Quantitative Methods in Linguistics – Lecture 6

Adrian Brasoveanu[*]

April 9, 2014

Contents

1 Second attempt: two means, i.e., predicting y values based on x_2 values 1

2 The two-mean linear (regression) model 6

3 Analysis of Variance (ANOVA) 8

4 T-tests: the simple version of the F-test we can use for single categorical parameters 11

5 Third attempt: "many" means, i.e., predicting y values based on x_1 values 18

6 Comparing `reg1` with the 1-mean model 22

7 The intercept and slope coefficients 25

7.1 Alternative slopes . . . 26

7.2 Alternative intercepts . . . 27

7.3 Alternative intercepts and slopes . . . 28

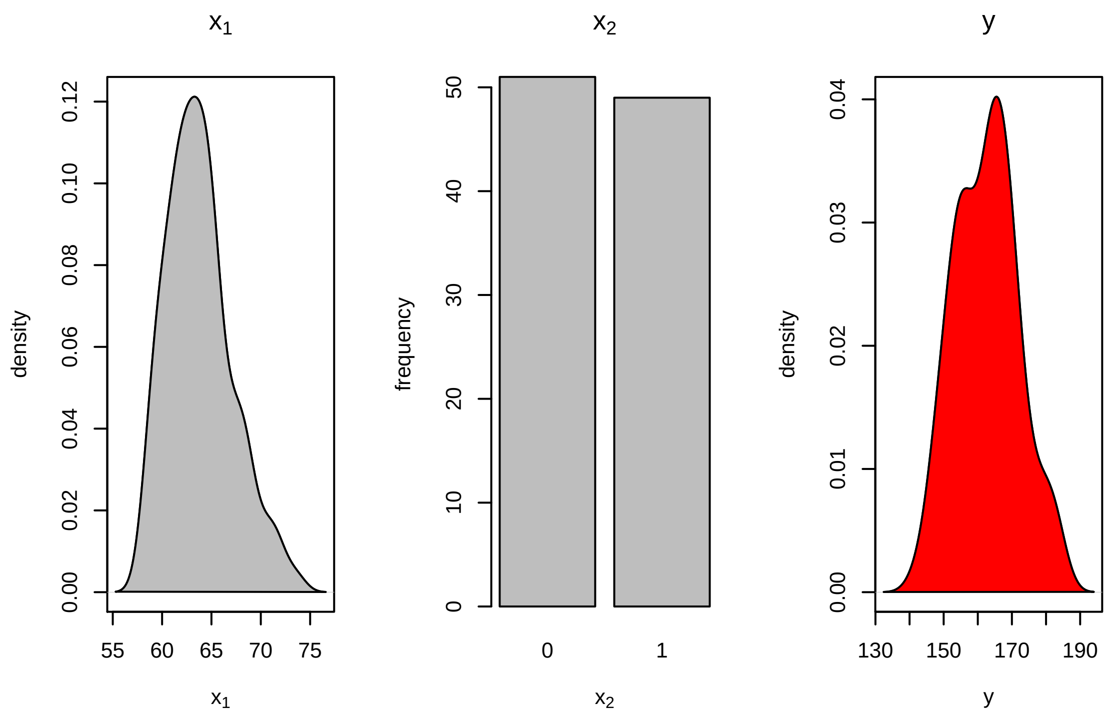
1 Second attempt: two means, i.e., predicting y values based on x_2 values

We generate the data again (it will be slightly different from the data set we generated before).

```
> x1 <- rnorm(100, 64, 3)
> summary(x1)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   58.3    61.5    63.5    63.8    65.3    73.6

> x2 <- rbinom(100, 1, 0.5)
> sum(x2)

[1] 49
```

[*]These notes have been generated with the 'knitr' package (Xie 2013) and are based on many sources, including but not limited to: Abelson (1995), Miles and Shevlin (2001), Faraway (2004), De Veaux et al. (2005), Braun and Murdoch (2007), Gelman and Hill (2007), Baayen (2008), Johnson (2008), Wright and London (2009), Gries (2009), Kruschke (2011), Diez et al. (2013), Gries (2013).

```
> y <- x1 * 2.5 + x2 * 6 + rnorm(100, 0, 4)
> summary(y)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    142     155     163     163     169     184

> par(mfrow = c(1, 3))
> plot(density(x1), xlab = expression(x[1]), main = expression(x[1]),
+     ylab = "density")
> polygon(density(x1), col = "gray", border = "black")
> plot(as.factor(x2), col = "gray", xlab = expression(x[2]), main = expression(x[2]),
+     ylab = "frequency")
> plot(density(y), xlab = expression(y), main = expression(y), ylab = "density")
> polygon(density(y), col = "red", border = "black")
```



```
> par(mfrow = c(1, 1))
```

We looked at the 1-mean model in the previous set of lecture notes:

```
> the_mean_model <- lm(y ~ 1)
> summary(the_mean_model)


Call:
lm(formula = y ~ 1)

Residuals:
    Min      1Q  Median      3Q     Max
-20.229  -7.381   0.395   6.169  21.435
```

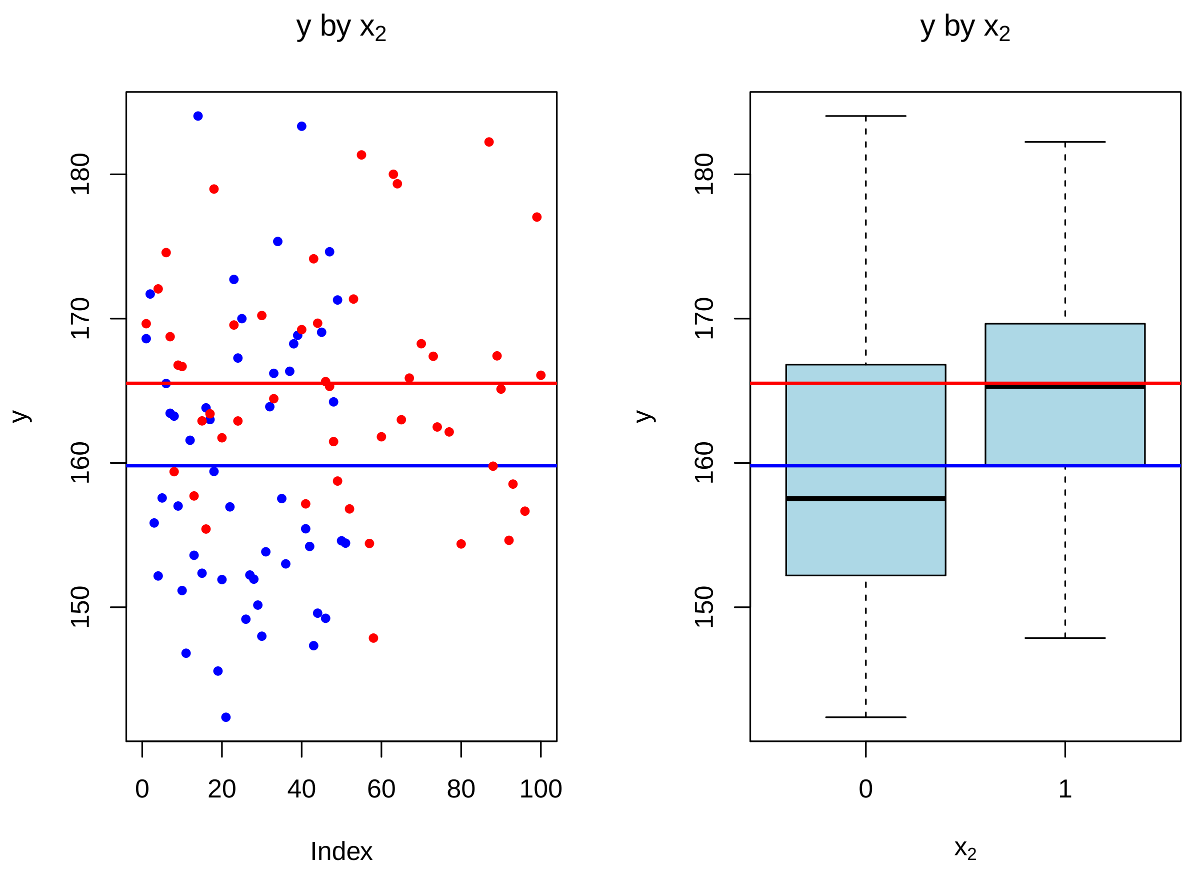
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  162.605      0.926     176   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.26 on 99 degrees of freedom
```

We now look at the two groups.

```
> y.x2 <- data.frame(y, x2)
> y.x2[1:10, ]

       y x2
1  169.7  1
2  168.6  0
3  171.7  0
4  172.1  1
5  155.8  0
6  174.6  1
7  168.8  1
8  159.4  1
9  166.8  1
10 166.7  1

> sum(x2)

[1] 49

> split.y.x2 <- split(y.x2, x2)
> (y.0 <- split.y.x2$"0")

       y x2
2  168.6  0
3  171.7  0
5  155.8  0
11 152.2  0
12 157.6  0
14 165.5  0
19 163.4  0
21 163.2  0
22 157.0  0
25 151.2  0
26 146.8  0
27 161.6  0
28 153.6  0
29 184.0  0
31 152.4  0
32 163.8  0
34 163.0  0
35 159.4  0
36 145.6  0
37 151.9  0
38 142.4  0
```

```
39 157.0  0
42 172.7  0
45 167.3  0
50 170.0  0
51 149.2  0
54 152.2  0
56 151.9  0
59 150.2  0
61 148.0  0
62 153.8  0
66 163.9  0
68 166.2  0
69 175.3  0
71 157.5  0
72 153.0  0
75 166.4  0
76 168.3  0
78 168.8  0
79 183.3  0
81 155.4  0
82 154.2  0
83 147.3  0
84 149.6  0
85 169.1  0
86 149.2  0
91 174.6  0
94 164.2  0
95 171.3  0
97 154.6  0
98 154.4  0

> (y.1 <- split.y.x2$"1")

        y x2
1   169.7  1
4   172.1  1
6   174.6  1
7   168.8  1
8   159.4  1
9   166.8  1
10  166.7  1
13  157.7  1
15  162.9  1
16  155.4  1
17  163.4  1
18  179.0  1
20  161.7  1
23  169.6  1
24  162.9  1
30  170.2  1
33  164.5  1
40  169.2  1
41  157.2  1
43  174.1  1
```

```
44  169.7  1
46  165.6  1
47  165.3  1
48  161.5  1
49  158.7  1
52  156.8  1
53  171.4  1
55  181.3  1
57  154.4  1
58  147.9  1
60  161.8  1
63  180.0  1
64  179.3  1
65  163.0  1
67  165.9  1
70  168.3  1
73  167.4  1
74  162.5  1
77  162.1  1
80  154.4  1
87  182.2  1
88  159.8  1
89  167.4  1
90  165.1  1
92  154.6  1
93  158.5  1
96  156.7  1
99  177.0  1
100 166.1  1

> nrow(y.1)

[1] 49

> sum(x2)

[1] 49

> par(mfrow = c(1, 2))
> plot(y.0$y, pch = 20, col = "blue", xlim = c(0, 100), ylim = range(y),
+     ylab = "y", main = expression(paste("y by ", x[2])))
> abline(mean(y.0$y), 0, col = "blue", lwd = 2)
> points(as.numeric(row.names(y.1)), y.1$y, pch = 20, col = "red")
> abline(mean(y.1$y), 0, col = "red", lwd = 2)
> round(mean(y.0$y), 3)

[1] 159.8

> round(mean(y.1$y), 3)

[1] 165.5

> round(mean(y.1$y) - mean(y.0$y), 3)

[1] 5.722

> plot(as.factor(x2), y, col = "lightblue", ylim = range(y), xlab = expression(x[2]),
+     ylab = "y", main = expression(paste("y by ", x[2])))
> abline(mean(y.0$y), 0, col = "blue", lwd = 2)
> abline(mean(y.1$y), 0, col = "red", lwd = 2)
```



```
> par(mfrow = c(1, 1))
```

Let's compare the errors for the 1-mean and the 2-mean models numerically:

```
> residuals <- y - mean(y)
> sum(residuals^2)

[1] 8488

> residuals.0 <- y.0$y - mean(y.0$y)
> residuals.1 <- y.1$y - mean(y.1$y)
> sum(c(residuals.0, residuals.1)^2)

[1] 7670
```

2 The two-mean linear (regression) model

Let's look at the linear model that has x_2 as a predictor in addition to the intercept:

```
> reg2 <- lm(y ~ 1 + x2)
> summary(reg2)
```

```
Call:
lm(formula = y ~ 1 + x2)

Residuals:
    Min      1Q  Median      3Q     Max 
-17.659  -6.844  -0.738   5.737  24.238 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   159.80       1.24  129.00   <2e-16 ***
x2              5.72       1.77    3.23   0.0017 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.85 on 98 degrees of freedom
Multiple R-squared:  0.0964,Adjusted R-squared:  0.0872 
F-statistic: 10.5 on 1 and 98 DF,  p-value: 0.00167
```

It is actually equivalent, and shorter, to write the `lm` formula without an explicit intercept. By default, the `lm` function will assume that we want to include the intercept:

```
> reg2 <- lm(y ~ x2)
> summary(reg2)


Call:
lm(formula = y ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max 
-17.659  -6.844  -0.738   5.737  24.238 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   159.80       1.24  129.00   <2e-16 ***
x2              5.72       1.77    3.23   0.0017 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.85 on 98 degrees of freedom
Multiple R-squared:  0.0964,Adjusted R-squared:  0.0872 
F-statistic: 10.5 on 1 and 98 DF,  p-value: 0.00167
```

We'll examine the `reg2` output more closely very soon. For now, let's understand how we obtain the intercept and x_2 coefficients.

```
> summary(reg2)$coef[2, 1:2]

  Estimate Std. Error 
     5.722      1.770 
```

The intercept of the `reg2` model is just the mean of the $x_2 = 0$ group, and the x_2 coefficient is the difference between the mean of the $x_2 = 1$ group and the intercept:

```
> mean(y.0$y)

[1] 159.8

> mean(y.1$y)

[1] 165.5

> mean(y.1$y) - mean(y.0$y)

[1] 5.722

> summary(reg2)$coef[1:2, 1:2]

            Estimate Std. Error
(Intercept)  159.802      1.239
x2             5.722      1.770
```

The SEs for these coefficient estimates are derived as shown below. Let's not worry about these formulas, we will understand them a bit better later on when we discuss the model that has x_1 as its sole predictor.

```
> sum_squared_errors_total <- sum((y.0$y - mean(y.0$y))^2) + sum((y.1$y -
+     mean(y.1$y))^2)
> variance_total <- sum_squared_errors_total/(length(y.0$y) + length(y.1$y) -
+     2)
> variance_total

[1] 78.27

> sd_total <- sqrt(variance_total)
> sd_total

[1] 8.847

> sum_squared_errors_predictor <- sum((x2 - mean(x2))^2)
> se_intercept <- sd_total/sqrt(length(y)) * sqrt(sum(x2^2)/sum_squared_errors_predictor)
> se_intercept

[1] 1.239

> se_x2 <- sd_total/sqrt(sum_squared_errors_predictor)
> se_x2

[1] 1.77

> summary(reg2)$coef[1:2, 1:2]

            Estimate Std. Error
(Intercept)  159.802      1.239
x2             5.722      1.770
```

3 Analysis of Variance (ANOVA)

We can compare the 2-mean model with the 1-mean model by doing an analysis of variance (ANOVA). That is, we will ask and answer the question:

- is the 2-mean model better than the 1-mean model, i.e., does keeping track of the variable x_2 significantly reduce the error?

```
> anova(the_mean_model, reg2)

Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ x2
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1     99 8488
2     98 7670  1       818 10.4 0.0017 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Let's unpack the output of the above call to `anova`. We first compute the squared error within the two groups, or simply, the *squared error within*:

```
> squared.error.2MEAN <- sum(residuals.0^2) + sum(residuals.1^2)
> squared.error.2MEAN

[1] 7670
```

We then compute the total squared error:

```
> squared.error.1MEAN <- sum(residuals^2)
> squared.error.1MEAN

[1] 8488
```

Finally, we compute the difference between the above two squared errors, which is the squared error between the two groups, or simply, the *squared error between*:

```
> squared.error.difference <- squared.error.1MEAN - squared.error.2MEAN
> round(squared.error.difference)

[1] 818
```

This difference between the two squared errors is the part of the total error that is accounted for / eliminated by grouping, i.e., by keeping track of the variable x_2.

We now compute the mean squared error within and the mean squared error between. We do this for the same reason we computed the mean squared error for the 1-mean model: we're trying to quantify the error of the *model*, which should be the same whether we have 100 or 1000 observations.

The mean squared error within is computed as follows:

```
> mean.squared.error.within <- squared.error.2MEAN/(length(y) - 2)
> mean.squared.error.within

[1] 78.27
```

We subtract 2 from the total number of observations because we estimated 2 means (2 parameters) as part of the computation of the squared error within. We therefore 'spent' two degrees of freedom (dof.s).

The mean squared error between is obtained by dividing the squared error between, i.e., the difference between total squared error and the squared error within, by the number of extra means / parameters we use when we compute the squared error within (2 parameters, i.e., the means for the 2 groups) relative to the means / parameters we use when we compute the total squared error (1 parameter, i.e., the grand mean):

```
> mean.squared.error.between <- squared.error.difference/(2 - 1)
> mean.squared.error.between

[1] 818.1
```

Now we reason as follows. If the variable x_2 does not have any effect, i.e., if there aren't actually 2 groups in the sample but only 1, the between-group error should be the same as the within-group error. We call this our *null hypothesis*, or H_0:

(1) H_0: mean squared error between = mean squared error within

This is our null hypothesis because we assume that whatever difference we see between the two groups is not actually there, but it is simply what we expect given 'regular' random variation between two (sub)samples from the same population.

It turns out that we need to formulate the null hypothesis in ratio form to be able to test it for *statistical significance*, i.e., to test how likely our sample is given the sole assumption of random variation:

(2) $$H_0\text{: } \frac{\text{mean squared error between}}{\text{mean squared error within}} = 1$$

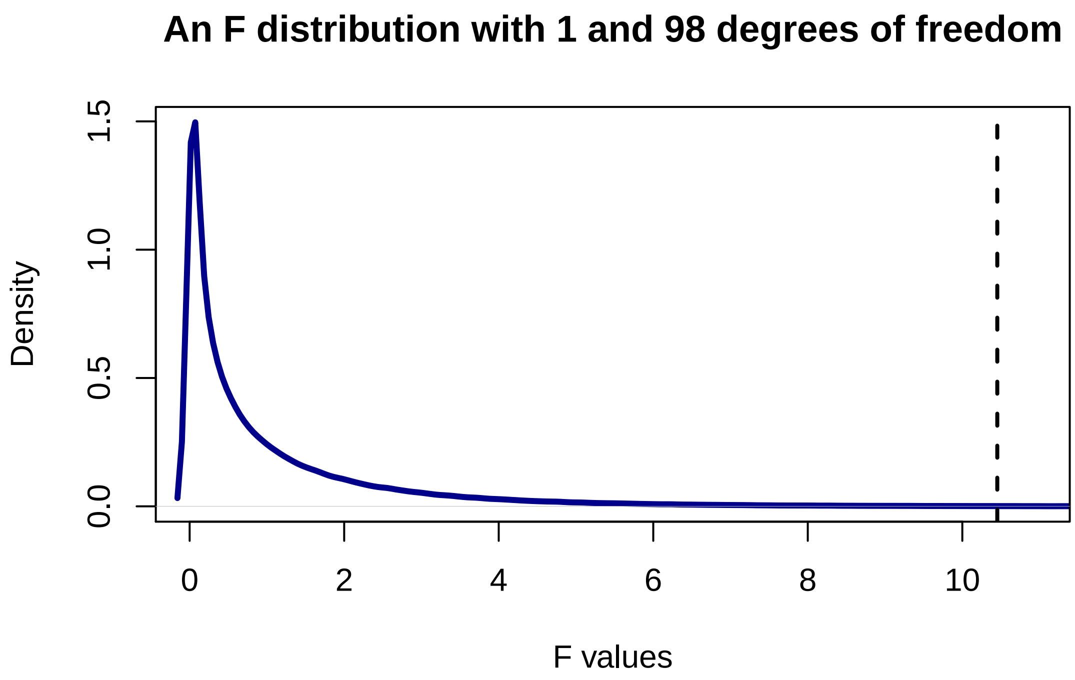
This ratio is our *test statistic*, i.e., the statistic / number that we calculate from our sample and that we test for statistical significance. This is the F value in the `anova` output.

```
> F_value <- mean.squared.error.between/mean.squared.error.within
> F_value

[1] 10.45
```

We call this an F value because we will test it against a probability distribution from the family of F distributions, also know as Snedecor's F distribution, or the FisherâĂŞSnedecor distribution (after Ronald Fisher and George Snedecor).

F distributions have two degree-of-freedom (dof) parameters, and they are exactly the numbers dof.s we used above to calculate the mean squared error between and the mean squared error within, namely 1 and 98 dof.s, respectively:

```
> plot(density(rf(1e+06, 1, 98)), lwd = 3, col = "darkblue", main = "An F distribution with 1 and 98 deg
+     xlab = "F values", xlim = c(0, F_value + 0.5))
> abline(v = F_value, lwd = 2, lty = 2)
```



We use the F distribution to estimate the probability of getting our F value (the ratio we obtained from our sample), or an even more extreme F value, under the null hypothesis that the F value is simply due to random variation and we don't really have two distinct groups / subpopulations in the population from which we drew our sample. This is the *p*-value (probability value) reported in the `anova` output:

```
> 1 - pf(mean.squared.error.between/mean.squared.error.within, 1, 98)

[1] 0.001669

> anova(the_mean_model, reg2)

Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ x2
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1     99 8488
2     98 7670  1       818 10.4 0.0017 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4 T-tests: the simple version of the F-test we can use for single categorical parameters

Let's examine the `reg2` output more closely and take stock:

```
> print(summary(reg2), dig = 3)
```

```
Call:
lm(formula = y ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max
-17.659  -6.844  -0.738   5.737  24.238

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   159.80       1.24  129.00   <2e-16 ***
x2              5.72       1.77    3.23   0.0017 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.85 on 98 degrees of freedom
Multiple R-squared:  0.0964,Adjusted R-squared:  0.0872
F-statistic: 10.5 on 1 and 98 DF,  p-value: 0.00167
```

First, the 5-number summary for the residuals is listed. The residuals are (basically) the same as the ones we computed by hand:

```
> print(summary(c(residuals.0, residuals.1)), dig = 3)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-17.700  -6.840  -0.738   0.000   5.740  24.200
```

We then have the coefficient estimates, and their SEs (we'll discuss t-values and their associated p-values in a moment):

```
> print(summary(reg2)$coef[, 1:2], dig = 3)

            Estimate Std. Error
(Intercept)   159.80       1.24
x2              5.72       1.77
```

The `intercept` estimate is the mean for the y responses for the $x_2 = 0$ group, while the x2 estimate is the difference between the y responses for the $x_2 = 0$ group and the y responses for the $x_2 = 1$ group. More precisely, it's the deflection of the mean for the $x_2 = 1$ group relative to the intercept, i.e., relative to the mean for the $x_2 = 0$ group:

```
> round(mean(y.0$y), 2)

[1] 159.8

> round(mean(y.1$y) - mean(y.0$y), 2)

[1] 5.72

> round(cumsum(summary(reg2)$coef[, 1]), 2)

(Intercept)          x2
      159.8       165.5

> round(c(mean(y.0$y), mean(y.1$y)), 2)

[1] 159.8 165.5
```

We then see the residual standard error and the associated degrees of freedom (dof.s). The residual standard error is the square root of the variance / mean squared error, and the variance is obtained by summing the squared residuals and dividing that sum by the dof.s, i.e., the number of observations (100) minus the number of means / parameters we estimated (2 means, one for each of the x_2 groups):

```
> residual.variance <- sum(c(residuals.0, residuals.1)^2)/(100 - 2)
> residual.sd <- sqrt(residual.variance)
> residual.sd

[1] 8.847

> summary(reg2)$sigma

[1] 8.847
```

We will discuss R-squared soon, let's just look at the last line of the `reg2` output. This reports the F-value (F-statistic), the dof.s and p-value we obtained when we compared the `reg2` model and the 1-mean-only / intercept-only model by means of an analysis of variance:

```
> summary(reg2)


Call:
lm(formula = y ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max 
-17.659  -6.844  -0.738   5.737  24.238 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   159.80       1.24  129.00   <2e-16 ***
x2              5.72       1.77    3.23   0.0017 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.85 on 98 degrees of freedom
Multiple R-squared:  0.0964,Adjusted R-squared:  0.0872 
F-statistic: 10.5 on 1 and 98 DF,  p-value: 0.00167

> anova(the_mean_model, reg2)

Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ x2
  Res.Df  RSS Df Sum of Sq    F Pr(>F)   
1     99 8488                            
2     98 7670  1       818 10.4 0.0017 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

An important thing to note is that the p-value obtained from the F-test / analysis of variance, is the same as the p-value associated with the x_2 coefficient estimate, and the t-value associated with that same coefficient is just the square root of our F-value:

```
> summary(reg2)$coef[2, ]

  Estimate Std. Error    t value   Pr(>|t|)
  5.721540   1.769724   3.233012   0.001669

> summary(reg2)$fstatistic

value numdf dendf
10.45  1.00 98.00

> c(sqrt(summary(reg2)$fstatistic[1]), summary(reg2)$coef[2, 3])

value
3.233 3.233

> c(1 - pf(summary(reg2)$fstatistic[1], summary(reg2)$fstatistic[2],
+     summary(reg2)$fstatistic[3]), summary(reg2)$coef[2, 4])

   value
0.001669 0.001669
```

That is, the t-value and *p*-value associated with a non-intercept coefficient are really just model comparisons between the model that includes that parameter and the model that is exactly the same except it does not include that parameter.

T-testing is just a simple version of F-testing, namely F-testing for two models that differ in only one parameter. We introduced F-tests first and we'll keep focusing on them because they can also be used to compare models that differ in more than one parameter. But we will show how we can compute the t-value and associated p-value directly.

The null hypothesis is not phrased in terms of a t-distribution of the difference between means:

(3) H_0: the difference between the group $x_2 = 0$ and $x_2 = 1$ is 0, that is, the two groups are not different

Again, this is our null hypothesis because we assume that whatever difference we see between the two groups is not actually there, but it is simply what we expect given 'regular' random variation between two (sub)samples from the same population.

So we want to determine how likely it would be to see a difference as big as we observed in our sample data or even bigger if the mean difference would really be 0, and it would be distributed according to a t-distribution with 98 dof.s (100 observations minus the 2 means we are estimating).

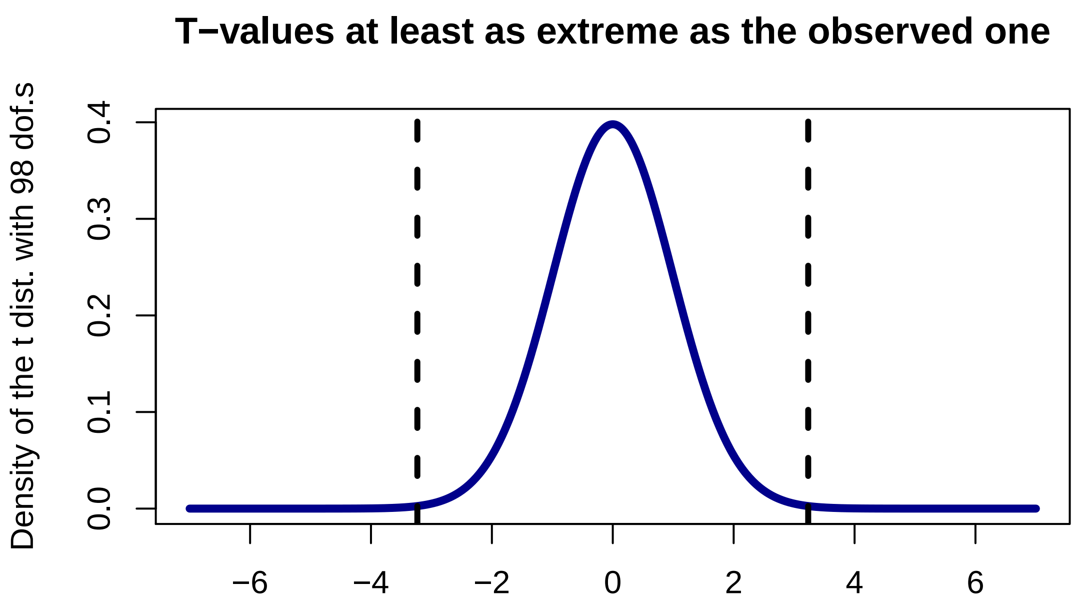
We do this by first computing the t-value based on our coefficient estimate and its SE, and then seeing how extreme that t-value given the t-distribution with the appropriate number of dof.s.

Let's compute the t-value first: this the coefficient estimate divided by its SE (this is an instance of 'standardization', we'll talk about it soon):

```
> summary(reg2)$coef[2, ]

  Estimate Std. Error    t value   Pr(>|t|)
  5.721540   1.769724   3.233012   0.001669

> (coef_x2 <- summary(reg2)$coef[2, 1])

[1] 5.722

> (SE_x2 <- summary(reg2)$coef[2, 2])

[1] 1.77

> (t_value_x2 <- coef_x2/SE_x2)

[1] 3.233
```

We now see how extreme this t-value is by checking how much probability there is under the t-distribution curve for values at least as extreme as our t-value (both negative and positive values):

```
> grid_points <- seq(-7, 7, length.out = 1000)
> plot(grid_points, dt(grid_points, df = 98), type = "l", lwd = 4, main = "T-values at least as extreme
+     xlab = "", ylab = "Density of the t dist. with 98 dof.s", col = "darkblue")
> abline(v = -t_value_x2, lty = 2, lwd = 3)
> abline(v = t_value_x2, lty = 2, lwd = 3)
```



The probability of obtaining t-values at least as extreme as the one we observe is the p-value reported by the `reg2` output:

```
> pt(-t_value_x2, df = 98)

[1] 0.0008347

> 1 - pt(t_value_x2, df = 98)

[1] 0.0008347

> pt(-t_value_x2, df = 98) + 1 - pt(t_value_x2, df = 98)

[1] 0.001669

> summary(reg2)$coef[2, 4]

[1] 0.001669
```

Since this probability is really small, we reject the null hypothesis.

Note however that t-values and associated p-values are not very informative compared to computing the 95% CI for the difference, which we compute given the coefficient estimate and the associated SE:

```
> coef_x2

[1] 5.722

> SE_x2

[1] 1.77

> c(coef_x2 + qt(0.025, df = 98) * SE_x2, coef_x2 + qt(0.975, df = 98) *
+     SE_x2)

[1] 2.210 9.234
```

Since this 95% CI for the difference (which is computed based on the same t-distribution as the above t-test) clearly excludes 0, we can very confidently reject the null hypothesis that the difference is actually 0.

But in addition to rejecting the null hypothesis, this 95% CI gives us a lot more information: it indicates what magnitude we can expect for the difference. The t-test and its t-value and p-value say nothing about this, they only tell us if a difference of 0 is plausible or not!

Thus, there are two reasons we don't care that much about t-tests:

- they are really special versions of F-tests, and the model comparisons they enable us to do are very limited compared to the generality of the F-test
- we can get a lot more information by computing a t-distribution based CI

There is a function explicitly dedicated to t-tests in R, whose output is similar to the `lm` based output:

```
> t.test(y.1$y, y.0$y, var.equal = T)


Two Sample t-test

data:  y.1$y and y.0$y
t = 3.233, df = 98, p-value = 0.001669
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.210 9.234
sample estimates:
mean of x mean of y
    165.5     159.8

> summary(reg2)


Call:
lm(formula = y ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max
-17.659  -6.844  -0.738   5.737  24.238

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   159.80       1.24  129.00   <2e-16 ***
x2              5.72       1.77    3.23   0.0017 **
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.85 on 98 degrees of freedom
Multiple R-squared:  0.0964,Adjusted R-squared:  0.0872
F-statistic: 10.5 on 1 and 98 DF,  p-value: 0.00167
```

Note that `t.test` computes the 95% CI for us.

But the only real reason for using the `t.test` function rather than the `lm` function is that the former does not necessarily assume that the variances of the two groups whose means we are comparing are equal:

```
> t.test(y.1$y, y.0$y, var.equal = F)


Welch Two Sample t-test

data:  y.1$y and y.0$y
t = 3.248, df = 94.88, p-value = 0.00161
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.224 9.219
sample estimates:
mean of x mean of y
    165.5     159.8
```

In our case, the variances are definitely equal – our responses y incorporate the same Gaussian noise (standard deviation = 4, hence variance = 4^2) for both the $x_2 = 0$ and the $x_2 = 1$ group.

So the results of running the t-test with the assumption of equal variances (which is what `lm` always does) or without that assumption are very similar.

Finally, a question for you: what is the t-test / hypothesis test for the intercept that is reported in the `reg2` output and why is that test completely unrealistic, which makes the resulting p-value completely uninteresting / uninformative despite the fact that it is very highly significant?

```
> summary(reg2)


Call:
lm(formula = y ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max
-17.659  -6.844  -0.738   5.737  24.238

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   159.80       1.24  129.00   <2e-16 ***
x2              5.72       1.77    3.23   0.0017 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.85 on 98 degrees of freedom
Multiple R-squared:  0.0964,Adjusted R-squared:  0.0872
F-statistic: 10.5 on 1 and 98 DF,  p-value: 0.00167
```

You should now also read chapter 8 of Gonick and Smith (1993), and at least skim chapter 9.

5 Third attempt: "many" means, i.e., predicting y values based on x_1 values

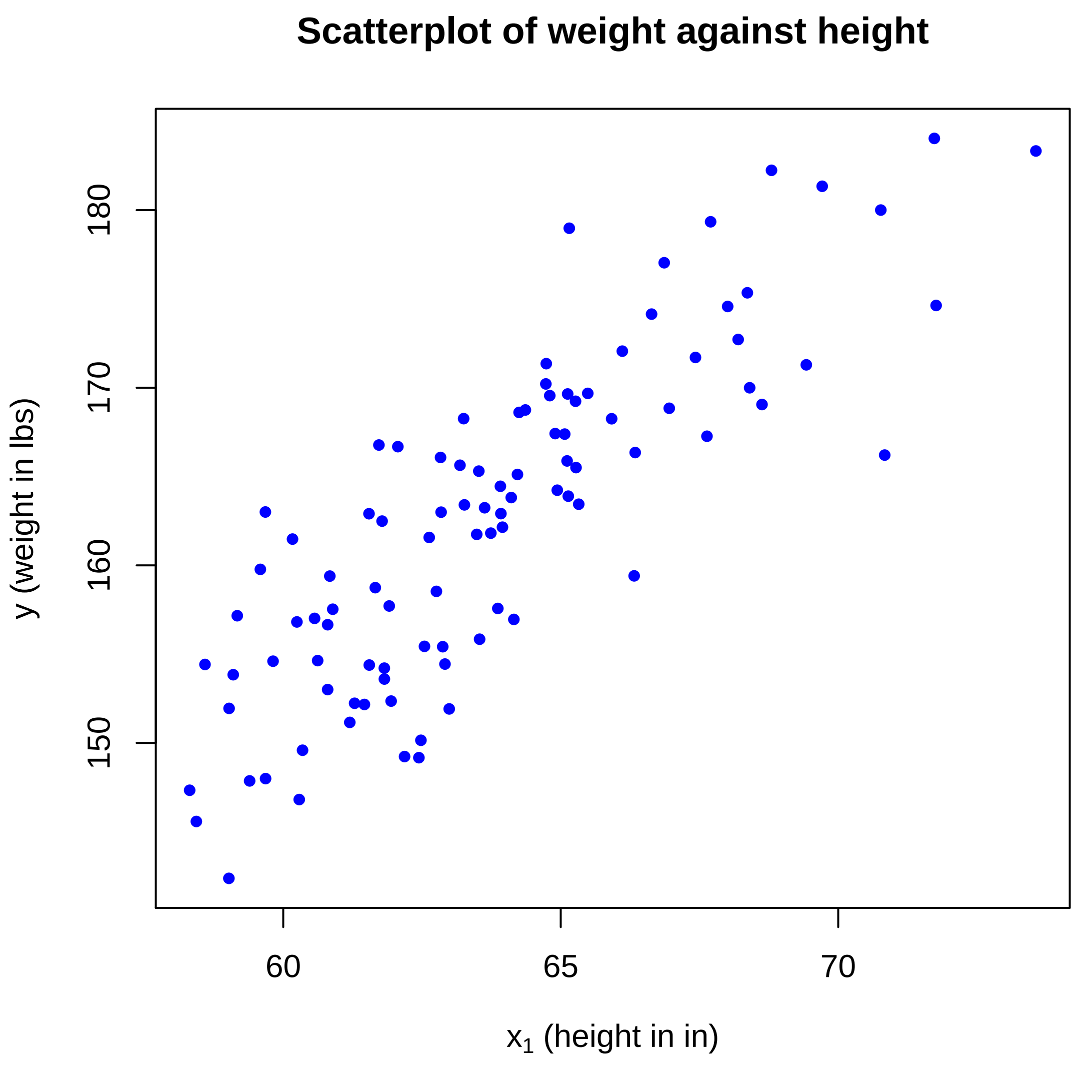
Note: "many means" is just a pedagogical aid. Making the continuous variable x_1 categorical (with whatever number of bins) is usually not a good idea.

What is really happening here is that there is *one dependency* of the response y on the *continuous* variable x_1, and this is how we will treat this mathematically (under the hood). For example, estimating the coefficient for x_1 will only 'consume' one dof, not "many" dof.s.

But it might be intuitively useful to think of the continuous variable x_1 as providing many y means, one for each possible value of x_1.

Before we run our linear (regression) model for weight (y) with height (x_1) as its only predictor, let's take a look at the data. We plot weight (y) against height (x_1):

```
> plot(x1, y, pch = 20, col = "blue", xlab = expression(paste(x[1],
+     " (height in in)")), ylab = "y (weight in lbs)", main = "Scatterplot of weight against height")
```



What to look for in a plot:

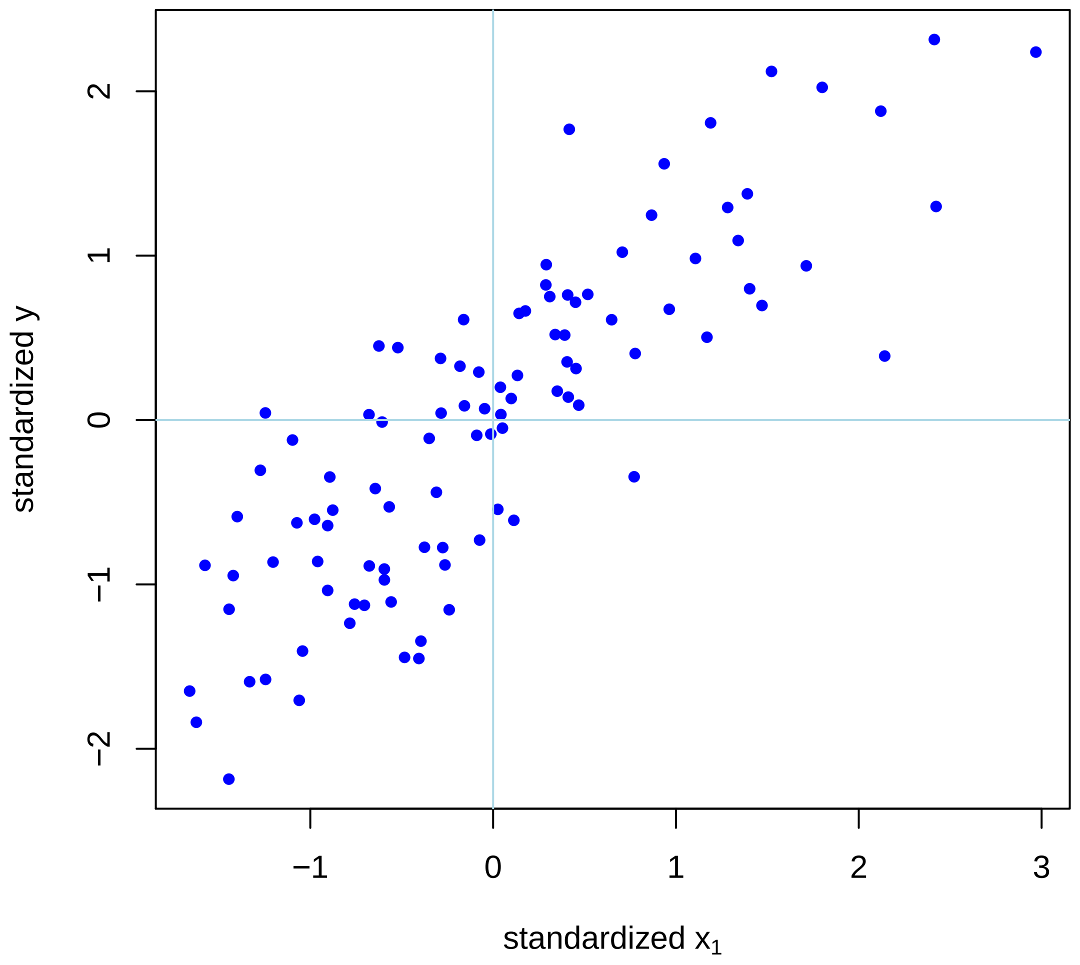
- direction of the cloud of points
- form / shape: linear, curving once, curving twice etc.
- amount of scatter
- outliers

It is even better to examine the standardized plot:

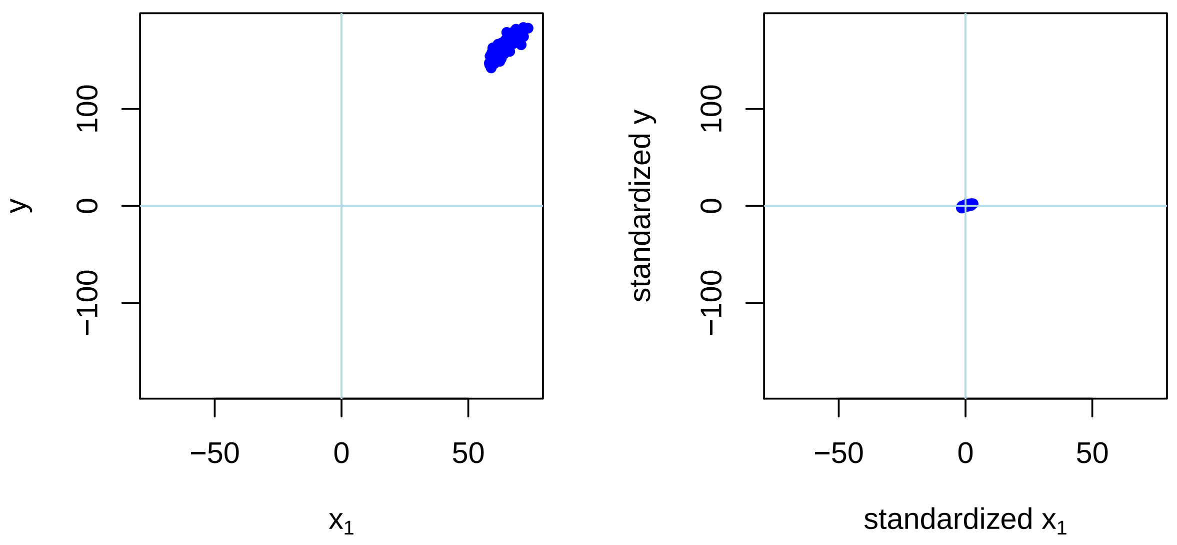
(4) Standardized plot: center both variables at their respective means and rescale variables by their respective standard deviations; see the discussion of the z-transform in Gonick and Smith (1993), chapter 5.

Standardization centers the plot at the origin and ensures that the two axes have the same unit of measurement:

```
> plot((x1 - mean(x1))/sd(x1), (y - mean(y))/sd(y), pch = 20, col = "blue",
+     xlab = expression(paste("standardized ", x[1])), ylab = "standardized y")
> abline(v = 0, col = "lightblue")
> abline(h = 0, col = "lightblue")
```



```
> par(mfrow = c(1, 2))
> plot(x1, y, pch = 20, col = "blue", xlim = range(c(x1, -x1)), ylim = range(c(y,
+     -y)), xlab = expression(x[1]))
> abline(v = 0, col = "lightblue")
> abline(h = 0, col = "lightblue")
> plot((x1 - mean(x1))/sd(x1), (y - mean(y))/sd(y), pch = 20, col = "blue",
+     xlab = expression(paste("standardized ", x[1])), ylab = "standardized y",
+     xlim = range(c(x1, -x1)), ylim = range(c(y, -y)))
> abline(v = 0, col = "lightblue")
> abline(h = 0, col = "lightblue")
```



```
> par(mfrow = c(1, 1))
```

Let’s now run our linear regression model:

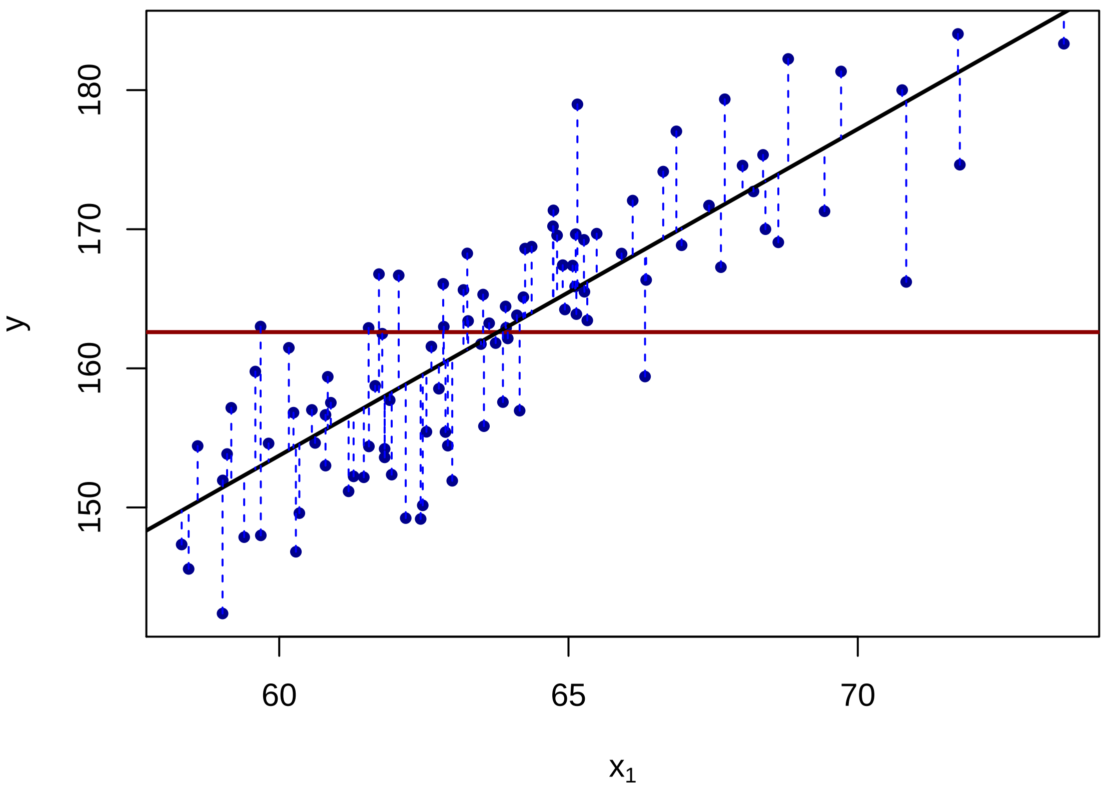
```
> reg1 <- lm(y ~ x1)
> summary(reg1)


Call:
lm(formula = y ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max 
-12.956  -4.385   0.526   3.746  13.156 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   12.893      9.976    1.29      0.2    
x1             2.347      0.156   15.03   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.12 on 98 degrees of freedom
Multiple R-squared:  0.697,Adjusted R-squared:  0.694 
F-statistic:  226 on 1 and 98 DF,  p-value: <2e-16
```

Just as before, the coefficients, and in particular the x_1 coefficient, a.k.a. the slope, are chosen so that the sum of squared errors is minimized.

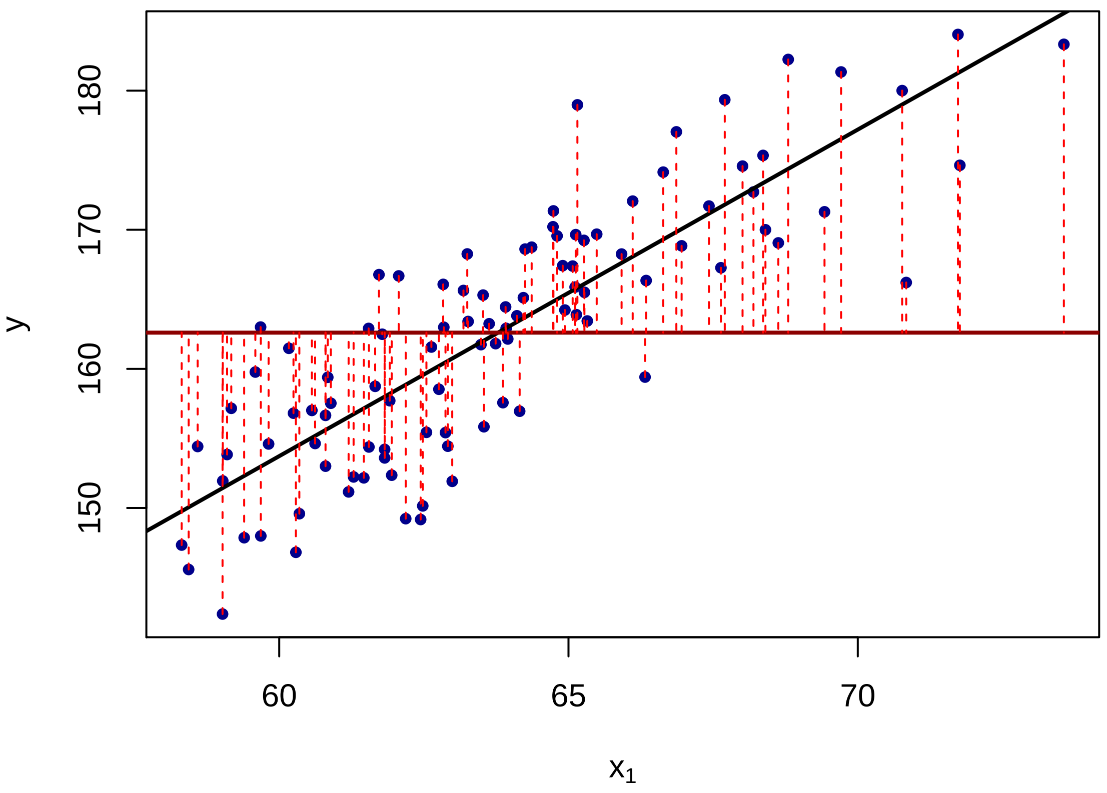
6 Comparing `reg1` with the 1-mean model

We will now compare the `reg1` and the 1-mean / intercept-only model. We can compare them visually by plotting the regression line, the mean line and the errors associated with each line, as follows:

```
> par(mfrow = c(2, 1), mai = c(1.02, 0.82, 0.22, 0.22))
> plot(x1, y, pch = 20, col = "darkblue", xlab = expression(x[1]))
> abline(mean(y), 0, col = "darkred", lwd = 2)
> abline(reg1, lwd = 2)
> segments(x1, y, x1, reg1$fitted.values, col = "blue", lty = 2)
> plot(x1, y, pch = 20, col = "darkblue", xlab = expression(x[1]))
> abline(reg1, lwd = 2)
> abline(mean(y), 0, col = "darkred", lwd = 2)
> segments(x1, y, x1, mean(y), col = "red", lty = 2)
```



y
180
170
160
150
60
65
70
x_1



y
180
170
160
150
60
65
70
x_1

```
> par(mfrow = c(1, 1), mai = c(1.02, 0.82, 0.82, 0.42))
```

Visually, we see that the total length of the segments in the first plot (which are the errors for the `reg1` model) is smaller than the total length of the segments in the second plot (the errors for the intercept-only model). When we compare the squared errors, what we do geometrically is compare areas, not just lengths – and if the difference in lengths is so clear, the difference in areas (not plotted) would be even more striking.

So let's turn to an ANOVA comparing the two models in terms of their squared errors:

```
> anova(the_mean_model, reg1)

Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ x1
  Res.Df  RSS Df Sum of Sq   F Pr(>F)
1     99 8488
2     98 2569  1      5919 226 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the p-value for the x_1 coefficient (i.e., the slope of the regression line) is the same in the ANOVA and the regression output. This value indicates whether the slope value is statistically significant, i.e., whether the model that keeps track of the x_1 values is better than the simplest possible model, which is the 1-mean / intercept-only model.

Let's compute the numbers reported in the ANOVA table by hand so that we can understand better what they mean. First, this is the squared error within, i.e., within the model / not accounted for by the model:

```
> squared.error.reg1 <- round(sum((summary(reg1)$residuals)^2))
> squared.error.reg1

[1] 2569
```

This is the total squared error:

```
> squared.error.1MEAN <- sum((y - mean(y))^2)
> round(squared.error.1MEAN)

[1] 8488
```

Therefore, the squared error between, i.e., the part of the total error that is accounted for / eliminated by keeping track of the variable x_1, is:

```
> squared.error.difference <- squared.error.1MEAN - squared.error.reg1
> squared.error.difference

[1] 5919
```

We can now compute the mean squared errors within and between that we need to get the F-ratio / F-value / F-statistic for our sample:

```
> mean.squared.error.within <- squared.error.reg1/(length(y) - 2)
> mean.squared.error.within

[1] 26.21
```

```
> mean.squared.error.between <- squared.error.difference/(2 - 1)
> mean.squared.error.between

[1] 5919
```

If the variable x_1 does not have any effect, the between error should be the same as the within error:

(5) H_0: mean squared error between = mean squared error within

(6) H_0 (in ratio form): $\frac{\text{mean squared error between}}{\text{mean squared error within}} = 1$

This ratio is our test statistic, namely the F-value in the anova output:

```
> mean.squared.error.between/mean.squared.error.within

[1] 225.8
```

This is an extremely large F-value relative to the F-distribution with 1 and 98 degrees of freedom (see the plot above for this distribution). The p-value is therefore very small and we are very confident that we can reject the null hypothesis. We therefore conclude that x_1 (height) is a highly significant predictor of y (weight) given our sample.

```
> 1 - pf(mean.squared.error.between/mean.squared.error.within, 1, 98)

[1] 0

> anova(reg1)

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value Pr(>F)
x1         1   5919    5919     226 <2e-16 ***
Residuals 98   2569      26
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7 The intercept and slope coefficients

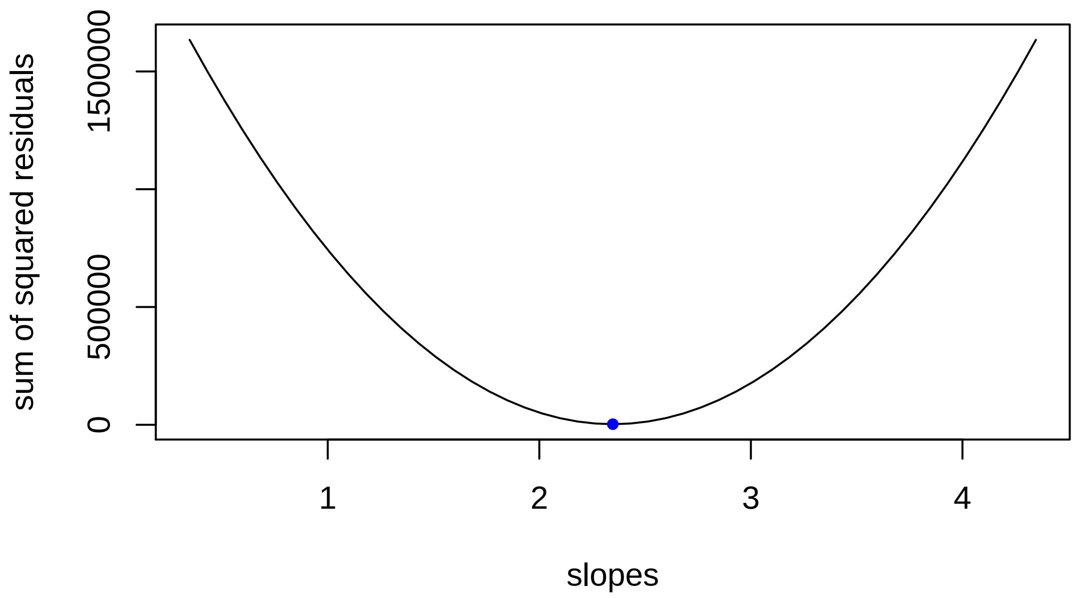
The coefficients for the intercept and the x_1 variable, i.e., slope, are chosen so that they minimize the squared error. This general principle applies here in the same way as for the previous two models (the intercept-only model and the `reg2` model).

```
> summary(reg1)$coefficients

            Estimate Std. Error t value  Pr(>|t|)
(Intercept)   12.893     9.9761   1.292 1.993e-01
x1             2.347     0.1562  15.027 3.529e-27

> (reg1.slope <- summary(reg1)$coefficients[2, 1])

[1] 2.347

> (reg1.intercept <- summary(reg1)$coefficients[1, 1])

[1] 12.89
```

7.1 Alternative slopes

We can see this by looking at alternative slopes and computing the squared error associated with them.

```
> (alternative.slopes <- seq(reg1.slope - 2, reg1.slope + 2, length.out = 49))

 [1] 0.3473 0.4306 0.5139 0.5973 0.6806 0.7639 0.8473 0.9306 1.0139 1.0973
[11] 1.1806 1.2639 1.3473 1.4306 1.5139 1.5973 1.6806 1.7639 1.8473 1.9306
[21] 2.0139 2.0973 2.1806 2.2639 2.3473 2.4306 2.5139 2.5973 2.6806 2.7639
[31] 2.8473 2.9306 3.0139 3.0973 3.1806 3.2639 3.3473 3.4306 3.5139 3.5973
[41] 3.6806 3.7639 3.8473 3.9306 4.0139 4.0973 4.1806 4.2639 4.3473

> alternative.slopes[25] == reg1.slope

[1] TRUE

> (errors.for.alternatives <- numeric(length = 49))

 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0

> for (i in 1:49) {
+     errors.for.alternatives[i] <- sum((y - (reg1.intercept + (alternative.slopes[i] *
+         x1)))^2)
+ }
> errors.for.alternatives

 [1] 1634119 1500989 1373524 1251725 1135590 1025120  920316  821177
 [9]  727702  639893  557749  481270  410457  345308  285824  232006
[17]  183852  141364  104541   73383   47890   28062   13899    5401
[25]    2569    5401   13899   28062   47890   73383  104541  141364
[33]  183852  232006  285824  345308  410457  481270  557749  639893
[41]  727702  821177  920316 1025120 1135590 1251725 1373524 1500989
[49] 1634119

> plot(alternative.slopes, errors.for.alternatives, type = "l", ylab = "sum of squared residuals",
+     xlab = "slopes")
> points(reg1.slope, sum((summary(reg1)$residuals)^2), pch = 20, col = "blue")
```

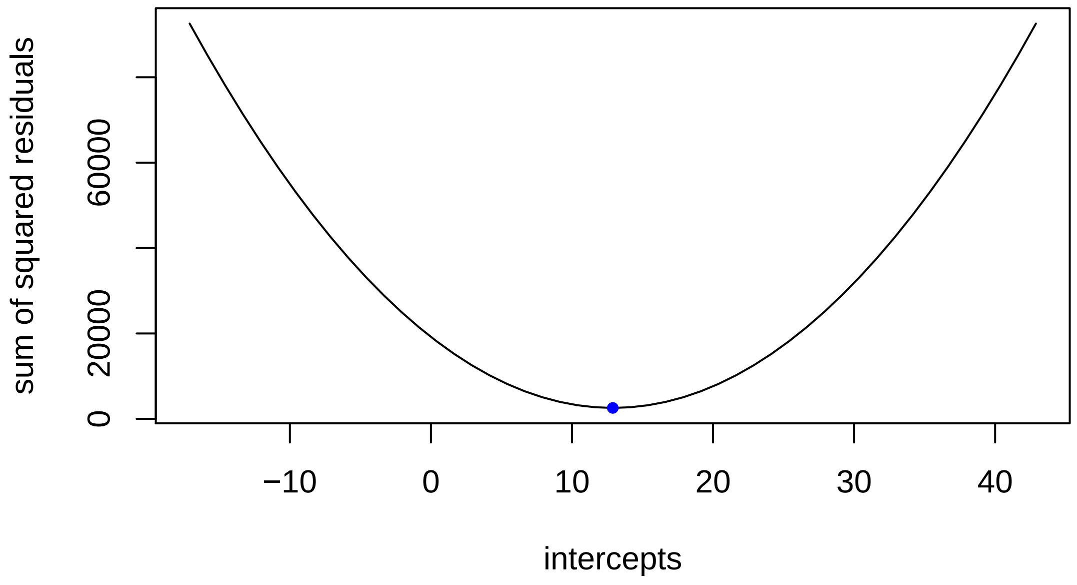


7.2 Alternative intercepts

Similarly, we can look at alternative intercepts:

```
> (alternative.intercepts <- seq(reg1.intercept - 30, reg1.intercept +
+     30, length.out = 49))

 [1] -17.1072 -15.8572 -14.6072 -13.3572 -12.1072 -10.8572  -9.6072
 [8]  -8.3572  -7.1072  -5.8572  -4.6072  -3.3572  -2.1072  -0.8572
[15]   0.3928   1.6428   2.8928   4.1428   5.3928   6.6428   7.8928
[22]   9.1428  10.3928  11.6428  12.8928  14.1428  15.3928  16.6428
[29]  17.8928  19.1428  20.3928  21.6428  22.8928  24.1428  25.3928
[36]  26.6428  27.8928  29.1428  30.3928  31.6428  32.8928  34.1428
[43]  35.3928  36.6428  37.8928  39.1428  40.3928  41.6428  42.8928

> alternative.intercepts[25] == reg1.intercept

[1] TRUE

> (errors.for.alternatives <- numeric(length = 49))

 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0

> for (i in 1:49) {
+     errors.for.alternatives[i] <- sum((y - (alternative.intercepts[i] +
+         (reg1.slope * x1)))^2)
+ }
> errors.for.alternatives
```

```
 [1] 92569 85225 78194 71475 65069 58975 53194 47725 42569 37725 33194
[12] 28975 25069 21475 18194 15225 12569 10225  8194  6475  5069  3975
[23]  3194  2725  2569  2725  3194  3975  5069  6475  8194 10225 12569
[34] 15225 18194 21475 25069 28975 33194 37725 42569 47725 53194 58975
[45] 65069 71475 78194 85225 92569

> plot(alternative.intercepts, errors.for.alternatives, type = "l",
+     ylab = "sum of squared residuals", xlab = "intercepts")
> points(reg1.intercept, sum((summary(reg1)$residuals)^2), pch = 20,
+     col = "blue")
```

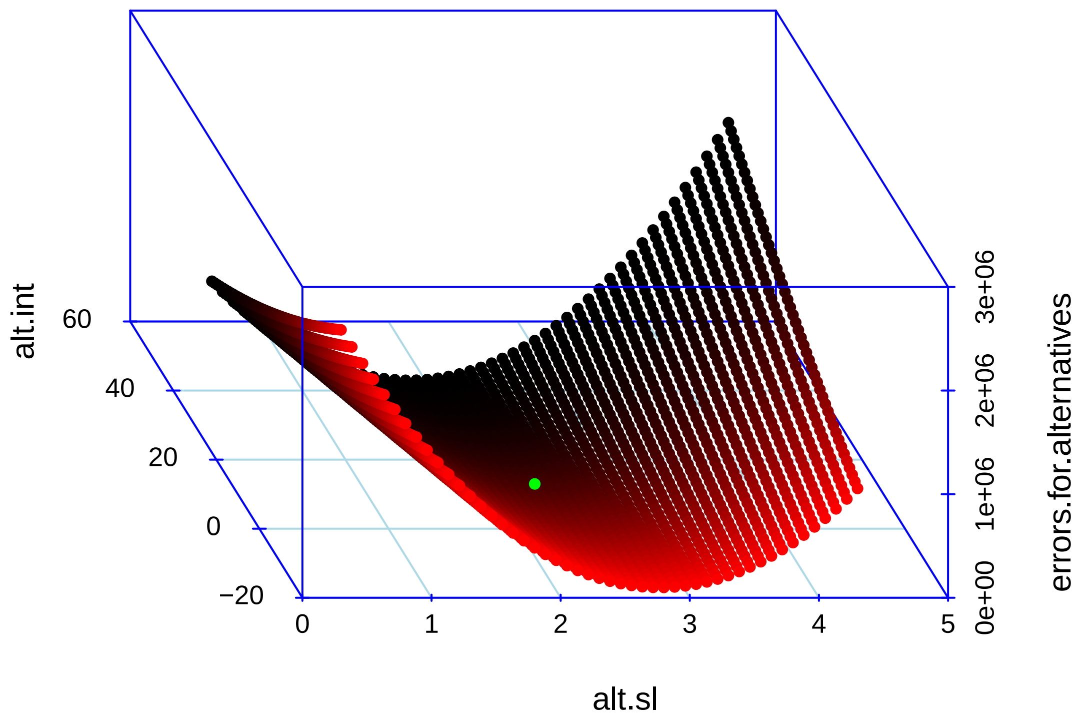


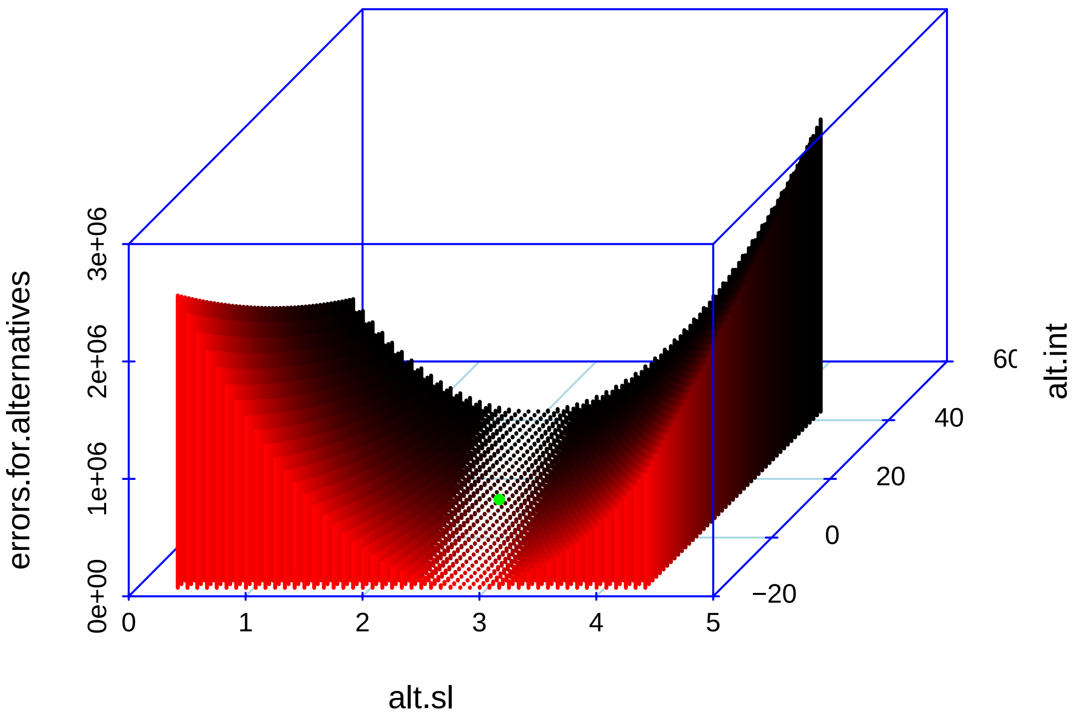
7.3 Alternative intercepts and slopes

And we can also look at alternative slopes and intercepts simultaneously:

```
> (alternative.slopes <- seq(reg1.slope - 2, reg1.slope + 2, length.out = 49))

 [1] 0.3473 0.4306 0.5139 0.5973 0.6806 0.7639 0.8473 0.9306 1.0139 1.0973
[11] 1.1806 1.2639 1.3473 1.4306 1.5139 1.5973 1.6806 1.7639 1.8473 1.9306
[21] 2.0139 2.0973 2.1806 2.2639 2.3473 2.4306 2.5139 2.5973 2.6806 2.7639
[31] 2.8473 2.9306 3.0139 3.0973 3.1806 3.2639 3.3473 3.4306 3.5139 3.5973
[41] 3.6806 3.7639 3.8473 3.9306 4.0139 4.0973 4.1806 4.2639 4.3473

> (alternative.intercepts <- seq(reg1.intercept - 30, reg1.intercept +
+     30, length.out = 49))

 [1] -17.1072 -15.8572 -14.6072 -13.3572 -12.1072 -10.8572  -9.6072
 [8]  -8.3572  -7.1072  -5.8572  -4.6072  -3.3572  -2.1072  -0.8572
[15]   0.3928   1.6428   2.8928   4.1428   5.3928   6.6428   7.8928
```

```
[22]   9.1428  10.3928  11.6428  12.8928  14.1428  15.3928  16.6428
[29]  17.8928  19.1428  20.3928  21.6428  22.8928  24.1428  25.3928
[36]  26.6428  27.8928  29.1428  30.3928  31.6428  32.8928  34.1428
[43]  35.3928  36.6428  37.8928  39.1428  40.3928  41.6428  42.8928

> alt.sl <- rep(alternative.slopes, each = 49)
> alt.int <- rep(alternative.intercepts, 49)
> errors.for.alternatives <- numeric(length = 49 * 49)
> for (i in 1:(49 * 49)) {
+     errors.for.alternatives[i] <- sum((y - (alt.int[i] + (alt.sl[i] *
+         x1)))^2)
+ }
> library("scatterplot3d")
> par(mfrow = c(2, 1))
> s3d <- scatterplot3d(alt.sl, alt.int, errors.for.alternatives, highlight.3d = TRUE,
+     col.axis = "blue", col.grid = "lightblue", pch = 20, type = "p",
+     angle = 120, lab = c(3, 3, 3), mar = c(3, 3, 3, 3))
> s3d$points3d(reg1.slope, reg1.intercept, sum((summary(reg1)$residuals)^2),
+     col = "green", type = "h", pch = 20)
> s3d <- scatterplot3d(alt.sl, alt.int, errors.for.alternatives, highlight.3d = TRUE,
+     col.axis = "blue", col.grid = "lightblue", pch = " ", lwd = 2,
+     type = "h", lty.hplot = 1, angle = 45, lab = c(3, 3, 3), mar = c(3,
+         3, 3, 3))
> s3d$points3d(reg1.slope, reg1.intercept, sum((summary(reg1)$residuals)^2),
+     col = "green", type = "p", pch = 20)
```


alt.int
60
40
20
0
−20
0
1
2
3
4
5
alt.sl
0e+00
1e+06
2e+06
3e+06
errors.for.alternatives


errors.for.alternatives
0e+00
1e+06
2e+06
3e+06
0
1
2
3
4
5
alt.sl
−20
0
20
40
60
alt.int

References

Abelson, R.P. (1995). *Statistics as Principled Argument*. L. Erlbaum Associates.

Baayen, R. Harald (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*. Cambridge University Press.

Braun, J. and D.J. Murdoch (2007). *A First Course in Statistical Programming with R*. Cambridge University Press.

De Veaux, R.D. et al. (2005). *Stats: Data and Models*. Pearson Education, Limited.

Diez, D. et al. (2013). *OpenIntro Statistics: Second Edition*. CreateSpace Independent Publishing Platform. URL: `http://www.openintro.org/stat/textbook.php`.

Faraway, J.J. (2004). *Linear Models With R*. Chapman & Hall Texts in Statistical Science Series. Chapman & Hall/CRC.

Gelman, A. and J. Hill (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press.

Gonick, L. and W. Smith (1993). *Cartoon Guide to Statistics*. HarperCollins.

Gries, S.T. (2009). *Quantitative Corpus Linguistics with R: A Practical Introduction*. Taylor & Francis.

— (2013). *Statistics for Linguistics with R: A Practical Introduction, 2nd Edition*. Mouton De Gruyter.

Johnson, K. (2008). *Quantitative methods in linguistics*. Blackwell Pub.

Kruschke, John K. (2011). *Doing Bayesian Data Analysis: A Tutorial with R and BUGS*. Academic Press/Elsevier.

Miles, J. and M. Shevlin (2001). *Applying Regression and Correlation: A Guide for Students and Researchers*. SAGE Publications.

Wright, D.B. and K. London (2009). *Modern regression techniques using R: A practical guide for students and researchers*. SAGE.

Xie, Yihui (2013). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC.