Simple Linear Regression in R

Part 4

Laura Bernhofen

2022-02-02

```
knitr::opts_chunk$set(echo = TRUE)
options(show.signif.stars = FALSE)
```

Load tidyverse

```
library(tidyverse)
```

Load the data set - rmr from the ISwR library

```
rmr <- ISwR::rmr
```

The Classical Simple Linear Regression Model

Model: $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$

Assumptions:

1. $E(\varepsilon_i) = 0$ for all $i = 1, 2, \ldots, n$
2. The model is linear: $E(Y_i) = \beta_0 + \beta_1 X_i$
3. The error terms are uncorrelated: $Corr(\varepsilon_i, \varepsilon_j) = 0$ for all $i \neq j$.
4. The variance of the error terms are constant. $Var(\varepsilon_i) = \sigma^2$ for all $i = 1, 2, \ldots n$

The Oridinary Least Squares (OLS) Estimates:

- $b_0 = \bar{Y} - b_1 \bar{X}$
- $b_1 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$ = $r\frac{S_Y}{S_X}$

The fitted model: $\hat{Y} = b_0 + b_1 X$ estimates $E(Y) = \beta_0 + \beta_1 X$

Properties of the OLS estimators

1. $E(b_0) = \beta_0$
2. $E(b_1) = \beta_1$

Properties (1) and (2) => that b_0 and b_1 are unbiased estimates of β_0 and β_1, respectively.

3. **Gauss-Markov Theorem**: Under assumptions 1-4 of the simple linear regression model, the ordinary least squares (OLS) estimators b_0 and b_1 have *minimum variance* among all *linear unbiased estimators*.

Note: The OLS estimators b_0 and b_1 are said to be the **Best Linear Unbiased Estimators (BLUE)** of β_0 and β_1.

Proof:

Estimating the unknown variance σ^2

- Recall from basic statistics, to estimate the unknown variance σ^2 of a random variable X, we collect a random sample $X_1, X_2, \ldots, X_n$,then calculate the sample variance $S^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$.
- The sample variance S^2 is an unbiased estimator of σ^2 with $n-1$ degrees of freedom. (i.e. $E(S^2) = \sigma^2$).
- In our linear regression model $Var(\varepsilon) = \sigma^2$. The residual e_i is an estimate of ε_i, therefore to estimate the unknown variance σ^2 in the linear regression model, we can use the same argument as above.
  - $\sum_{i=1}^{n}(e_i - \bar{e})^2 = \sum_{i=1}^{n}(e_i - 0)^2 = \sum_{i=1}^{n} e_i^2 = \text{SSE}$
  - $\hat{\sigma}^2 = \frac{\text{SSE}}{n-2}$ = MSE (Mean Square Error)
- The MSE = $\frac{\text{SSE}}{n-2}$ is an unbiased estimator for the unknown variance σ^2 in the linear regression model.
- Note: The MSE = $\frac{\text{SSE}}{n-2}$ has $n-2$ degrees of freedom.

Complete Output for the Model we fit using base R

```
mrout <- lm(metabolic.rate ~ body.weight, data = rmr)
summary(mrout)
```

```
##
## Call:
## lm(formula = metabolic.rate ~ body.weight, data = rmr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -245.74 -113.99  -32.05  104.96  484.81
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 811.2267    76.9755  10.539 2.29e-13
## body.weight   7.0595     0.9776   7.221 7.03e-09
##
## Residual standard error: 157.9 on 42 degrees of freedom
## Multiple R-squared:  0.5539, Adjusted R-squared:  0.5433
## F-statistic: 52.15 on 1 and 42 DF,  p-value: 7.025e-09
```

To get the output as a tibble (tidyverse) format

1. In the console install the package {broom} using the command install.package("broom")
2. load the {broom} package

```
library(broom)
```

3. Instead of using the function summary(), we can use the tidy() function from the {broom} package to get some of the basic output.

- format: tidy(object) where object is the name you assigned the regression output using the lm() function.

```
out1 <- tidy(mrout)
out1
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   811.      77.0       10.5  2.29e-13
## 2 body.weight     7.06     0.978      7.22 7.03e- 9
```

```
# I have used the notation `r out1$estimate[1]` to input the value of b_0 and `r out1$estimate
[2]` to imput the value of b_1 in my text.
```

- $b_0 = \hat{\beta}_0$ = 811.2266745
- $b_1 = \hat{\beta}_1$ = 7.0595278
- Fitted model: $\widehat{metabolicrate} = 811.23 + 7.06 \cdot bodywt$

4. We can get additional information from our regression model using the glance() function from the {broom} package.

- format: glance(object) where again object is the name you assigned the regression output using the lm() function.

```
out2 <- glance(mrout)
out2
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic       p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>         <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.554         0.543  158.      52.1 0.00000000703     1  -284.  574.  580.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
#I have used the notation `r out2$sigma` to get the value from the tibble printed in my text.
#Notice also in the output table there is a message that there are 3 more variables reported in
 the output:
#deviance = SSE, df.residual = df associated with the SSE, nobs = number of observations used to
fit our model.
```

- sigma = $\sqrt{\text{MSE}}$ = 157.9052389 which estimates $\sigma = \sqrt{Var(\varepsilon)}$

The Normal Classical Simple Linear Regression Model

Model: $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$

Assumptions:

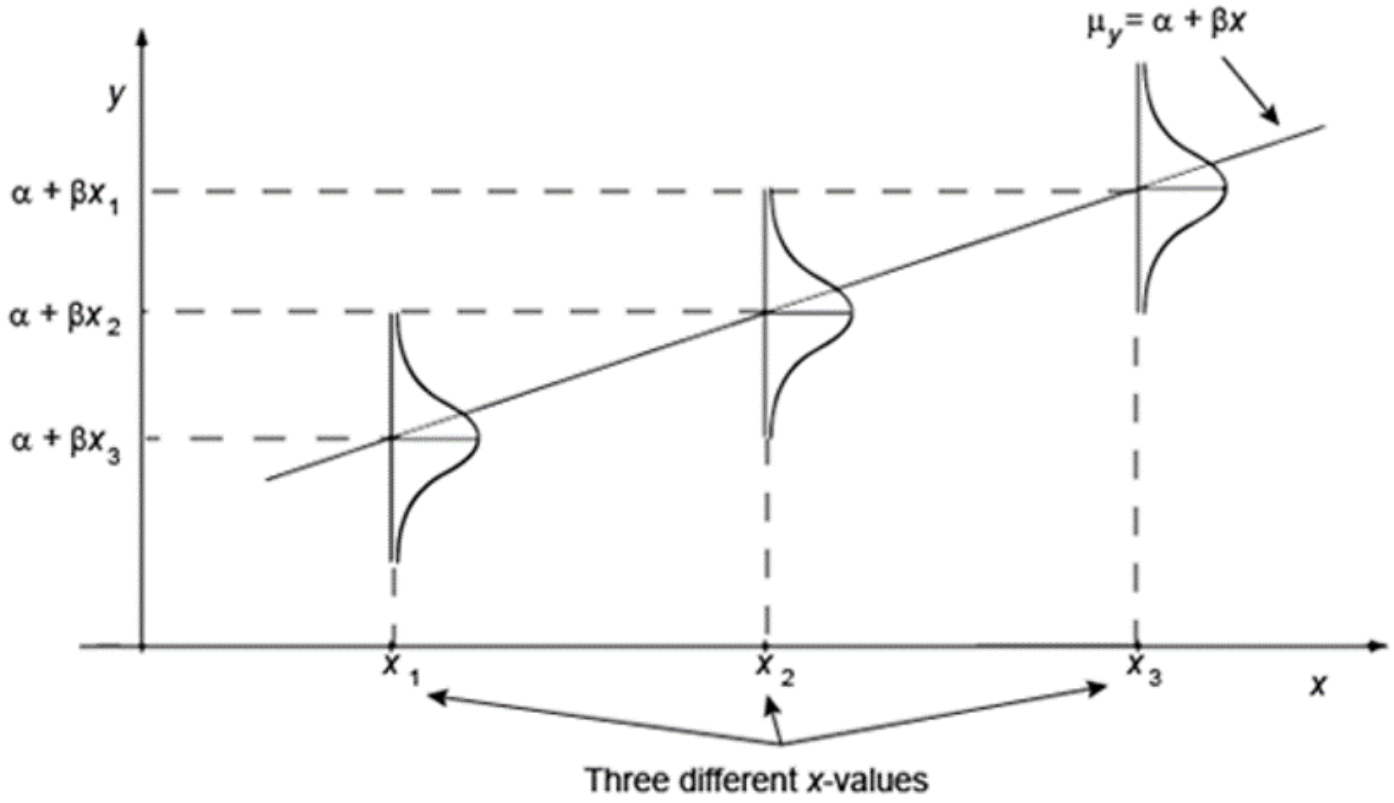
1. $E(\varepsilon_i) = 0$ for all $i = 1, 2, \ldots, n$
2. The model is linear: $E(Y_i) = \beta_0 + \beta_1 X_i$
3. The error terms are uncorrelated: $Corr(\varepsilon_i, \varepsilon_j) = 0$ for all $i \neq j$.
4. The variance of the error terms are constant. $Var(\varepsilon_i) = \sigma^2$ for all $i = 1, 2, \ldots n$
5. $\varepsilon_i \sim N(0, \sigma^2)$ for all $i = 1, \ldots, n$

Note: As consequences of the normal error assumption,

i. $Y_i \sim N(\beta_0 + \beta_1 X_i, \sigma^2)$ for all $i = 1, \ldots, n$

ii. The error terms $\varepsilon_i, \varepsilon_j$ are **independent** for all $i \neq j$.

iii. Y_i, Y_j are **independent** for all $i \neq j$.



Normal Linear Regression Model assumptions graph

Properties of the OLS estimators in the Normal Simple Regression Model

In addition to the original properties of the OLS estimators which didn't depend on the the normality assumption, when we add the assumption $\varepsilon \sim N(0, \sigma^2)$, we get

1. b_0 and b_1 are the **Minimum Variance Unbiased Estimators (MVUE)** of β_0 and β_1.
    - In the set of all unbiased estimators, the OLS estimators have the smallest variance!
2. The OLS estimators b_0 and b_1 are also the **Maximum Likelihood Estimators (MLE's)** of β_0 and β_1.

Inferences in the Simple Linear Regression Model

The Distribution of b_1 and b_0

1. $b_1 \sim N(\beta_1, \frac{\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2})$

Proof:

- b_1 is the unbiased estimator of β_1, therefore, $E(b_1)$ = β_1
- Recall: $b_1 = \sum_{i=1}^n k_i Y_i$ where $k_i = \frac{X_i - \bar{X}}{\sum_{i=1}^n (X_i - \bar{X})^2}$
- Since $\varepsilon_i \sim N(0, \sigma^2)$ => Y_i's are normally distributed and the Y_i's are independent for all $i = 1, 2, .., n$
- If $Y_1, Y_2, \ldots, Y_n$ are independent and $Y_i \sim N(\mu_i, \sigma_i^2)$ then $\sum_{i=1}^n Y_i \sim N(\sum_{i=1}^n \mu_i, \sum_{i=1}^n \sigma_i^2)$

$$
\begin{aligned}
Var(\sum_{i=1}^n k_i Y_i) &= \sum_{i=1}^n Var(k_i Y_i) \text{ since } Y_i \text{ are independent} \\
&= \sum_{i=1}^n k_i^2 Var(Y_i) = \sum_{i=1}^n k_i^2 \sigma^2 \text{ since the variance is constant for all } Y_i\text{'s.} \\
&= \sigma^2 \sum_{i=1}^n \frac{(X_i - \bar{X})^2}{[\sum_{i=1}^n (X_i - \bar{X})^2]^2} \\
&= \sigma^2 \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{[\sum_{i=1}^n (X_i - \bar{X})^2]^2} \\
&= \sigma^2 \frac{1}{\sum_{i=1}^n (X_i - \bar{X})^2}
\end{aligned}
$$

Question: What happens to the variance of b_1, when we have a wide range of values in our predictor variables?

2. $b_0 \sim N(\beta_0, \sigma^2 [\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2}])$

- This can be derived similarly as above using $b_0 = \sum_{i=1}^n m_i Y_i$ where $m_i = \frac{1}{n} + \frac{\bar{X}(X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2}$

Other Distributional Results

- $\frac{b_1 - \beta_1}{\sqrt{Var(b_1)}} \sim$

Problem: σ^2 is unknown so $Var(b_1) = \frac{\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2}$ is unknown!

Question: What do we do?

Inferences involving β_1

Testing Hypotheses Involving β_1

Test:

- $H_0 : \beta_1 = c$
- $H_a : \beta_1 \neq c$

Test Statistic:

- General format for many test statistics: $TS = \frac{\text{estimate} - \text{parameter}_0}{SE(estimate)}$
- Specific Test Statistic:

Def: The **p-value** is the probability of observing a value of our Test Statistic more extreme than the value we have observed from our data given the null hypothesis (H_0) is true.

Calculating p-values in R.

1. One-sided upper-tailed test:

- $H_0 : \theta = \theta_0$
- $H_a : \theta > \theta_0$
- p-value = P($T_{df} > t$), where t = value of the test statistic given our data.
    - We use the function **pt(t, df, lower.tail = FALSE)**

2. One-sided lower-tailed test:

- $H_0 : \theta = \theta_0$
- $H_a : \theta < \theta_0$
- p-value = P($T_{df} < t$), where t = value of the test statistic given our data.
    - We use the function **pt(t, df)**

3. Two-sided test:

- $H_0 : \theta = \theta_0$

- $H_a : \theta \neq \theta_0$
- p-value = P($|T_{df}| > |t|$), where t = value of the test statistic given our data.
    - We can use the function **2** $\times$ **pt(abs(t), df, lower.tail = FALSE)**

Decision Rule:

- If the p-value is small, we **reject** H_0 and conclude there is sufficient statistical evidence to conclude claim H_a is true.
- If the p-value is large, we **do not reject** H_0 and conclude there is insufficient statistical evidence to claim H_a is true.

Note: What is small/large depends upon the context of the problem.

Rule of Thumb:

- p-value < 0.001 - Very Strong Evidence
- p-value < 0.01 - Strong Evidence
- p-value < 0.05 - Some Evidence
- p-value < 0.1 - Very weak evidence
- p-value $\geq$ 0.1 - No evidence

Testing whether there exists a linear relationship between Y and X in the linear model:
$Y = \beta_0 + \beta_1 X + \varepsilon$

Test:

- $H_0 : \beta_1 = 0$
- $H_a : \beta_1 \neq 0$

Test Statistic:

- T =

example 1: Does our data support that there exists a linear relationship between metabolic rate and body weight in females?

```
#Recall we have already fit our model.  If not we would need to first fit the model using mrout
 <- lm(metabolic.rate ~ body.weight)
out1 <- tidy(mrout)
out1
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   811.      77.0       10.5  2.29e-13
## 2 body.weight     7.06     0.978      7.22 7.03e- 9
```

```
out2 <- glance(mrout)
out2
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic       p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>         <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.554         0.543  158.      52.1 0.00000000703     1  -284.  574.  580.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Test:

- $H_0 : \beta_1 = 0$
- $H_a : \beta_1 \neq 0$

Test Statistic:

```
TSb1 <- (out1$estimate[2] - 0)/(out1$std.error[2])
TSb1
```

```
## [1] 7.221301
```

```
dfb1 <- out2$df.residual
dfb1
```

```
## [1] 42
```

p-value

```
pvalb1 <- 2*pt(abs(TSb1), dfb1, lower.tail = FALSE)
pvalb1
```

```
## [1] 7.02538e-09
```

Conclusion: Given the test statistic T = 7.2213005 and its associated p-value = 7.0253804^{-9}, if H_0 were true it would be highly unlikely that we would observe a test statistic of this magnitude or more extreme. Consequently, we will reject H_0 and conclude that there is overwhelming statistical evidence to indicate that there is a linear relationship between metabolic rate and body weight in our model.

Remark: The table of output given by tidy() function gives us the information we need to test the hypotheses:

- $H_0 : \beta_1 = 0$

- $H_a : \beta_1 \neq 0$

The value of the test statistic is given by the column statistic and the p-value is reported in the column labeled p.value.

example 2: Your Basal Metabolic Rate (BMR) is the number of calories you burn as your body performs basic (basal) life-sustaining function. Commonly also termed as Resting Metabolic Rate (RMR), which is the calories burned if you stayed in bed all day.
For Women: BMR = 447.593 + (9.247 x weight in kg) + (3.098 x height in cm) – (4.330 x age in years) Garnet Health (https://www.garnethealth.org/news/basal-metabolic-rate-calculator).
Test whether our the slope of our model is consistent with their result, i.e. $\beta_1 = 9.25$

Test:

- H_0:
- H_a:

Test statistic:

```
TSb12 <- (out1$estimate[2] - 9.25)/(out1$std.error[2])
TSb12
```

```
## [1] -2.240668
```

p-value:

```
pval2 <- 2*pt(abs(TSb12), dfb1, lower.tail = FALSE)
pval2
```

```
## [1] 0.03039367
```

Conclusion: Given the test statistic T = -2.40 and the associated p-value = 0.0304, if H_0 we true it would be …. likely to observe a test statistic of this magnitude or more extreme; consequently, we ……. H_0 and claim there …….. evidence to indicate that our model is consistent with the result from Garnet Health.

Confidence Interval for β_1

General Formula for a Confidence Interval: estimate $\pm$ margin of error = estimate $\pm$ Sampling_Dist(estimate) $\times$ SE(estimate)

To use R to calculate a confidence interval for the parameters β_0 and β_1, we can use the tidy() function from the {broom} package.

- tidy(object, conf.int=TRUE, conf.level=1-α), where $0 - is the desired confidence level. Default conf.level = 0.95.

Example: Construct a 98% confidence interval for β_1.

```
out3 <- tidy(mrout, conf.int=TRUE, conf.level=0.98)
out3
```

```
## # A tibble: 2 x 7
##   term        estimate std.error statistic  p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)   811.      77.0       10.5  2.29e-13   625.      997.
## 2 body.weight     7.06     0.978      7.22 7.03e- 9     4.70      9.42
```

```
# To print the confidence limits we can use the sytax `r out3$conf.low[2]` for the lower confide
nce limit and
# 'r out3$conf.high[2]` to report the upper confidence limit for beta_1.
```

A 98% confidence interval for β_1 is given by (4.6952364, 9.4238192).

Testing Hypotheses Involving β_0

Test:

- $H_0 : \beta_0 = a$
- $H_a : \beta_0 \neq a$

Test Statistic:

- $T = \frac{b_0 - a}{SE(b_0)} \sim t_{n-2}$

Exercise: Test whether we need the constant term in our model: $E(Y) = \beta_0 + \beta_1 X$

Test:

- H_0 :
- H_a :

Test Statistic:

p-value =

Conclusion:

Confidence interval for β_0

- $b_0 \pm t^*_{n-2} \times SE(b_0)$

Exercise: You construct a 99% confidence interval for β_0

Using R:

```
out3b <- tidy(mrout, conf.int = TRUE, conf.level = 0.99)
out3b
```

```
## # A tibble: 2 x 7
##   term        estimate std.error statistic  p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)   811.      77.0       10.5  2.29e-13   604.     1019.
## 2 body.weight     7.06     0.978      7.22 7.03e- 9     4.42      9.70
```

99% Confidence Interval for β_0:

Confidence Interval for $E(Y|X = x_0)$

Comment: This is the range of values over which we would expect the E(Y) (i.e. the mean response) to vary given that the explanatory variable $X = x_0$, where x_0 is a specified value.

Parameter: $E(Y|x_0) = \beta_0 + \beta_1 x_0$ (true regression line)

Estimate: $\hat{Y} = b_0 + b_1 x_0$ (fitted line)

$SE(\hat{Y}) = s_p \sqrt{\frac{1}{n} + \frac{(x_0 - \bar{X})^2}{(n-1)S_X^2}}$ where $s_p = \sqrt{MSE}$ with $df = n - 2$ and $S_X^2 = \sum_{i=1}^{n}(X_i - \bar{X})^2$

Confidence Interval: $\hat{Y} \pm t^*_{n-2} \times SE(\hat{Y})$

Using R to find a $(1 - \alpha)$ 100% Confidence interval for $E(Y|X = x_0)$

1. Fit your linear model and save it to an object, like we have previously done. (If you already have it saved as an object you can skip this step)

```
mrout <- lm(metabolic.rate ~ body.weight, data = rmr)
```

2. Create a new data frame that contains the desired level(s) of X_i

```
newbwt <- data.frame(body.weight = c(59, 70, 90))
```

3. Use the function **predict()** to calculate the confidence intervals for the given values of X_i

- format predict(object, newdata, interval = “confidence”, level = desired.confidence)

Example: Construct a 95% Confidence Interval for the mean metabolic rate when the body weight of the females given are 59kg, 70kg, 90 kg.

```
predict(mrout, newbwt, interval = "confidence", level = 0.95)
```

```
##        fit      lwr      upr
## 1 1227.739 1170.386 1285.092
## 2 1305.394 1256.398 1354.389
## 3 1446.584 1390.035 1503.133
```

Output

- fit = $E(Y|x_0)$
- lwr = lower limit of the confidence interval for $E(Y|x_0)$
- upr = upper limit of the confidence interval for $E(Y|x_0)$

Note: To assign the output to an object and include the values of the predictor variables in the output use the following code chunk

```
predict(mrout, newbwt, interval = "confidence", level = 0.95) %>%
  cbind(newbwt) ->   #Adds a new column to the output table that contains the values of the pred
ictor variable
  ciEY
ciEY
```

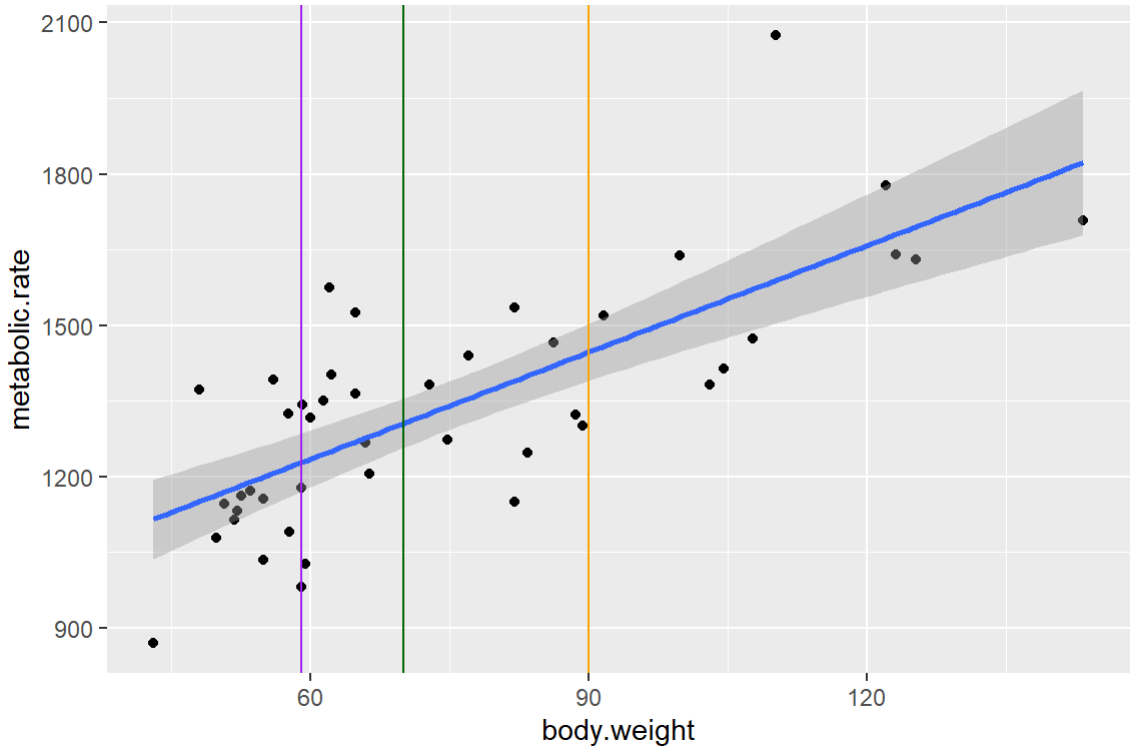
```
##        fit      lwr      upr body.weight
## 1 1227.739 1170.386 1285.092          59
## 2 1305.394 1256.398 1354.389          70
## 3 1446.584 1390.035 1503.133          90
```

Results The average metabolic rate of a female who weighs 59 kg is 1228 kcal/24hr. We are 95% confident that the average metabolic rate a women who weighs 59 kg is captured by the interval (1170 kcal/24hr, 1285 kcal/24hr.) The other results for the other intervals can be similarly interpreted.

Graphical Representation of the Confidence Intervals for $E(Y|X = x_0)$

```
ggplot(rmr, aes(x = body.weight, y = metabolic.rate)) +
  geom_point() +
  geom_smooth(method = 'lm', se = TRUE) +
  geom_vline(xintercept = 59, col = "purple") +
  geom_vline(xintercept = 70, color = "darkgreen") +
  geom_vline(xintercept = 90, color = "orange")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Question: Why do the confidence intervals have different widths? Where will the width of the confidence inteval for $E(Y|X = x_0)$ be the narrowest?

Hint: $SE(\hat{Y}) = s_p\sqrt{\frac{1}{n} + \frac{(x_0-\bar{X})^2}{(n-1)S_X^2}}$ where $s_p = \sqrt{MSE}$ with $df = n - 2$ and $S_X^2 = \sum_{i=1}^{n}(X_i - \bar{X})^2$

Confidence Bands for $E(Y|X)$

Comment: This is the range of values over which we would expect $E(Y)$ (i.e. the average response) to vary given that the explanatory variable X ranges over all possible values X could take on. (This can be thought of as a confidence band for the true regression line $E(Y) = \beta_0 - \beta_1 X$)

Parameter: $E(Y|X) = \beta_0 + \beta_1 X$ (true regression line)

Estimate: $\hat{Y} = b_0 + b_1 X$ (fitted line)

$SE(\hat{Y}) = s_p\sqrt{\frac{1}{n} + \frac{(X-\bar{X})^2}{(n-1)S_X^2}}$ where $s_p = \sqrt{MSE}$ with $df = n - 2$ and $S_X^2 = \sum_{i=1}^{n}(X_i - \bar{X})^2$

Confidence Band: $\hat{Y} \pm \sqrt{2F^*_{2,n-2}(1-\alpha)} \times SE(\hat{Y})$

Note: We use a different distribution because we want a joint confidence level of $1 - \alpha$ for all values of X.

Unfortunately R doesn't seem to have a built-in function to compute the confidence bands, but Prof. Gerard created a function that will do it for us.

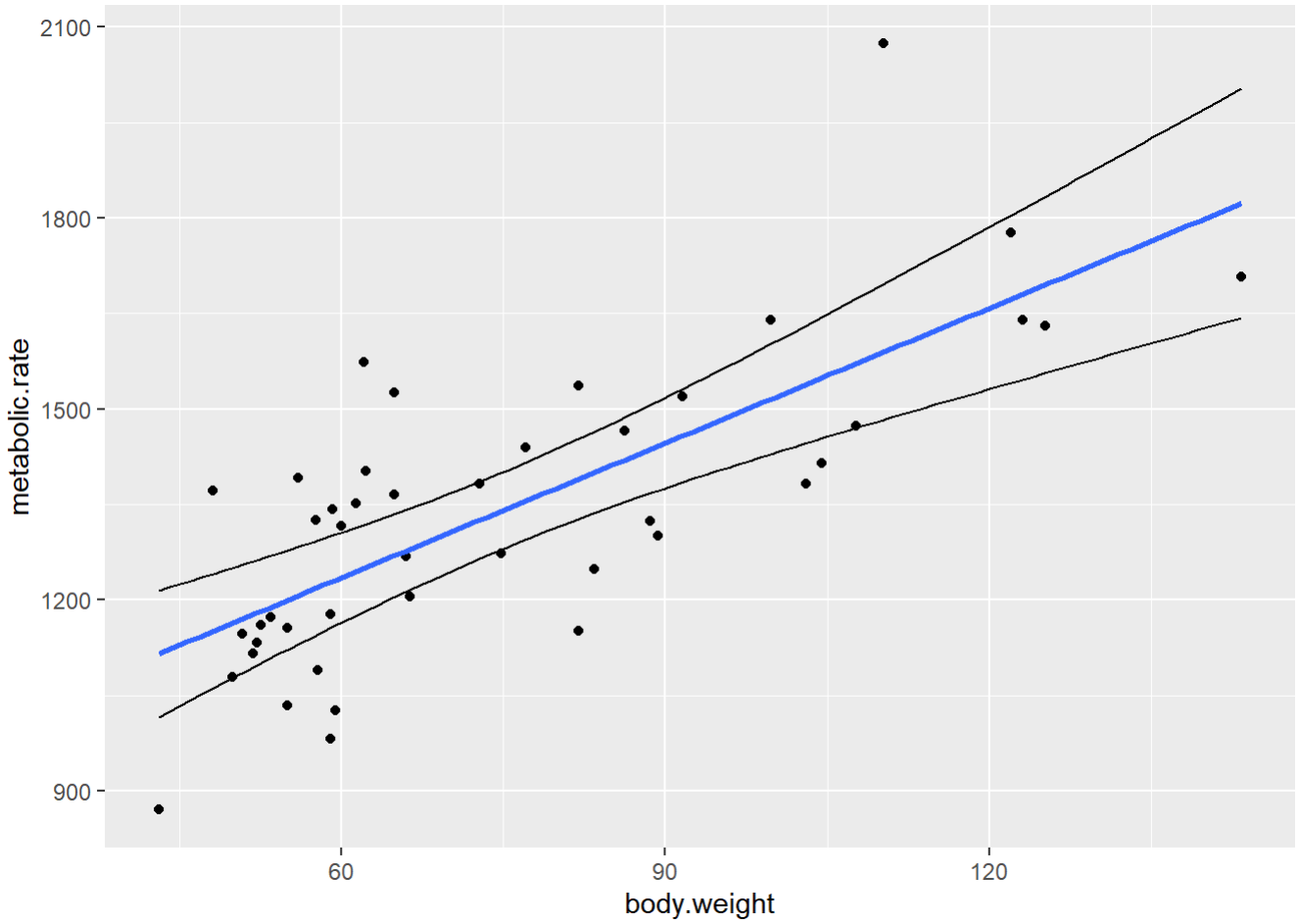
```
#' Working-Hotelling bands for simple linear regression.
#'
#' Intervals of the form "fit +/- w * standard-error", where w^2 is
#' found by \code{p * qf(level, p, n - p)}.
#'
#' @param object An object of class "lm".
#' @param newdata A data frame containing the new data.
#' @param level The confidence level of the band.
#'
#' @author David Gerard
whbands <- function(object, newdata, level = 0.95) {
  stopifnot(inherits(object, "lm"))
  stopifnot(inherits(newdata, "data.frame"))
  stopifnot(is.numeric(level), length(level) == 1)
  pout <- stats::predict(object = object,
                         newdata = newdata,
                         se.fit = TRUE,
                         interval = "none")
  n <- nrow(stats::model.frame(object))
  p <- ncol(stats::model.frame(object))
  w <- sqrt(p * stats::qf(p = level, df1 = p, df2 = n - p))
  lwr <- pout$fit - w * pout$se.fit
  upr <- pout$fit + w * pout$se.fit
  pout$fit <- cbind(fit = pout$fit, lwr = lwr, upr = upr)
  return(pout)
}
```

Applying his function to our model to get the 95% confidence bands for $E(Y)$

```
mrout <- lm(metabolic.rate ~ body.weight, data = rmr)
newdf <- data.frame(body.weight = seq(from = min(rmr$body.weight),
to = max(rmr$body.weight), length.out = 100))
whfit <- whbands(object = mrout, newdata = newdf)
whfit$fit %>%
cbind(newdf) -> newdf

ggplot() +
  geom_point(data = rmr, mapping = aes(x = body.weight, y = metabolic.rate))+
  geom_smooth(data = rmr, mapping = aes(x = body.weight, y = metabolic.rate), method = 'lm', se
 = FALSE) +
  geom_line(data = newdf, mapping = aes(x = body.weight, y = lwr)) +
  geom_line(data = newdf, mapping = aes(x = body.weight, y = upr))
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Prediction Interval for $Y|X = x_0$

Note: A prediction interval gives the range of values that an individual(future) value of the response variable attains given a specific value of the explanatory variable with a specified degree of confidence.

Ex. According to our model, if a woman weighs 59 kg, over what values would her metabolic rate likely range?

Remark: The difference between a confidence interval and a prediction interval:

- In a **confidence interval** we want to know over what values the **mean response** would vary.
- In a **prediction interval** we want to know over what values a **specific value of the response** would vary.

For a prediction interval we want to estimate: $Y|x_0 = \beta_0 + \beta_1 x_0 + \varepsilon$

Estimator: $\hat{Y} = b_0 + b_1 x_0$ (fitted line)

$SE(pred(\hat{Y})) = s_p\sqrt{(1 + \frac{1}{n} + \frac{(x_0 - \bar{X})^2}{(n-1)s_X^2}}$ where $s_p = \sqrt{MSE}$ with $df = n - 2$

Prediction Interval: $\hat{Y} \pm t^*_{n-2} \times SE(pred(\hat{Y}))$

Constructing a Prediction Interval for $Y|X = x_0$ in R

- The method for constructing a Prediction Interval is identical to the method for constructing a confidence interval for $E(Y|X= x_0) except we when we use the predict function, we use the argument interval = “prediction”

Example: Construct a 95% Prediction Interval for the metabolic rate of a woman who weighs 59kg.

```
newpidf <- data.frame(body.weight = 59)
predict(mrout, newpidf, interval = "prediction", level = 0.95) %>%
  cbind(newpidf) ->
  piout
piout
```

```
##        fit      lwr      upr body.weight
## 1 1227.739 903.9531 1551.525          59
```

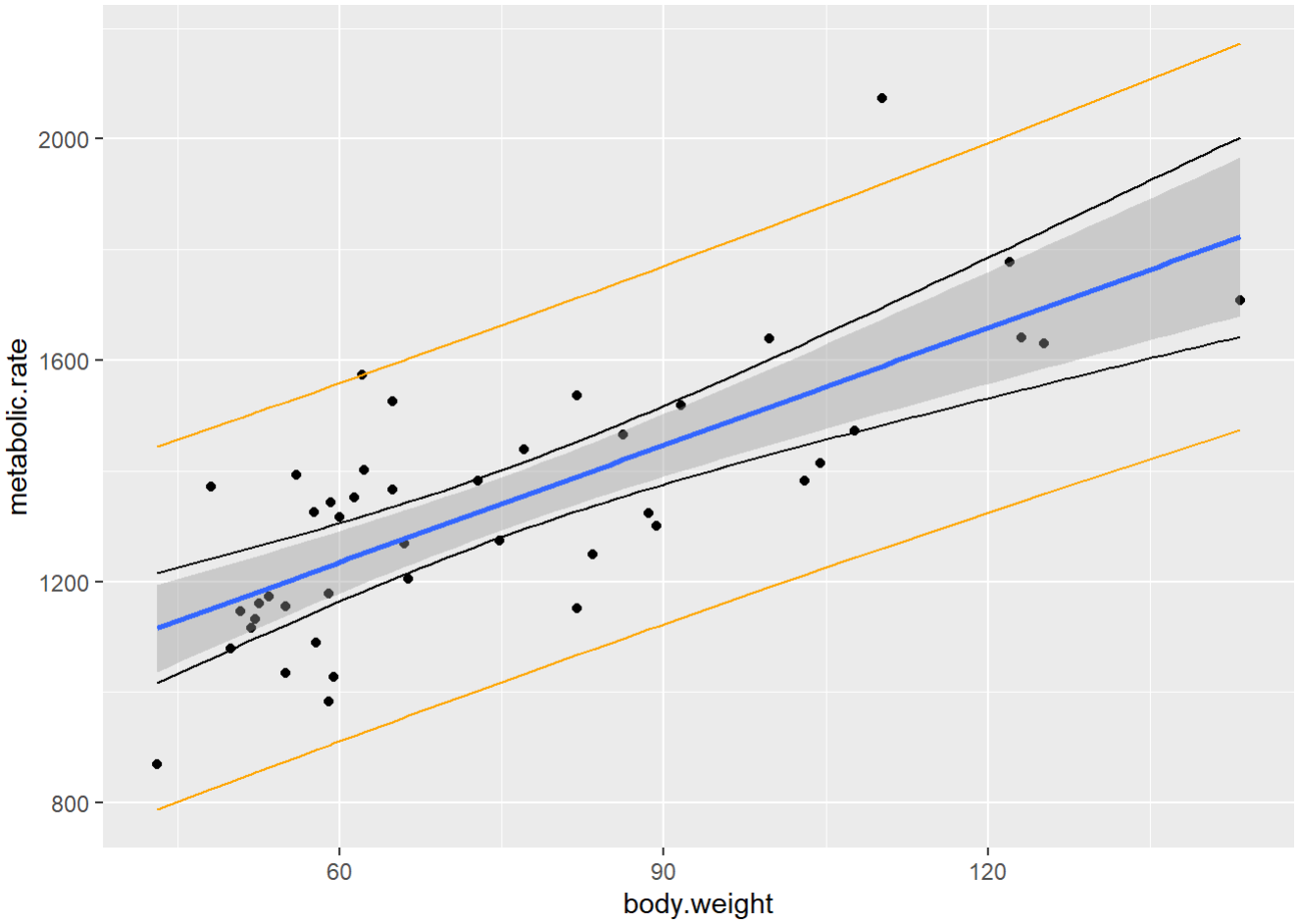
95% Prediction Interval: We are 95% confident that for a woman who weights 59 kg, her metabolic rate will be captured by the interval (904 kcal/24hr, 1552 kcal/24hr)

Comparison of the Confidence Intevals/Prediction Interval

```
# Prediction interval for all values in data set
predict(mrout, rmr, interval = "prediction", level = 0.95) %>%
  cbind(rmr) ->
  predintout
```

```
ggplot() +
  geom_point(data = rmr, mapping = aes(x = body.weight, y = metabolic.rate))+
  geom_smooth(data = rmr, mapping = aes(x = body.weight, y = metabolic.rate), method = 'lm', se
 = TRUE) +
  geom_line(data = newdf, mapping = aes(x = body.weight, y = lwr)) +
  geom_line(data = newdf, mapping = aes(x = body.weight, y = upr)) +
  geom_line(data = predintout, mapping = aes(x = body.weight, y = lwr), color = "orange")+
  geom_line(data = predintout, mapping = aes(x = body.weight, y = upr), color = "orange")
```

```
## `geom_smooth()` using formula 'y ~ x'
```


metabolic.rate
2000
1600
1200
800
60
90
120
body.weight