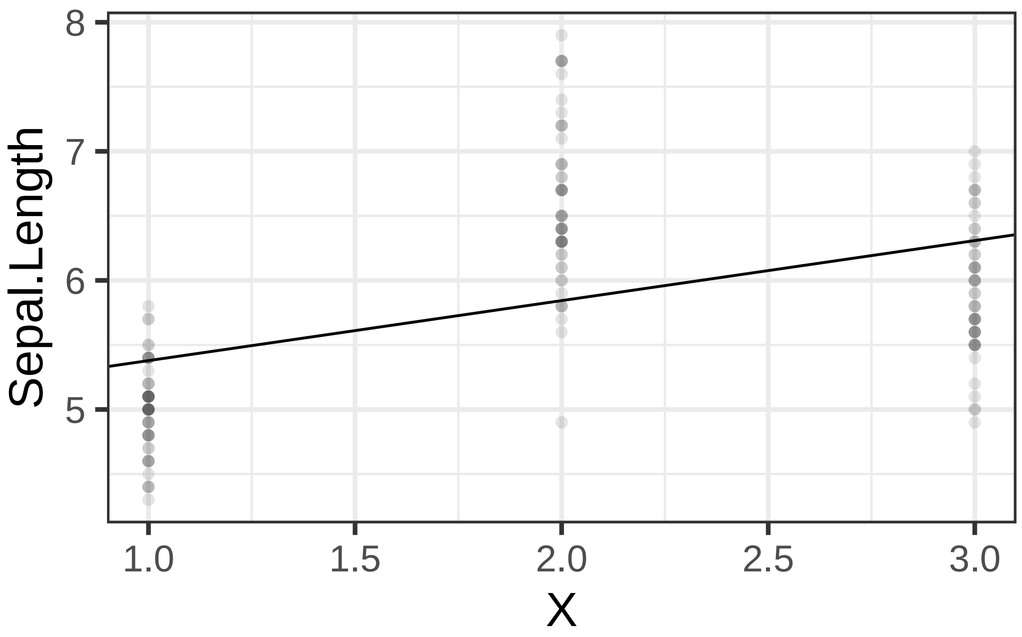
```
iris_lm1 = lm(Sepal.Length ~ X, data = iris)
iris_lm1 |> parameters() |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(148)	p
(Intercept)	4.91	0.16	(4.60, 5.23)	30.83	$< .001$
X	0.47	0.07	(0.32, 0.61)	6.30	$< .001$

12.5 Let's see how that model looks:

```
iris_plot1 = iris |>
  ggplot(
    aes(
      x = X,
      y = Sepal.Length)
  ) +
  geom_point(alpha = .1) +
  geom_abline(
    intercept = coef(iris_lm1)[1],
    slope = coef(iris_lm1)[2]) +
  theme_bw(base_size = 18)
```

```r
print(iris_plot1)
```



We have forced the model to use a straight line for the three estimated means. Maybe not a good idea?

12.6 Let's see what R does with categorical variables by default:

```r
iris_lm2 = lm(Sepal.Length ~ Species, data = iris)
iris_lm2 |> parameters() |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(147)	p
(Intercept)	5.01	0.07	(4.86, 5.15)	68.76	< .001
Species (versicolor)	0.93	0.10	(0.73, 1.13)	9.03	< .001
Species (virginica)	1.58	0.10	(1.38, 1.79)	15.37	< .001

12.7 Re-parametrize with no intercept

If you don't want the default and offset option, you can use "-1" like we've seen previously:

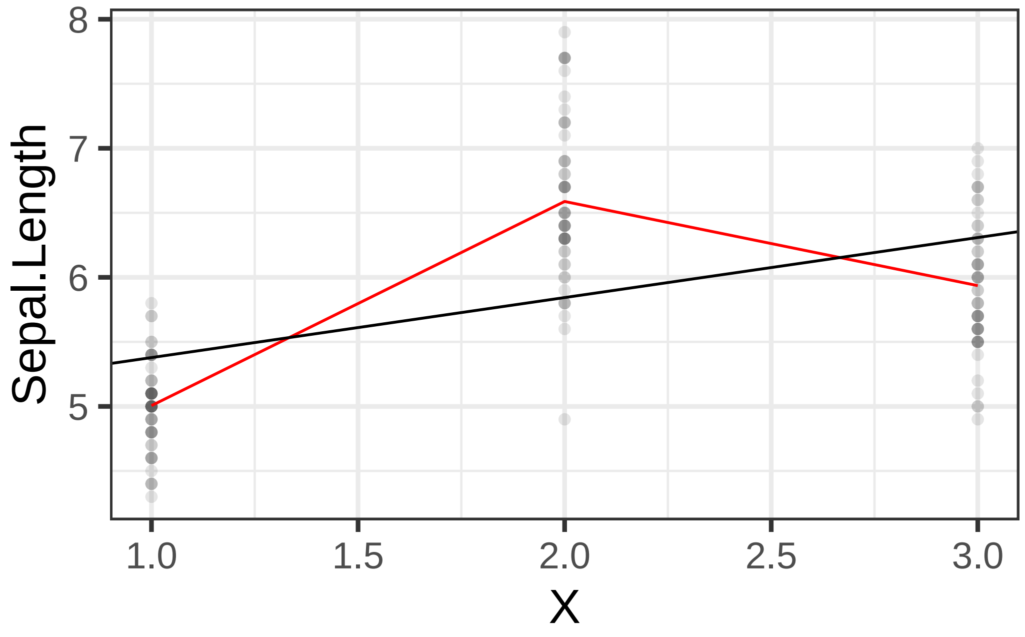
```r
iris.lm2b = lm(Sepal.Length ~ Species - 1, data = iris)
iris.lm2b |> parameters() |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(147)	p
Species (setosa)	5.01	0.07	(4.86, 5.15)	68.76	< .001
Species (versicolor)	5.94	0.07	(5.79, 6.08)	81.54	< .001
Species (virginica)	6.59	0.07	(6.44, 6.73)	90.49	< .001

12.8 Let's see what these new models look like:

```r
iris_plot2 =
  iris |>
  mutate(
    predlm2 = predict(iris_lm2)) |>
  arrange(X) |>
  ggplot(aes(x = X, y = Sepal.Length)) +
  geom_point(alpha = .1) +
  geom_line(aes(y = predlm2), col = "red") +
  geom_abline(
    intercept = coef(iris_lm1)[1],
    slope = coef(iris_lm1)[2]) +
  theme_bw(base_size = 18)
```

```r
print(iris_plot2)
```



12.9 Let's see how R did that:

```
formula(iris_lm2)
```

```
Sepal.Length ~ Species
```

```
model.matrix(iris_lm2) |> as_tibble() |> unique()
```

```
# A tibble: 3 x 3
  `(Intercept)` Speciesversicolor Speciesvirginica
          <dbl>             <dbl>            <dbl>
1             1                 0                0
2             1                 1                0
3             1                 0                1
```

This is called a "corner point parametrization".

```
formula(iris.lm2b)
```

```
Sepal.Length ~ Species - 1
```

```r
model.matrix(iris.lm2b) |> as_tibble() |> unique()
```

```
# A tibble: 3 x 3
  Speciessetosa Speciesversicolor Speciesvirginica
          <dbl>             <dbl>            <dbl>
1             1                 0                0
2             0                 1                0
3             0                 0                1
```

This can be called a “group point parametrization”.

There are more options; see Dobson & Barnett §6.4.1.

References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.