Workbook 2

Paul Edwards

03/03/2021

Fun and figures with R

1.

```
#Mathematical calculations
1+1
```

```
## [1] 2
```

```
2-2
```

```
## [1] 0
```

```
3*3
```

```
## [1] 9
```

```
4/4
```

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

```
(5^5)^5
```

```
## [1] 2.980232e+17
```

```
(6^6)^-6
```

```
## [1] 9.69516e-29
```

```
7^-7
```

```
## [1] 1.214266e-06
```

```
8.8*8.8
```

```
## [1] 77.44
```

```
99+(9/9)
```

```
## [1] 100
```

```
go_factorial <- function(x) {
  y <- 1
  for (i in 1:x) {
    y <- y * ((1:x)[i])
  }
  print(y)
}
go_factorial(10)
```

```
## [1] 3628800
```

2.

```
#Square root function
sqrt(324)
```

```
## [1] 18
```

3.

```
#Find row with max sepal length and subset dataframe
maxSL <- which.max(iris$Sepal.Length)
iris[maxSL,]
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
132	7.9	3.8	6.4	2	virginica

1 row

4.

```
#Find row with min sepal length and subset dataframe
minSL <- which.min(iris$Sepal.Length)
iris[minSL,]
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
14	4.3	3	1.1	0.1	setosa

1 row

5.

```
#Calculate range between maximum and minimum sepal length
rangeSL <- max(iris$Sepal.Length)-min(iris$Sepal.Length)
rangeSL
```

```
## [1] 3.6
```

6.

```
#Construct dataframe with 7 columns and 100 randomly generated rows
df1 <- data.frame(
  a = runif(100),
  b = runif(100),
  c = runif(100),
  d = runif(100),
  e = runif(100),
  f = runif(100),
  g = runif(100)
)
str(df1)
```

```
## 'data.frame':    100 obs. of  7 variables:
##  $ a: num  0.996 0.394 0.373 0.121 0.563 ...
##  $ b: num  0.0189 0.1252 0.9189 0.7864 0.7299 ...
##  $ c: num  0.871 0.863 0.671 0.327 0.28 ...
##  $ d: num  0.0541 0.417 0.3478 0.9938 0.123 ...
##  $ e: num  0.676 0.929 0.239 0.872 0.11 ...
##  $ f: num  0.9946 0.9735 0.5976 0.6235 0.0346 ...
##  $ g: num  0.6921 0.0526 0.7399 0.6519 0.641 ...
```

7.

```
#Copy dataframe and rename columns
df2 <- df1
names(df2)[] <- c("Doc","Grumpy","Happy","Sleepy","Bashful","Sneezy","Dopey")
head(df2)
```

	Doc <dbl>	Grumpy <dbl>	Happy <dbl>	Sleepy <dbl>	Bashful <dbl>	Sneezy <dbl>	Dopey <dbl>
1	0.9961749	0.01885487	0.8706613	0.05405949	0.6759843	0.99462367	0.69211787
2	0.3935370	0.12516160	0.8629563	0.41697197	0.9293223	0.97348509	0.05257585
3	0.3728479	0.91887237	0.6705147	0.34783416	0.2394163	0.59755586	0.73988932
4	0.1208364	0.78642778	0.3265937	0.99381317	0.8718778	0.62348518	0.65191548
5	0.5626295	0.72992503	0.2803583	0.12301862	0.1095953	0.03456399	0.64099757
6	0.2542582	0.19160038	0.7227453	0.89165768	0.9928319	0.05612350	0.01759766

6 rows

8.

```
#Run linear regression model of height by weight
mod1 <- lm(height ~ weight, data = women)
summary(mod1)
```

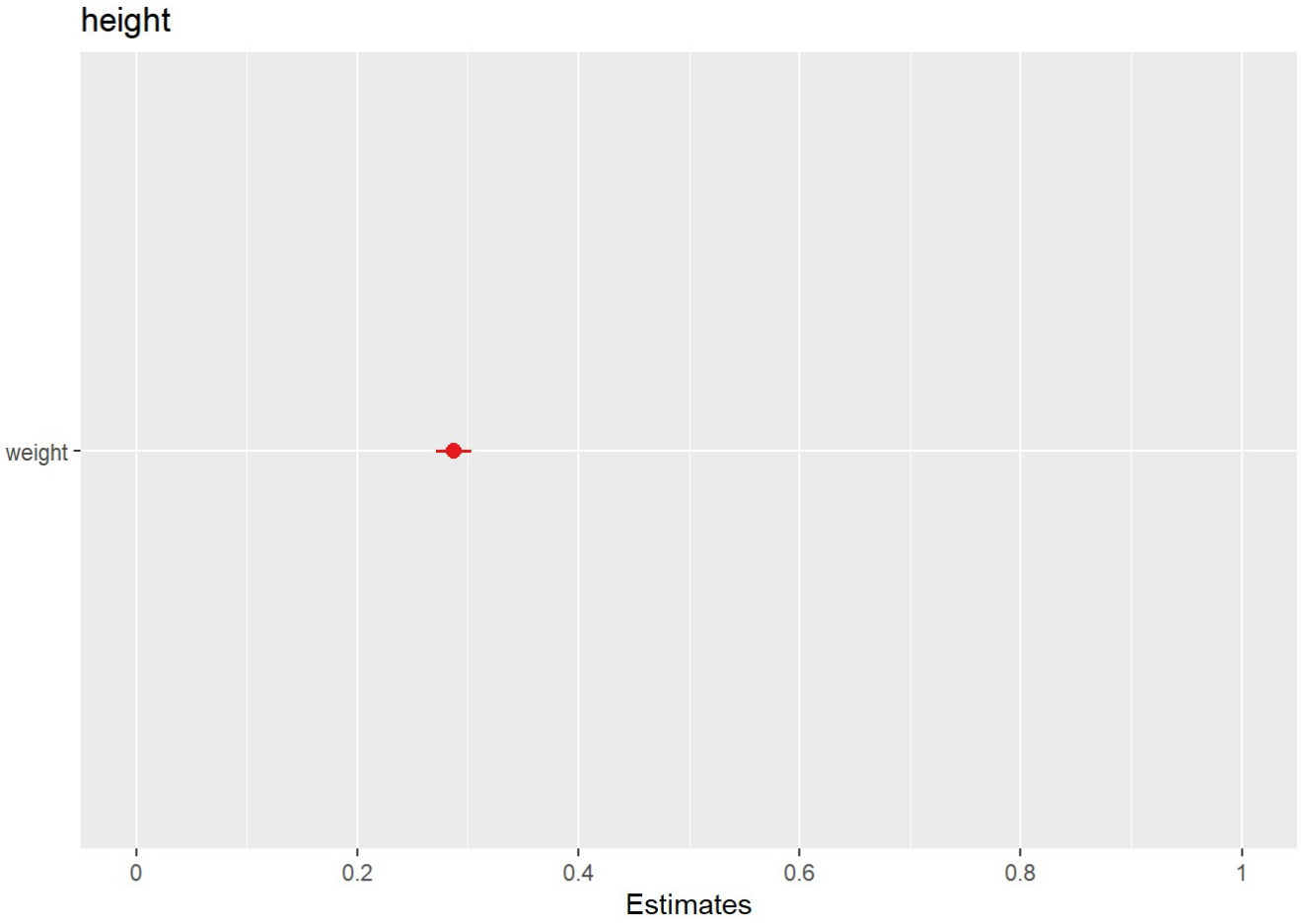
```
##
## Call:
## lm(formula = height ~ weight, data = women)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83233 -0.26249  0.08314  0.34353  0.49790
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 25.723456   1.043746   24.64 2.68e-12 ***
## weight       0.287249   0.007588   37.85 1.09e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.44 on 13 degrees of freedom
## Multiple R-squared:  0.991,  Adjusted R-squared:  0.9903
## F-statistic:  1433 on 1 and 13 DF,  p-value: 1.091e-14
```

9.

```
#Display model results in a table
sjPlot::tab_model(mod1)
```

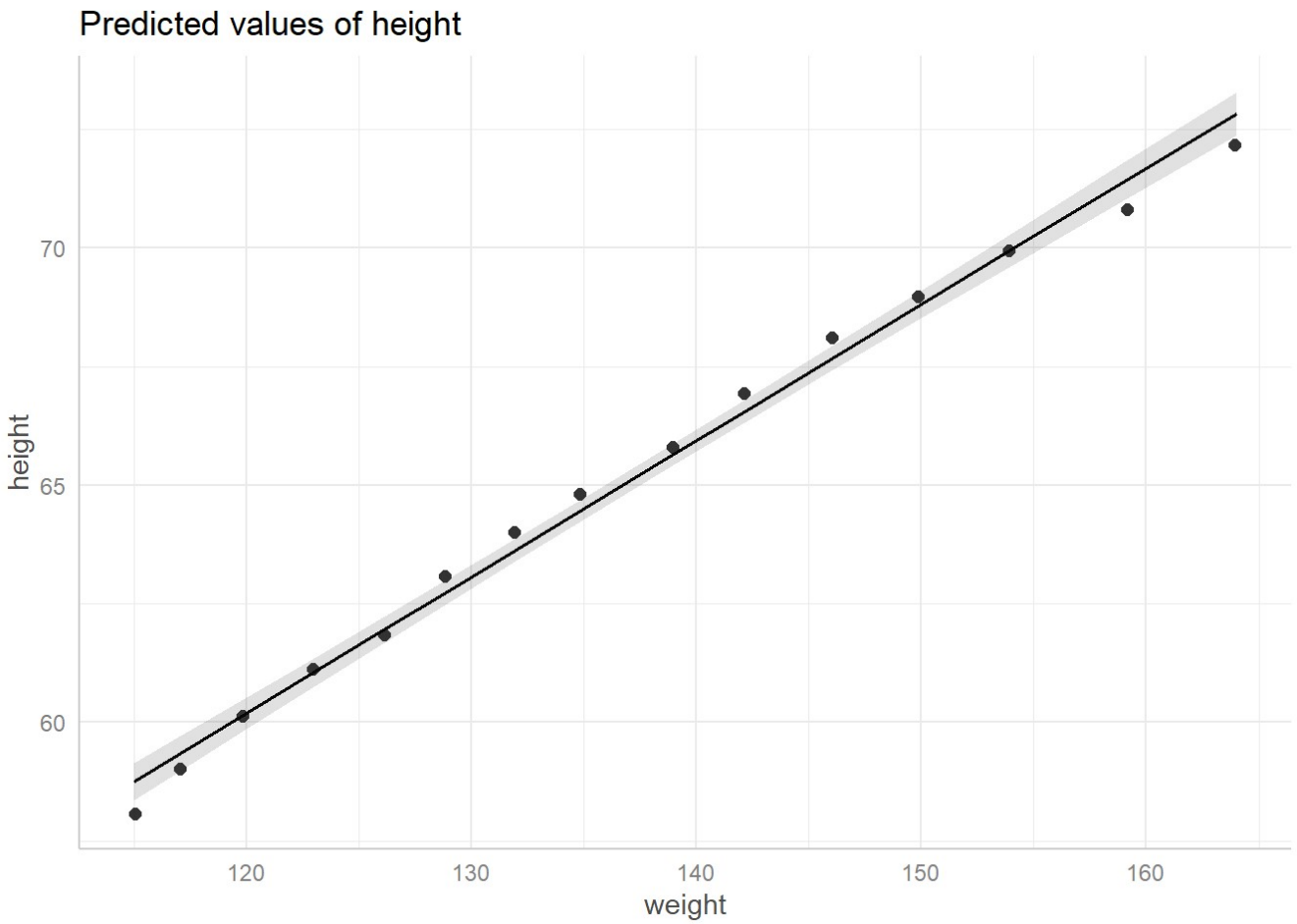
	height		
Predictors	*Estimates*	*CI*	*p*
(Intercept)	25.72	23.47 – 27.98	**<0.001**
weight	0.29	0.27 – 0.30	**<0.001**
Observations	15		
R^2 / R^2 adjusted	0.991 / 0.990		

```
#Display model results in a coefficient plot
sjPlot::plot_model(mod1)
```



10.

```
#Display model results in a prediction plot with observations
plot1 <- ggeffects::ggpredict(mod1, terms = "weight")
plot(plot1,
        add.data = TRUE,
        dot.alpha = .8,
        jitter = .2)
```



11.

```
#Example calculation
sum(women$weight > 140) / length(women$weight)
```

```
## [1] 0.4
```

This calculation is taking the number of women who weigh over 140 pounds, divided by the total number of women in the sample i.e. the proportion of women who weigh over 140 pounds.

12.

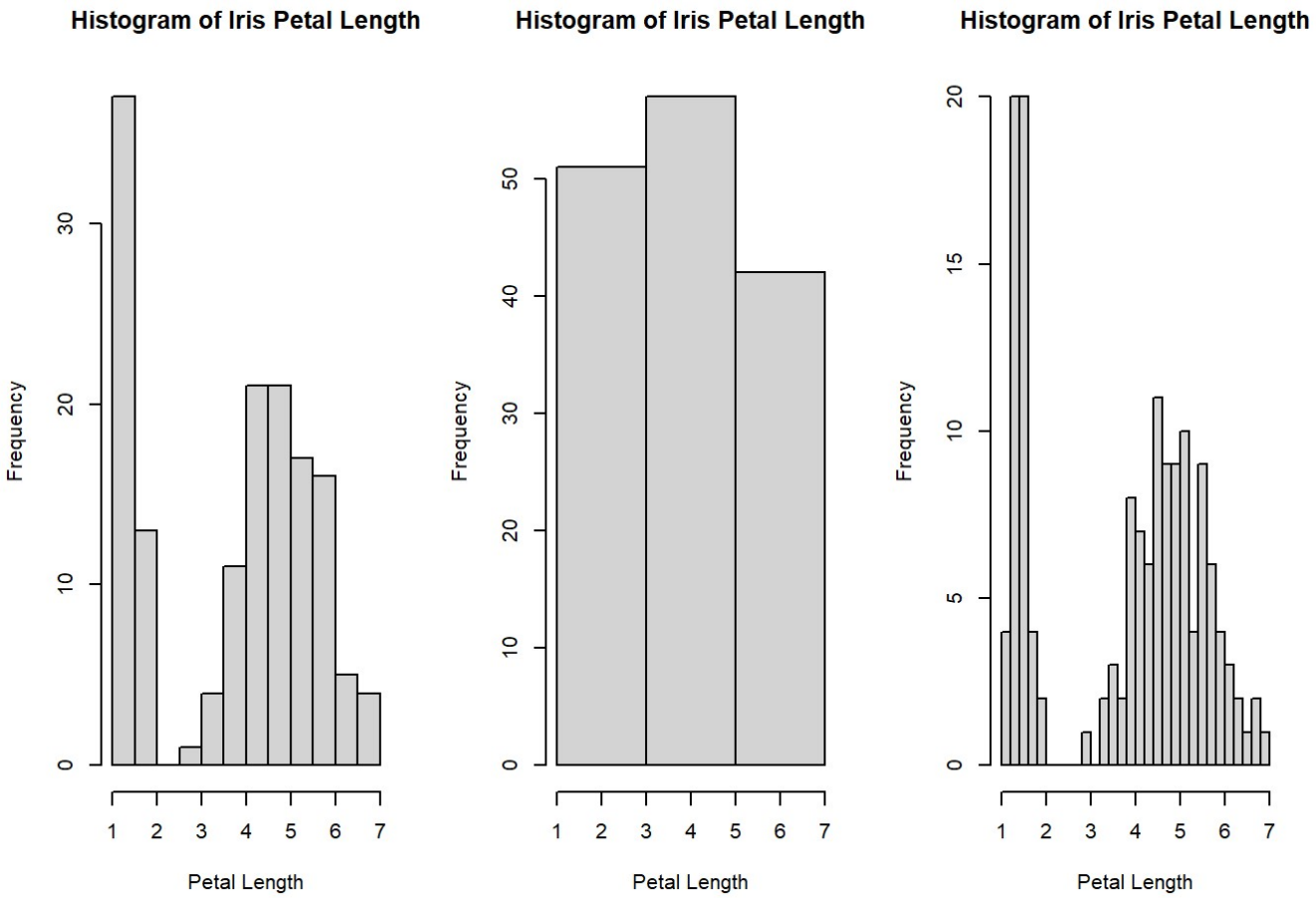
```
#Calculate proportion of women over the average weight
mwgt <- mean(women$weight)
sum(women$weight > mwgt) / length(women$weight)
```

```
## [1] 0.4666667
```

46.67% of women weigh over the mean weight of women in this sample.

13.

```
#Set plot area to row of 3 graphs
par(mfrow=c(1,3))
#Plot histograms with varying break length
hist(iris$Petal.Length,
    main="Histogram of Iris Petal Length",
    xlab="Petal Length")
hist(iris$Petal.Length,
    main="Histogram of Iris Petal Length",
    xlab="Petal Length",
    breaks=seq(1,7,2))
hist(iris$Petal.Length,
    main="Histogram of Iris Petal Length",
    xlab="Petal Length",
    breaks=seq(1,7,0.2))
```



Histogram breaks determine the size and number of bins, or columns in which data is displayed. Too few bins and interesting points within the distribution will be lost; too many bins will result in noisy, meaningless data.

14.

```
#Identify coding error
mh <- mean(women$height)
sum(women$weight > mh) / length(women$height)
```

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

This code has the height variable mis-typed. It is attempting to sum the weight of women with weights greater than the mean height. The two different scales mean that all women have weights greater than the mean height.

15.

```
#Transpose and rename dataframe columns
women2 <- data.frame(women[,c(2,1)])
names(women2)[] <- c("w","h")
str(women2)
```

```
## 'data.frame':    15 obs. of  2 variables:
##  $ w: num  115 117 120 123 126 129 132 135 139 142 ...
##  $ h: num  58 59 60 61 62 63 64 65 66 67 ...
```

16.

```
#Read data
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.0.3
```

```
testdata <- readr::read_csv(url("https://raw.githubusercontent.com/go-bayes/psych-447/main/data/testdata1.csv"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_double(),
##   weight = col_double(),
##   height = col_double()
## )
```

```
str(testdata)
```

```
## spec_tbl_df [100 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id    : num [1:100] 1 2 3 4 5 6 7 8 9 10 ...
##  $ weight: num [1:100] 80.5 87.8 95.7 74.6 68.3 ...
##  $ height: num [1:100] 153 182 188 142 137 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   weight = col_double(),
##   ..   height = col_double()
##   .. )
```

```
#Save data
library(here)
```

```
## Warning: package 'here' was built under R version 4.0.4
```

```
## here() starts at C:/Users/Paul/Desktop/Stuff/University/PSYC447/Weekly Journals-Workbooks/psyc447-workbooks
```

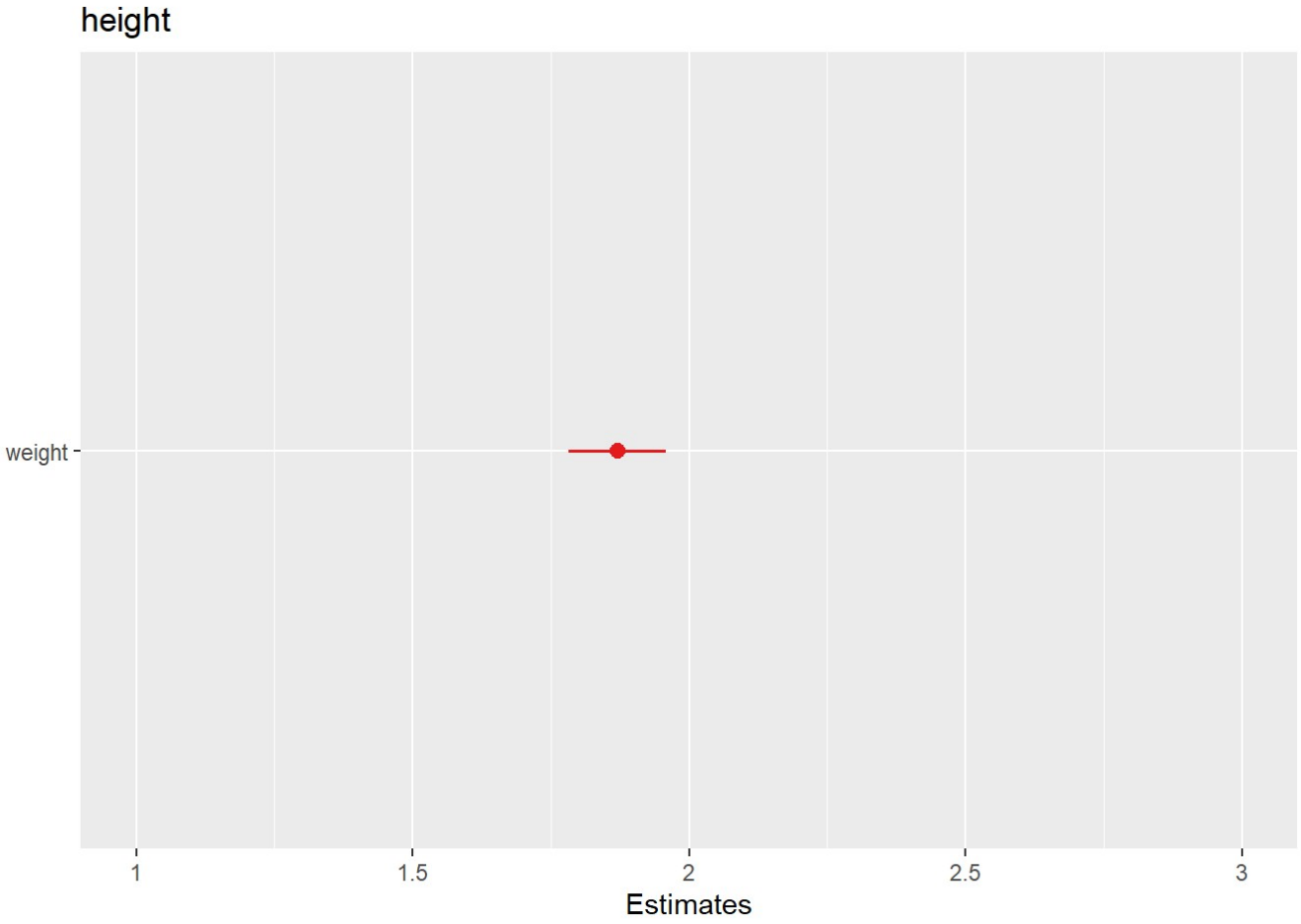
```
saveRDS(testdata, here::here("data", "td.RDS"))

td <- readRDS(here::here("data", "td.RDS"))
str(td)
```

```
## spec_tbl_df [100 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id    : num [1:100] 1 2 3 4 5 6 7 8 9 10 ...
##  $ weight: num [1:100] 80.5 87.8 95.7 74.6 68.3 ...
##  $ height: num [1:100] 153 182 188 142 137 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   weight = col_double(),
##   ..   height = col_double()
##   .. )
```

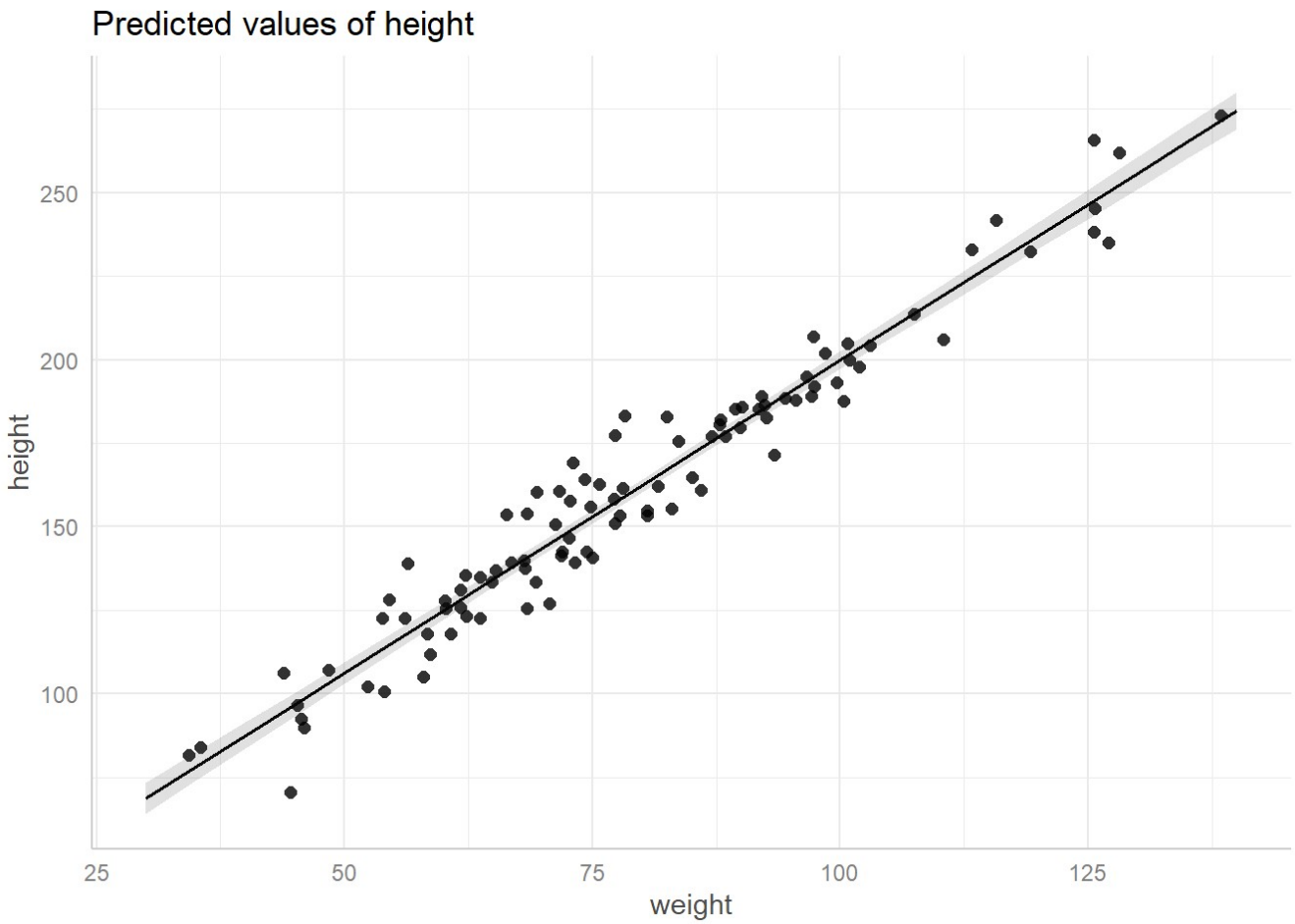
```
#Run linear regression model of height by weight
mod2 <- lm(height ~ weight, data = td)
summary(mod2)
```

```
##
## Call:
## lm(formula = height ~ weight, data = td)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.4761  -6.2215  -0.1467   4.8392  23.9751
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.74337    3.64557   3.496 0.000712 ***
## weight       1.87029    0.04432  42.201  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.683 on 98 degrees of freedom
## Multiple R-squared:  0.9478, Adjusted R-squared:  0.9473
## F-statistic:  1781 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
#Display model results in a coefficient plot
sjPlot::plot_model(mod2)
```




```
#Display model results in a prediction plot with observations
plot2 <- ggeffects::ggpredict(mod2, terms = "weight")
plot(plot2,
        add.data = TRUE,
        dot.alpha = .8,
        jitter = .2)
```



The intercept represents the predicted value of height when weight is zero. Here, the intercept is nonsensical as it would mean a height of 12.7cm given a weight of 0kg, which is generally not possible unless you are in space.