Quantitative Methods in Linguistics – Lecture 7

Adrian Brasoveanu[*]

April 11, 2014

Contents

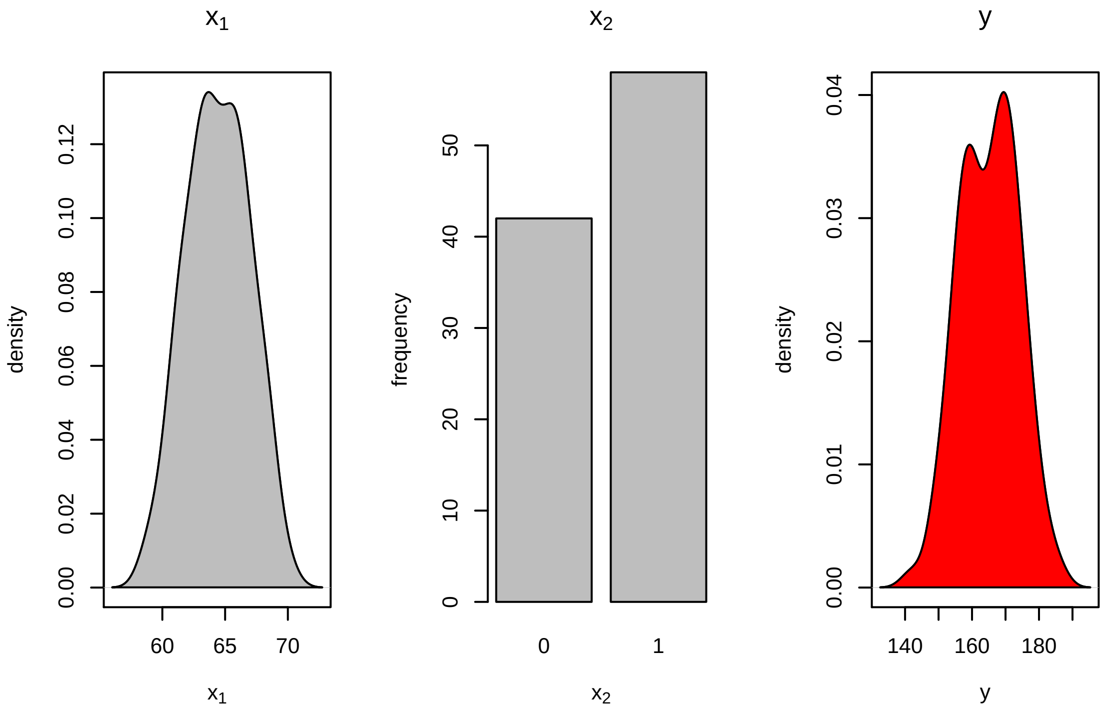
1 Recap and related issues

1.1 Generating the "dataset"

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

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   58.7    62.8    64.5    64.4    66.2    70.1
```

[*] 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).

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

[1] 58

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

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    142     158     166     165     171     186

> 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))
```

1.2 The two regressions we looked at

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

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

Residuals:
    Min      1Q  Median      3Q     Max 
-23.190  -6.691   0.632   6.341  20.839 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   165.12       0.87     190   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.7 on 99 degrees of freedom

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


Call:
lm(formula = y ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max 
-18.630  -5.645  -0.307   5.202  17.536 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   160.56       1.21  133.07  < 2e-16 ***
x2              7.86       1.58    4.96  2.9e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.82 on 98 degrees of freedom
Multiple R-squared:  0.201,Adjusted R-squared:  0.193 
F-statistic: 24.6 on 1 and 98 DF,  p-value: 2.92e-06

> 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 7498                              
2     98 5992  1      1506 24.6 2.9e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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


Call:
```

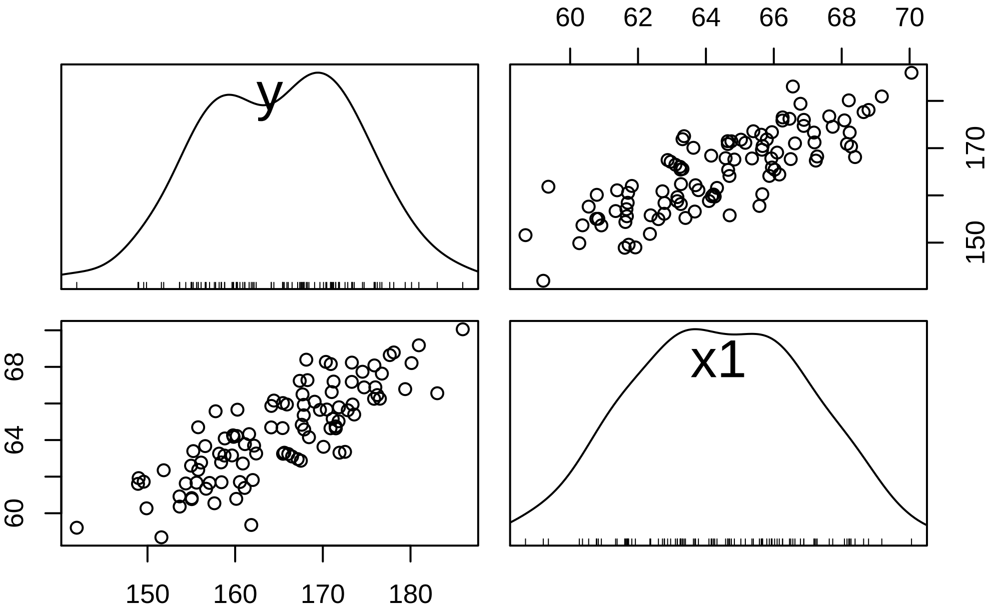
```
lm(formula = y ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max 
-10.684  -3.621   0.208   3.920  11.788 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -18.931     13.026   -1.45     0.15    
x1             2.857      0.202   14.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.02 on 98 degrees of freedom
Multiple R-squared:  0.671,Adjusted R-squared:  0.668 
F-statistic:  200 on 1 and 98 DF,  p-value: <2e-16

> 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 7498                            
2     98 2467  1      5032 200 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.3 Scatter plot matrices (SPMs)

Scatter plot matrices (SPM for short):

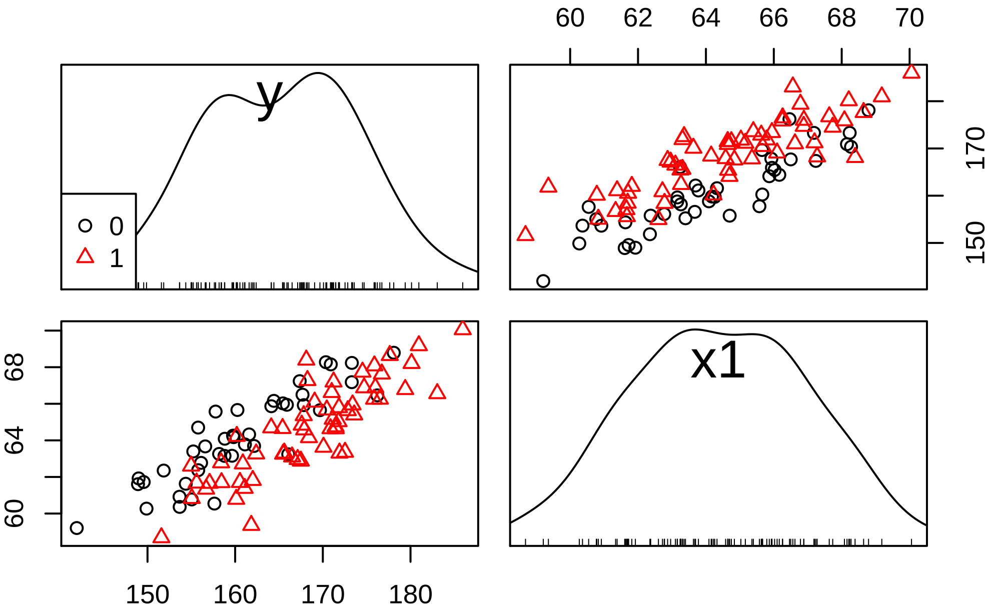
- smoothed histograms of the variables are displayed on the (upper-left to lower-right) diagonal
- the other panels display plots of all pairs of variables

```
> library("car")
> spm(~y + x1, smooth = FALSE, reg.line = FALSE)
```



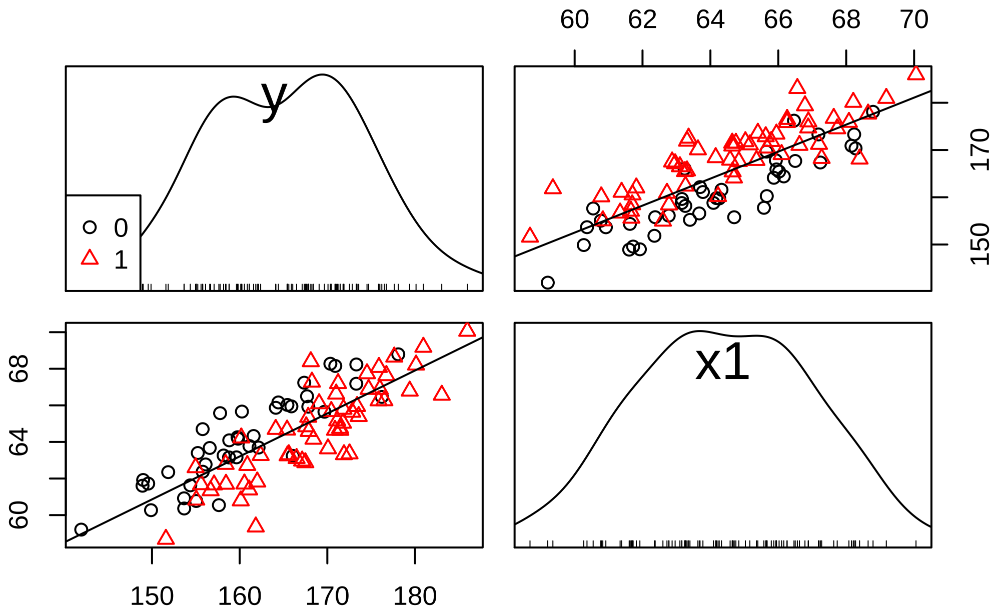
We can group the observations based on a factor:

```
> spm(~y + x1, smooth = FALSE, groups = as.factor(x2), reg.line = FALSE)
```



And we can add regression lines:

```
> spm(~y + x1, smooth = FALSE, groups = as.factor(x2))
```



2 R-squared (R^2)

```
> summary(reg1)


Call:
lm(formula = y ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max
-10.684  -3.621   0.208   3.920  11.788

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -18.931     13.026   -1.45     0.15
x1             2.857      0.202   14.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.02 on 98 degrees of freedom
Multiple R-squared:  0.671,Adjusted R-squared:  0.668
F-statistic:  200 on 1 and 98 DF,  p-value: <2e-16
```

R^2 – the *proportion* of the total squared error that is accounted for by the model, i.e.:

- error between / total error

```
> squared.error.reg1 <- sum((summary(reg1)$residuals)^2)  # squared error within (within the model / no
> squared.error.1MEAN <- sum((y - mean(y))^2)  # total squared error
> squared.error.difference <- squared.error.1MEAN - squared.error.reg1  # squared error between, i.e.,
> r_squared <- squared.error.difference/squared.error.1MEAN
> r_squared

[1] 0.6711

> summary(reg1)$r.squared

[1] 0.6711
```

We now understand (almost) everything in the `lm` output.[1]

3 Correlation

Correlation: R, i.e., the square root of R^2.

```
> cor(y, x1)

[1] 0.8192

> sqrt(summary(reg1)$r.squared)

[1] 0.8192

> sqrt(squared.error.difference/squared.error.1MEAN)

[1] 0.8192
```

Thus, correlation measures how much of the variation in y is accounted for by the variation in x_1:

- to put it differently, it measures how strongly y and x_1 are correlated / associated
- graphically, correlation measures the degree of scatter around the regression line

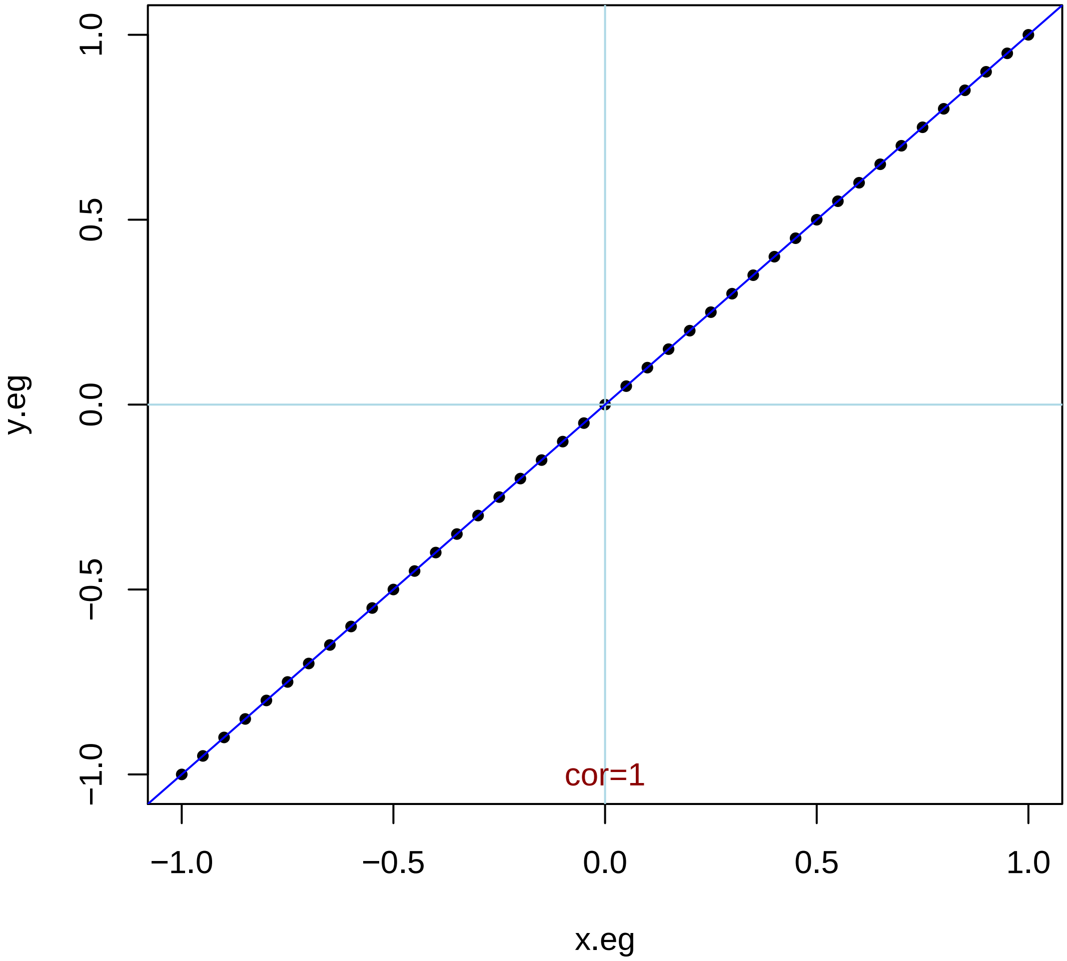
The correlation is always between -1 and 1. It's actually the cosine of the angle between the predictor and the response vectors in n dimensional space, where n is the number of observations, i.e., coordinates.

- the closer the correlation is to 0, the more scatter / the less association
- positive correlation: upward slope
- negative number: downward slope

```
> (x.eg <- seq(-1, 1, by = 0.05))

 [1] -1.00 -0.95 -0.90 -0.85 -0.80 -0.75 -0.70 -0.65 -0.60 -0.55 -0.50
[12] -0.45 -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10 -0.05  0.00  0.05
[23]  0.10  0.15  0.20  0.25  0.30  0.35  0.40  0.45  0.50  0.55  0.60
[34]  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00

> (y.eg <- seq(-1, 1, by = 0.05))
```

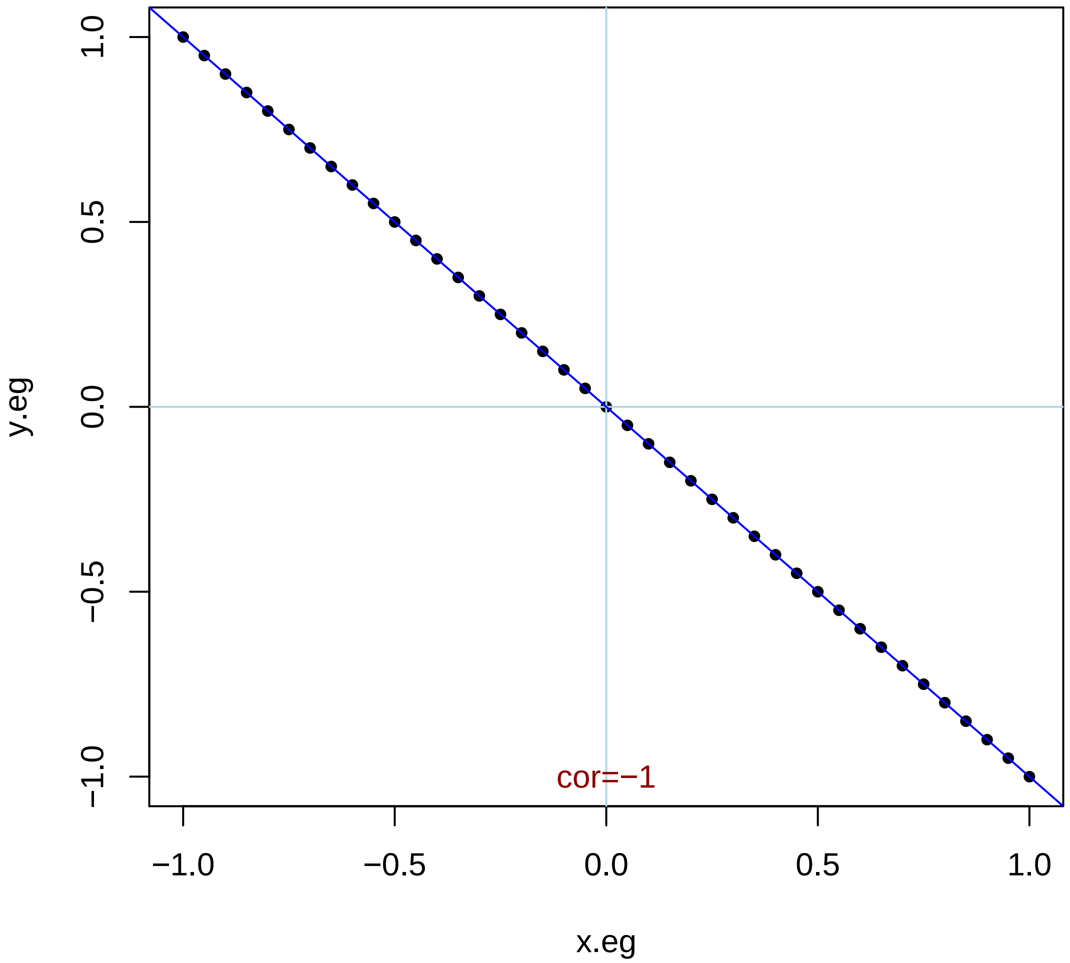
[1] The adjusted R^2 adjusts for the number of predictors in the model; it increases when a new predictor is added only if the new predictor improves the model more than would be expected by chance. Its interpretation is not as clear as the interpretation of R^2, so we will not discuss it further.

```
 [1] -1.00 -0.95 -0.90 -0.85 -0.80 -0.75 -0.70 -0.65 -0.60 -0.55 -0.50
[12] -0.45 -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10 -0.05  0.00  0.05
[23]  0.10  0.15  0.20  0.25  0.30  0.35  0.40  0.45  0.50  0.55  0.60
[34]  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00

> plot(x.eg, y.eg, pch = 20, xlim = range(-1, 1), ylim = range(-1, 1))
> abline(v = 0, col = "lightblue")
> abline(h = 0, col = "lightblue")
> abline(lm(y.eg ~ x.eg), col = "blue", lwd = 1)
> text(0, -1, paste("cor=", round(cor(y.eg, x.eg), 2), sep = ""), col = "darkred",
+      cex = 1)
```

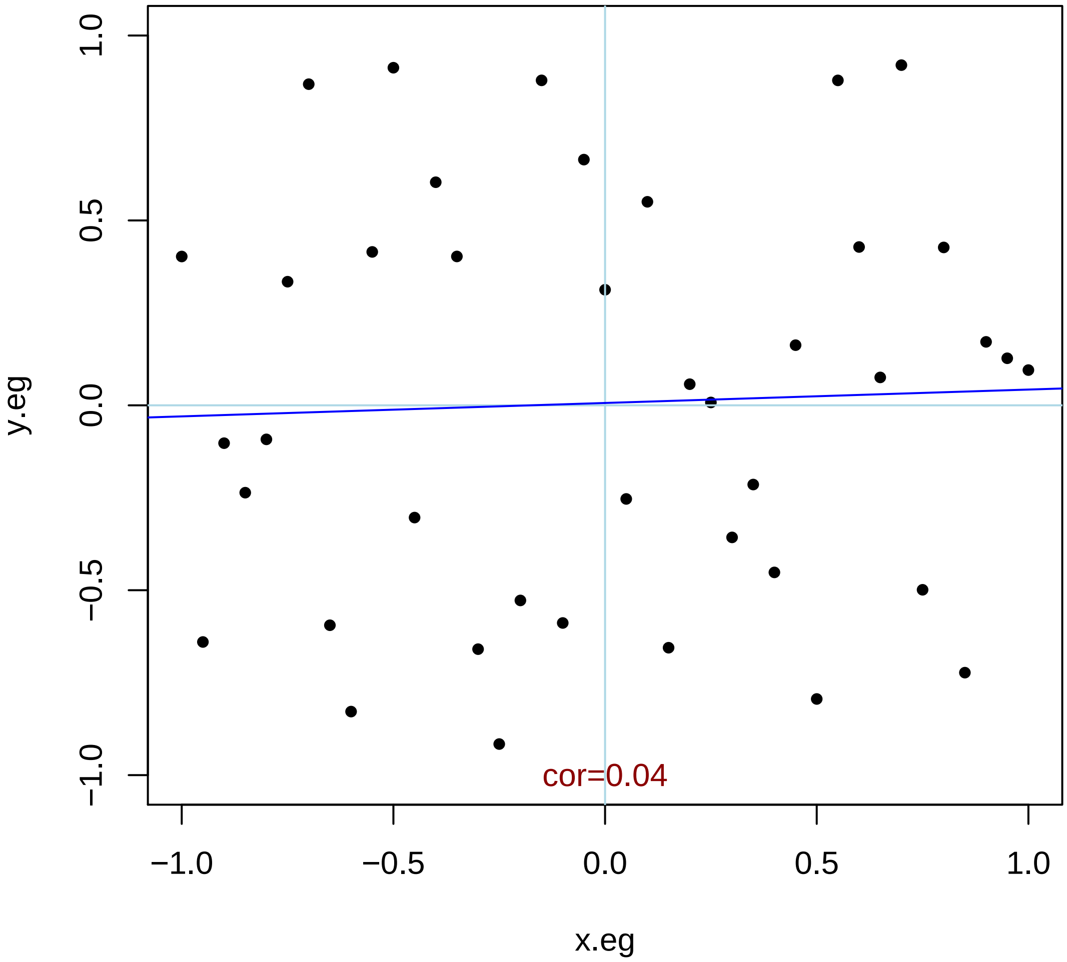


```
> (x.eg <- seq(-1, 1, by = 0.05))

 [1] -1.00 -0.95 -0.90 -0.85 -0.80 -0.75 -0.70 -0.65 -0.60 -0.55 -0.50
[12] -0.45 -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10 -0.05  0.00  0.05
[23]  0.10  0.15  0.20  0.25  0.30  0.35  0.40  0.45  0.50  0.55  0.60
```

```
[34]  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00

> (y.eg <- -seq(-1, 1, by = 0.05))

 [1]  1.00  0.95  0.90  0.85  0.80  0.75  0.70  0.65  0.60  0.55  0.50
[12]  0.45  0.40  0.35  0.30  0.25  0.20  0.15  0.10  0.05  0.00 -0.05
[23] -0.10 -0.15 -0.20 -0.25 -0.30 -0.35 -0.40 -0.45 -0.50 -0.55 -0.60
[34] -0.65 -0.70 -0.75 -0.80 -0.85 -0.90 -0.95 -1.00

> plot(x.eg, y.eg, pch = 20, xlim = range(-1, 1), ylim = range(-1, 1))
> abline(v = 0, col = "lightblue")
> abline(h = 0, col = "lightblue")
> abline(lm(y.eg ~ x.eg), col = "blue", lwd = 1)
> text(0, -1, paste("cor=", round(cor(y.eg, x.eg), 2), sep = ""), col = "darkred",
+     cex = 1)
```

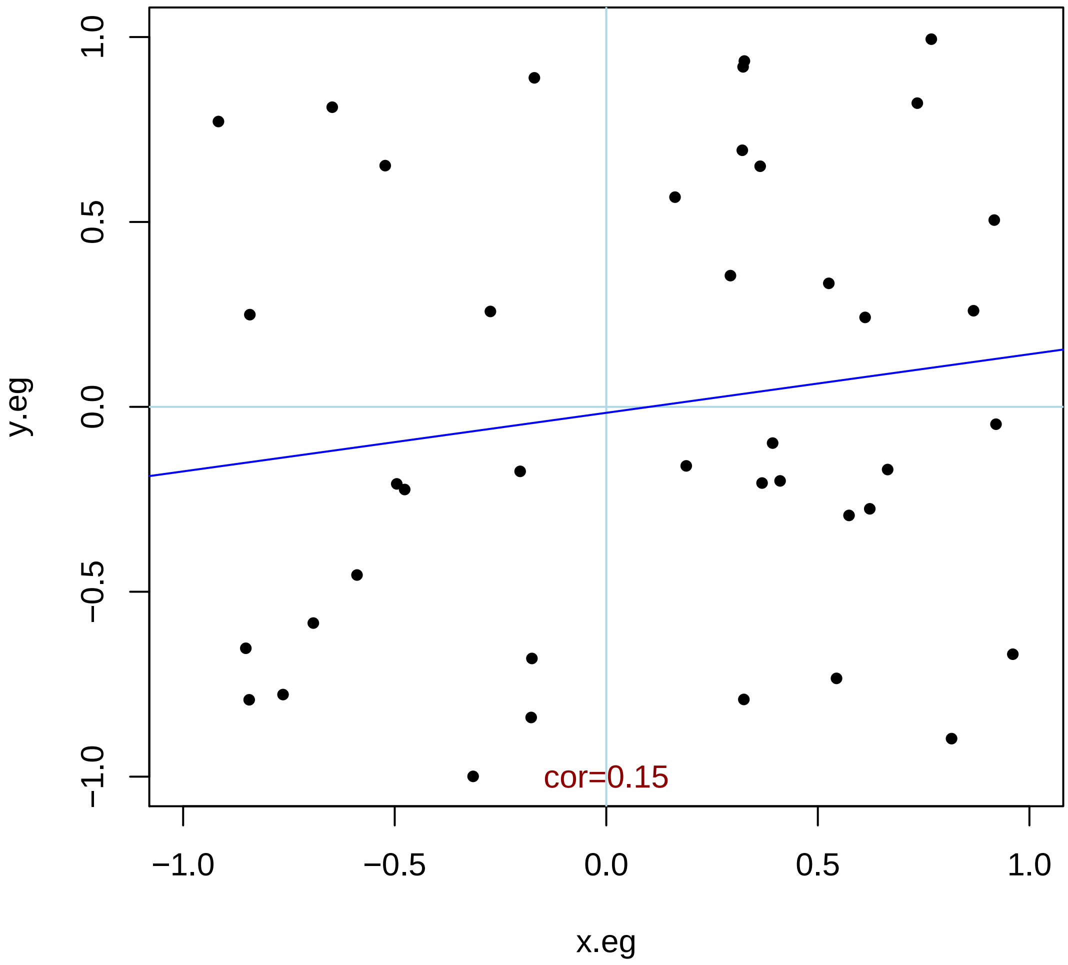


```
> (x.eg <- seq(-1, 1, by = 0.05))

 [1] -1.00 -0.95 -0.90 -0.85 -0.80 -0.75 -0.70 -0.65 -0.60 -0.55 -0.50
[12] -0.45 -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10 -0.05  0.00  0.05
[23]  0.10  0.15  0.20  0.25  0.30  0.35  0.40  0.45  0.50  0.55  0.60
[34]  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00

> (y.eg <- runif(41, -1, 1))

 [1]  0.402498 -0.639957 -0.102366 -0.236195 -0.091991  0.334301  0.868485
 [8] -0.594678 -0.828178  0.414890  0.913012 -0.303560  0.603309  0.402626
[15] -0.659322 -0.915922 -0.527625  0.878808 -0.588549  0.664495  0.312577
[22] -0.253097  0.550351 -0.655465  0.057213  0.007786 -0.357145 -0.214208
[29] -0.451769  0.162721 -0.794050  0.878694  0.428176  0.075490  0.919979
[36] -0.498813  0.426961 -0.722915  0.171659  0.127313  0.095328

> plot(x.eg, y.eg, pch = 20, xlim = range(-1, 1), ylim = range(-1, 1))
> abline(v = 0, col = "lightblue")
> abline(h = 0, col = "lightblue")
> abline(lm(y.eg ~ x.eg), col = "blue", lwd = 1)
> text(0, -1, paste("cor=", round(cor(y.eg, x.eg), 2), sep = ""), col = "darkred",
+     cex = 1)
```



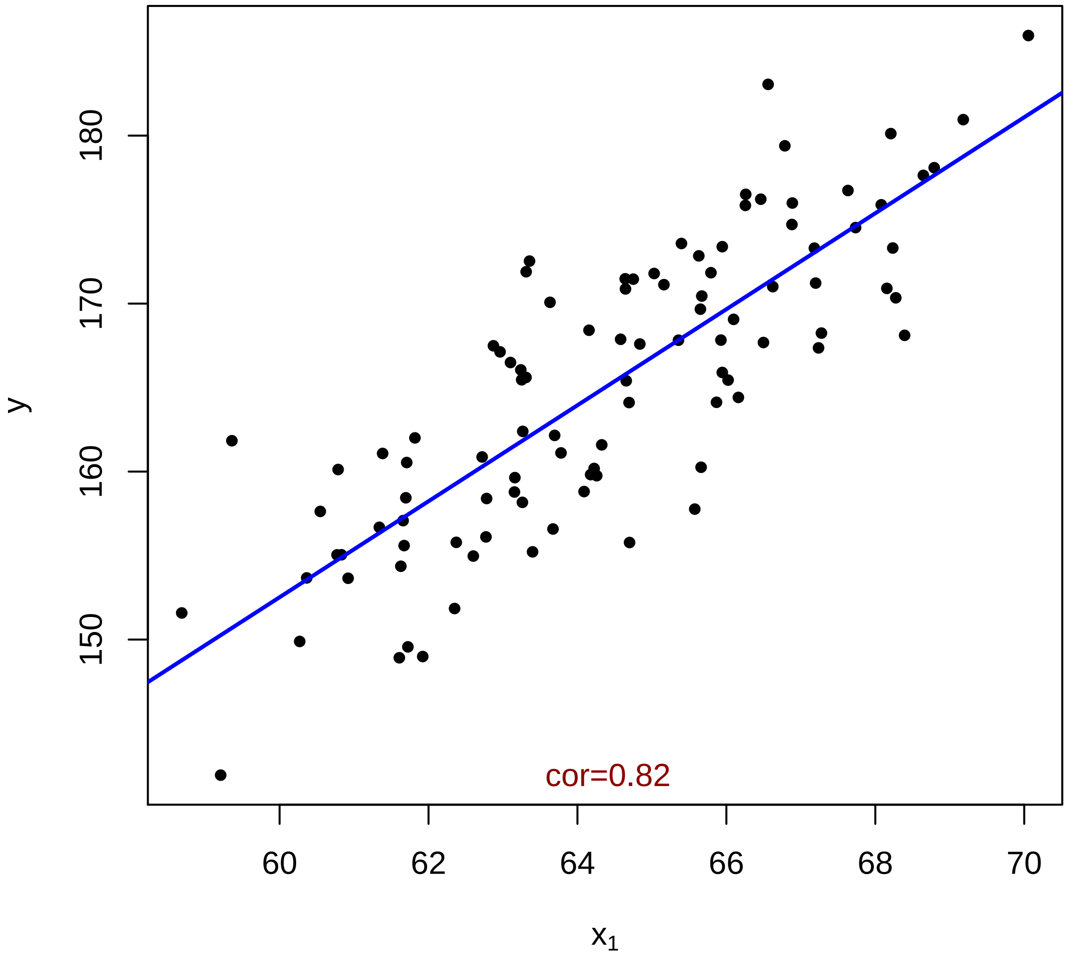
```
> (x.eg <- runif(41, -1, 1))

 [1]  0.1624  0.4108  0.7350  0.2934 -0.5224  0.5441  0.6649 -0.1700
 [9]  0.9210 -0.8519  0.3234  0.1890 -0.6924 -0.4952 -0.1775  0.3682
[17]  0.7681  0.3251  0.5259  0.3931  0.8160 -0.4764 -0.7639  0.8680
[25] -0.9166  0.6117  0.3637  0.5737  0.9608 -0.2739 -0.8424 -0.6477
[33] -0.2036 -0.1758 -0.3147  0.6227  0.9169  0.3264  0.3214 -0.5891
[41] -0.8439

> (y.eg <- runif(41, -1, 1))

 [1]  0.56713 -0.20016  0.82134  0.35494  0.65233 -0.73421 -0.16944
 [8]  0.88978 -0.04686 -0.65277  0.91948 -0.15956 -0.58464 -0.20825
[15] -0.83990 -0.20598  0.99433 -0.79118  0.33401 -0.09790 -0.89708
[22] -0.22350 -0.77801  0.25998  0.77164  0.24193  0.65071 -0.29348
[29] -0.66887  0.25802  0.24940  0.81036 -0.17437 -0.68031 -0.99922
[36] -0.27573  0.50512  0.93516  0.69387 -0.45472 -0.79212
```

```
> plot(x.eg, y.eg, pch = 20, xlim = range(-1, 1), ylim = range(-1, 1))
> abline(v = 0, col = "lightblue")
> abline(h = 0, col = "lightblue")
> abline(lm(y.eg ~ x.eg), col = "blue", lwd = 1)
> text(0, -1, paste("cor=", round(cor(y.eg, x.eg), 2), sep = ""), col = "darkred",
+      cex = 1)
```



The correlation for the predictor and response in the reg1 model:

```
> plot(x1, y, pch = 20, xlab = expression(x[1]))
> abline(reg1, col = "blue", lwd = 2)
> text(mean(x1), min(y), paste("cor=", round(cor(y, x1), 2), sep = ""),
+      col = "darkred", cex = 1)
```



Correlation is the 3rd type of descriptive / summary statistics. That is, we are usually interested and report 3 types of descriptive / summary statistics:

- measures of center: mean (also median, mode)
- measures of dispersion: standard deviation (also IQR, entropy)
- measures of association: correlation

4 An alternative way of calculating correlation

This type of calculation makes it much clearer that correlation is the cosine of an angle.

We first start with the covariance, which is the mean of the dot product of the vector of centered y and x_1 vectors (centered: with a mean of 0; dot product: the sum of the products of the coordinate-wise values):

```
> sum((y - mean(y)) * (x1 - mean(x1)))/(length(y) - 1)

[1] 17.79
```

```
> ((y - mean(y)) %*% (x1 - mean(x1)))/(length(y) - 1)

      [,1]
[1,] 17.79

> cov(y, x1)

[1] 17.79
```

We divide by $n - 1$ because we spend one dof on the mean of y.
Note that the covariance of a variable with itself is the variance of that variable:

```
> ((y - mean(y)) %*% (y - mean(y)))/(length(y) - 1)

      [,1]
[1,] 75.74

> cov(y, y)

[1] 75.74

> var(y)

[1] 75.74
```

Correlation is just covariance divided / "normalized" by the standard deviations of, i.e., the variation in, the correlated variables:

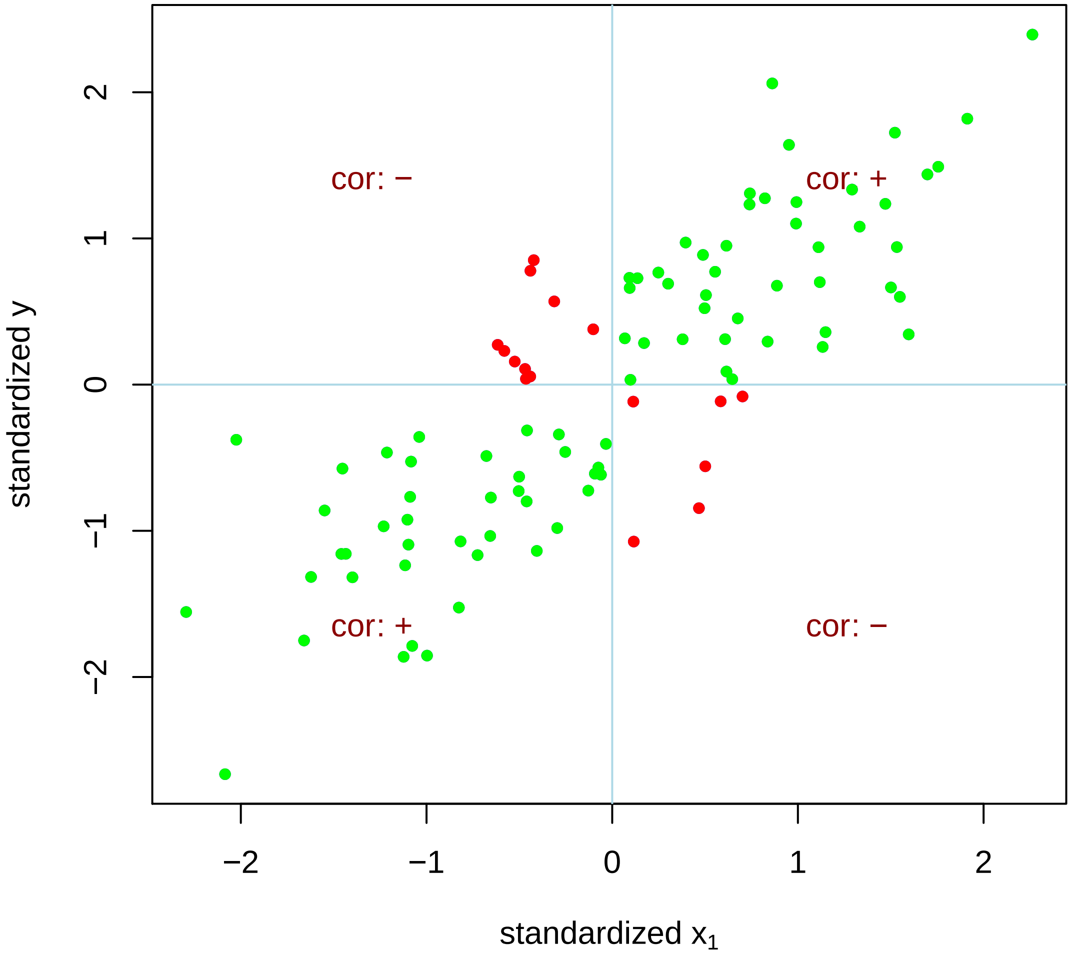
```
> cov(y, x1)/(sd(y) * sd(x1))

[1] 0.8192

> cor(y, x1)

[1] 0.8192
```

Take a closer look at this formula for correlation:

- $\frac{\textbf{cov}(y,x_1)}{\textbf{sd}(y)\textbf{sd}(x_1)}$ is the averaged (divided by $n - 1$) dot product of the standardized y and x_1:
$\frac{\textbf{cov}(y,x_1)}{\textbf{sd}(y)\textbf{sd}(x_1)} = \frac{1}{n-1}\frac{(y-\bar{y})\cdot(x_1-\bar{x}_1)}{\textbf{sd}(y)\textbf{sd}(x_1)} = \frac{1}{n-1}\frac{y-\bar{y}}{\textbf{sd}(y)}\frac{x_1-\bar{x}_1}{\textbf{sd}(x_1)}$
- that is, correlation is just a numerical summary of the standardized plot of y and x_1

```
> standardized.x1 <- (x1 - mean(x1))/sd(x1)
> standardized.y <- (y - mean(y))/sd(y)
> plot(standardized.x1, standardized.y, pch = 20, col = "blue", xlab = expression(paste("standardized "
+     x[1])), ylab = "standardized y")
> abline(v = 0, col = "lightblue")
> abline(h = 0, col = "lightblue")
> text(min(standardized.x1) + 1, min(standardized.y) + 1, "cor: +",
+     cex = 1, col = "darkred")
> text(max(standardized.x1) - 1, max(standardized.y) - 1, "cor: +",
+     cex = 1, col = "darkred")
> text(min(standardized.x1) + 1, max(standardized.y) - 1, "cor: -",
+     cex = 1, col = "darkred")
```

```
> text(max(standardized.x1) - 1, min(standardized.y) + 1, "cor: -",
+      cex = 1, col = "darkred")
> for (i in 1:length(standardized.x1)) {
+     if (standardized.x1[i] * standardized.y[i] > 0) {
+         points(standardized.x1[i], standardized.y[i], pch = 20, col = "green")
+     }
+     if (standardized.x1[i] * standardized.y[i] < 0) {
+         points(standardized.x1[i], standardized.y[i], pch = 20, col = "red")
+     }
+ }
```



Importantly, the correlation is just the slope of the regression line for the standardized variables:

```
> lm(standardized.y ~ standardized.x1)$coef[2]

standardized.x1
         0.8192

> cor(y, x1)
```

```
[1] 0.8192
```

The slope of the regression line for the original variables can be obtained from the "standardized" slope (i.e., the correlation) and the standard deviations of the two variables:

```
> x1_slope <- as.numeric(lm(y ~ x1)$coef[2])
> x1_slope

[1] 2.857

> standardized_x1_slope <- as.numeric(lm(standardized.y ~ standardized.x1)$coef[2])
> standardized_x1_slope

[1] 0.8192

> standardized_x1_slope * sd(y)/sd(x1)

[1] 2.857
```

Size of correlations: not as important as it is for natural sciences. Given the nature of the empirical domain, namely human behavior, it is hard to identify any single variable / group of variables that accounts for a significant part of the variation in the response variable.

It is more important that the correlation is statistically significant (but: large samples make even small correlations statistically significant although they are unlikely to be practically significant).

Guidelines for interpreting correlation values (from Cohen 1988):

(1) If the absolute value of the correlation is:

- approx. 0.1: small correlation
- approx. 0.3: medium correlation
- 0.5 or more: large correlation

5 Inference for simple linear regression: from sample to population

'Simple' linear regression means: with only one predictor (in addition to the intercept). As opposed to 'multiple' linear regression: with multiple predictors (again, in addition to the intercept).

The `lm` (regression) coefficients, which determine a regression line, provide a summary of how the variables y and x_1 are associated *in our sample*. What we want to know is how y and x_1 are associated *in the population*, i.e., we want to find the population-level generalization, not merely the generalization / summary for our sample.

In particular, we want to understand how we obtain the SEs for the coefficients (also the t-values and p-values, but that follows easily from the SEs):

```
> summary(reg1)$coefficients

            Estimate Std. Error t value  Pr(>|t|)
(Intercept)  -18.931    13.0264  -1.453 1.493e-01
x1             2.857     0.2021  14.140 2.130e-25
```

If we had all the values for y and x_1 in the population, we would be able to find the exact, population-level intercept and slope for the regression line by using least squares. Let's represent that model as follows:

(2) $\mu_y = \beta_0 + \beta_1 x_1$

Note the Greek variables: they explicitly indicate we deal with population-level (hidden / unobserved) variables.

In particular, we use μ_y, which stands for "the mean of y", to indicate explicitly that for each particular value of x_1, the predicted value for y is the **mean** of y values for all the cases / observations that have that particular value for x_1.

In other words, our population-level regression model for the actual values, not only for the mean values, is in fact:

(3) $y = \mu_y + \epsilon$, , where ϵ stands for 'error' / noise

(4) Equivalently: $y = \beta_0 + \beta_1 x_1 + \epsilon$

Recall that this is exactly the way we simulated our data (except for the fact that we also had the x_2 predictor in the mix).

The error term ϵ is the spread (the variance, or its square root, i.e., the standard deviation) of the actual y values around mu_y for each particular value for x_1.

The error in the regression model for the entire *population* comes from:

- the fact that other variables besides x_1 influence the value for y (variables like x_2 or others that we have not taken into account)
- the random / probabilistic nature of the phenomenon itself, for example, weight is partly determined by your genetic makeup, and that genetic makeup is a *probabilistic* / non-deterministic function of the genetic makeup of your biological parents

The error in the regression model for the *sample* comes from:

- the same 2 reasons as for the population, plus
- *sampling variation*

Our task: based on the *sample* regression coefficients and the *sample* regression error / residuals, do the best inference you can about the population coefficients and error given your CLT-based knowledge about sampling variation.

This boils down to estimating standard errors (SEs) and confidence intervals (CIs) for the *population* regression coefficients (plus t-values and p-values for hypothesis testing, but we saw that CIs are more informative for single coefficients).

Note that this focuses on accounting for sampling variation in our statistical / inductive inference. The variation induced by other predictors that we didn't take into account and about the probabilistic nature of the phenomenon itself (if applicable) is *not addressed at this level*.

Statistical inference at this level is conditional on / assumes that our model is correct is, i.e., we took into account all the population-level sources of variation and factored them into our model. Of course, this is not true: we do not have 'miraculous / oracular' knowledge about the inner workings of nature. To factor in our uncertainty about the model, we need to consider a class of models (or a range of classes of models) and do model comparison. This is the point where statistical inference and theory construction become more and more indistinguishable.

The main point here is to understand that our inference about coefficient estimates and their SEs assumes we have perfect knowledge / certainty about the statistical model we use, i.e.:

- we are completely certain that we use all and only the right predictors
- we use the right way to combine the predictors when we compute the mean of the responses
- we use the correct family of probability distributions to model the distribution of the actual responses around their mean

This is obviously not true. Thus, it is essential to keep in mind that we need to do model criticism / evaluation, i.e., model comparison, trying different classes of models (not only linear models as we do here), running follow-up experiments, trying to bring more of the theoretical structure into your statistical models etc.

We will only do a little bit of model comparison here, for example, we will compare the `reg1` and `reg2` models, which use only one predictor (height and gender, respectively) to model the response variable (weight), with the model that takes into account both predictors – which is actually the **true** model we used to simulate our data. But this process of model criticism and iterative model construction is as open-ended as the process of scientific-theory construction; they are ultimately the same thing.

But for now, let's assume (incorrectly) that our model `reg1` is the true model, i.e., the model that nature uses to generate weights given certain heights, and let's try to estimate the population coefficients given our sample and our CLT-based knowledge of sampling variation / sampling error.

5.1 The sampling error for the slope

(5) The sampling error for the slope depends on:

a. the spread around the regression line, i.e., the mean error measured "least-squares" style, i.e., (the square root of) the mean of the sum of squared residuals: the **smaller** the better

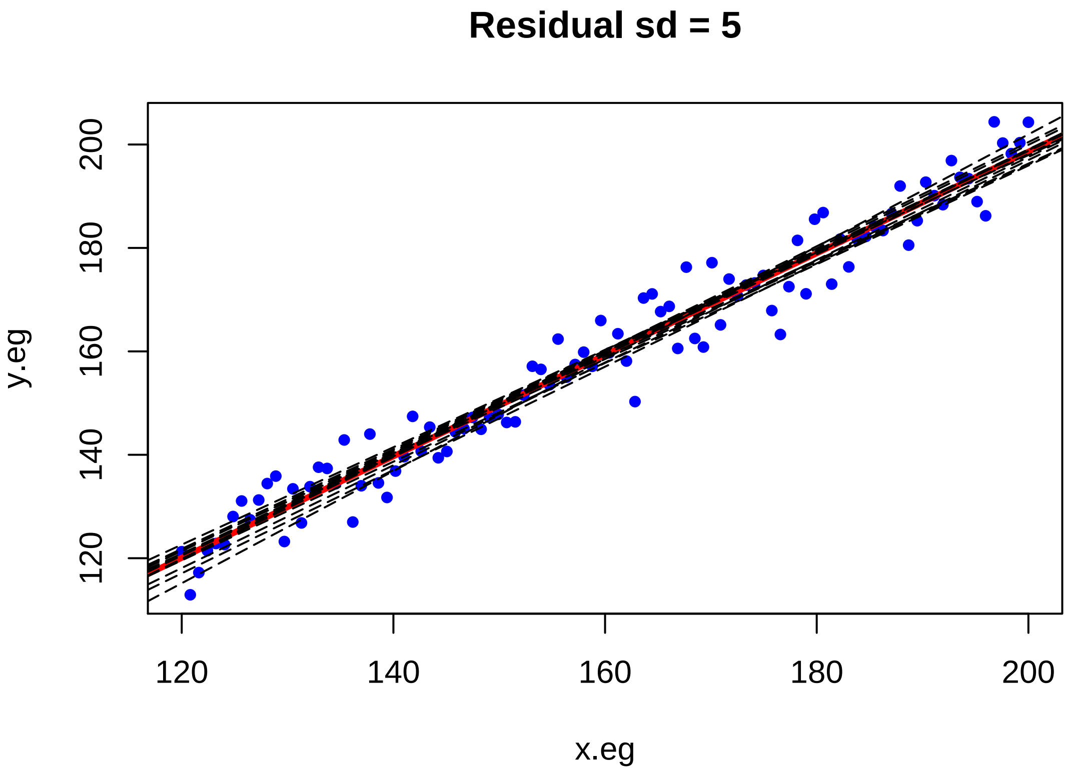
b. the spread of the predictor, i.e., the spread of the x_1 values: the **bigger** the better

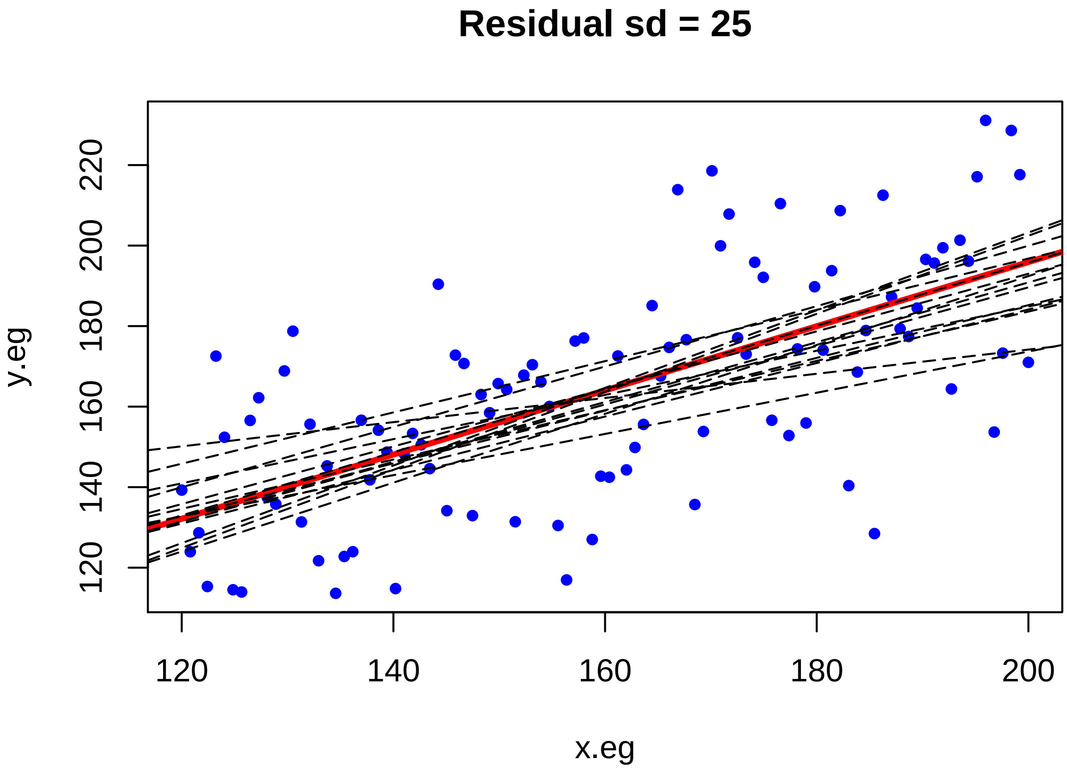
c. the size of the sample: the **bigger** the better

We used (5a) and (5c) to calculate the standard error of the mean. The new measure that is specific to regression is (5b). Let's examine them in turn.

5.1.1 The spread around the regression line, i.e., mean squared residuals: the smaller the better

```
> x.eg <- seq(120, 200, length = 100)
> par(mfrow = c(2, 1))
> y.eg <- x.eg + rnorm(100, 0, 5)
> plot(x.eg, y.eg, pch = 20, col = "blue", main = "Residual sd = 5")
> abline(lm(y.eg ~ x.eg), col = "red", lwd = 3)
> for (i in 1:15) {
+     index <- sample(1:100, 20)
+     abline(lm(y.eg[index] ~ x.eg[index]), lty = 5)
+ }
> y.eg <- x.eg + rnorm(100, 0, 25)
> plot(x.eg, y.eg, pch = 20, col = "blue", main = "Residual sd = 25")
> abline(lm(y.eg ~ x.eg), col = "red", lwd = 3)
> for (i in 1:15) {
+     index <- sample(1:100, 20)
+     abline(lm(y.eg[index] ~ x.eg[index]), lty = 5)
+ }
```

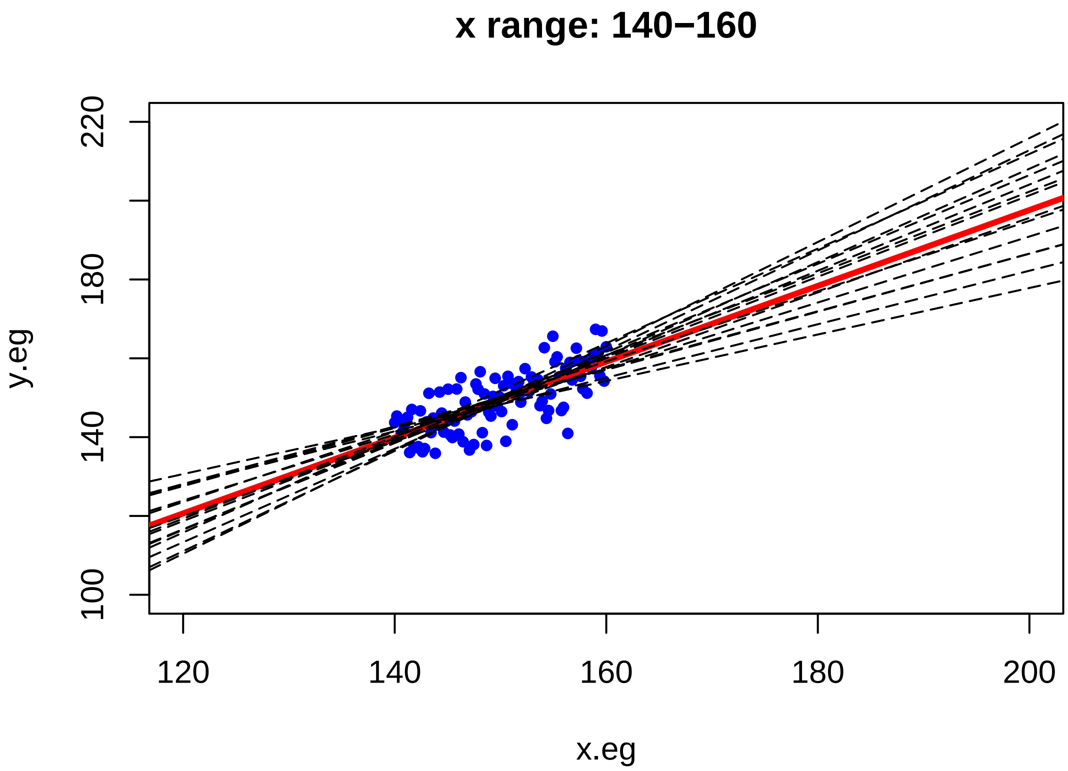

Residual sd = 5

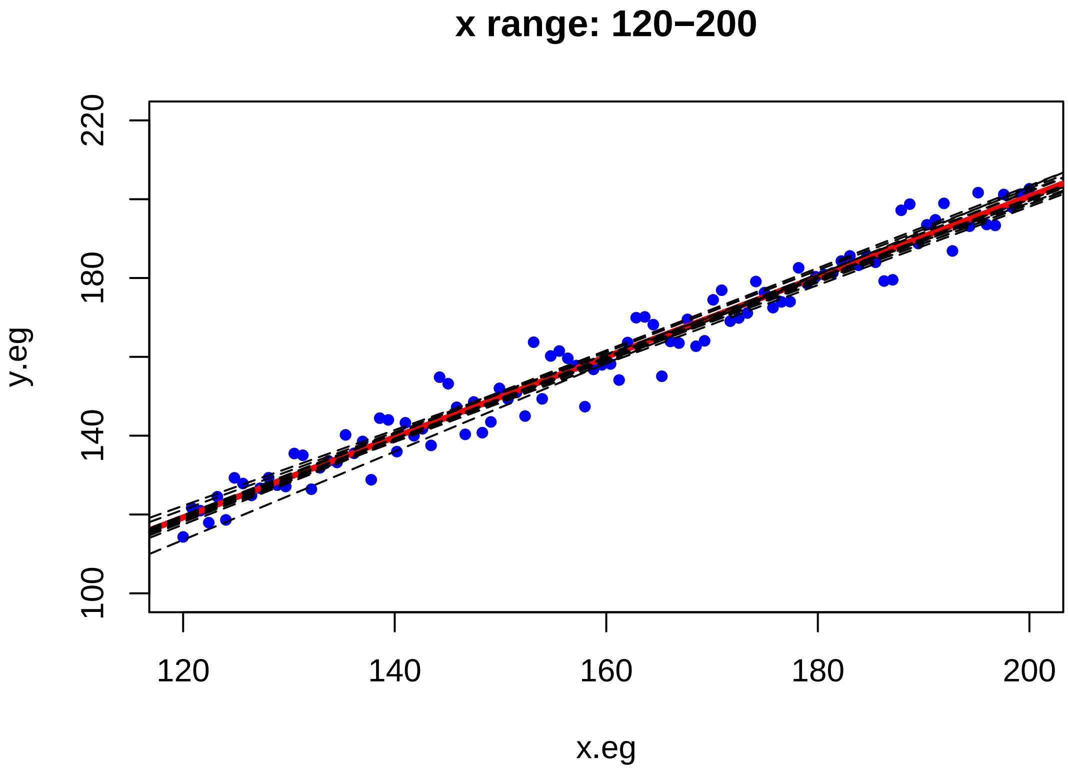

Residual sd = 25

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

5.1.2 The spread of the predictor values: the bigger the better

```
> x.eg <- seq(140, 160, length = 100)
> y.eg <- x.eg + rnorm(100, 0, 5)
> par(mfrow = c(2, 1))
> plot(x.eg, y.eg, pch = 20, col = "blue", xlim = range(120, 200), ylim = range(100,
+     220), main = "x range: 140-160")
> abline(lm(y.eg ~ x.eg), col = "red", lwd = 3)
> for (i in 1:15) {
+     index <- sample(1:100, 20)
+     abline(lm(y.eg[index] ~ x.eg[index]), lty = 5)
+ }
> x.eg <- seq(120, 200, length = 100)
> y.eg <- x.eg + rnorm(100, 0, 5)
> plot(x.eg, y.eg, pch = 20, col = "blue", xlim = range(120, 200), ylim = range(100,
+     220), main = "x range: 120-200")
> abline(lm(y.eg ~ x.eg), col = "red", lwd = 3)
> for (i in 1:15) {
+     index <- sample(1:100, 20)
+     abline(lm(y.eg[index] ~ x.eg[index]), lty = 5)
+ }
```

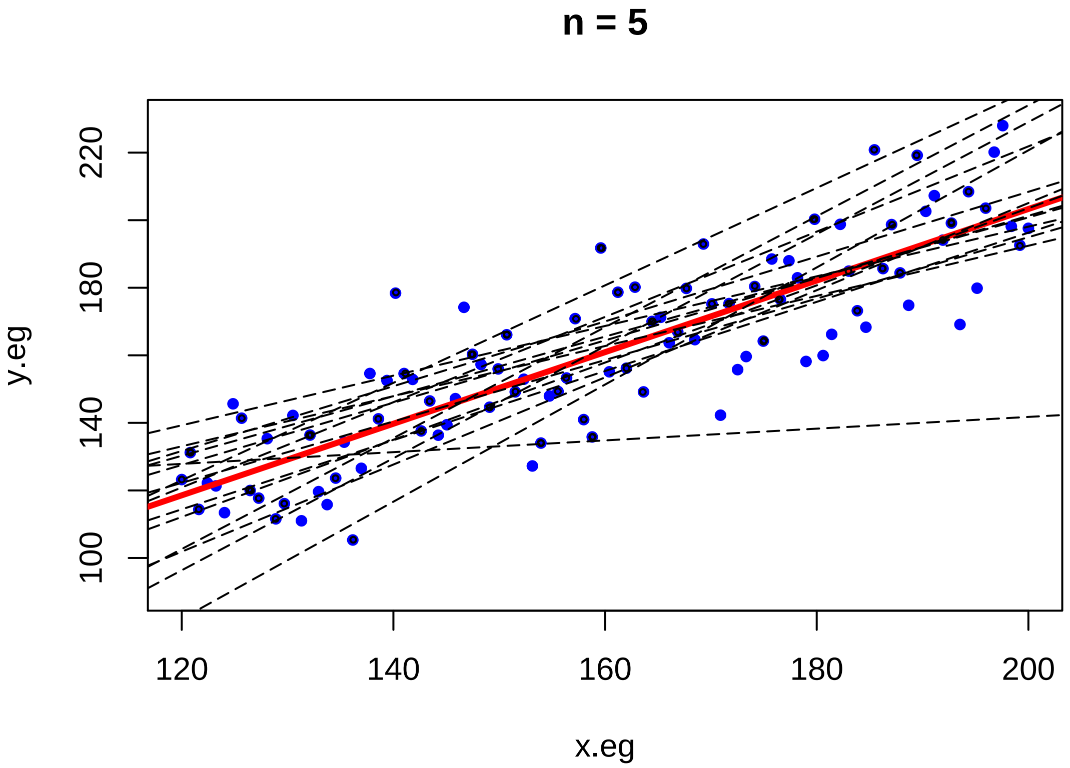

x range: 140−160
y.eg
100
140
180
220
x.eg
120
140
160
180
200

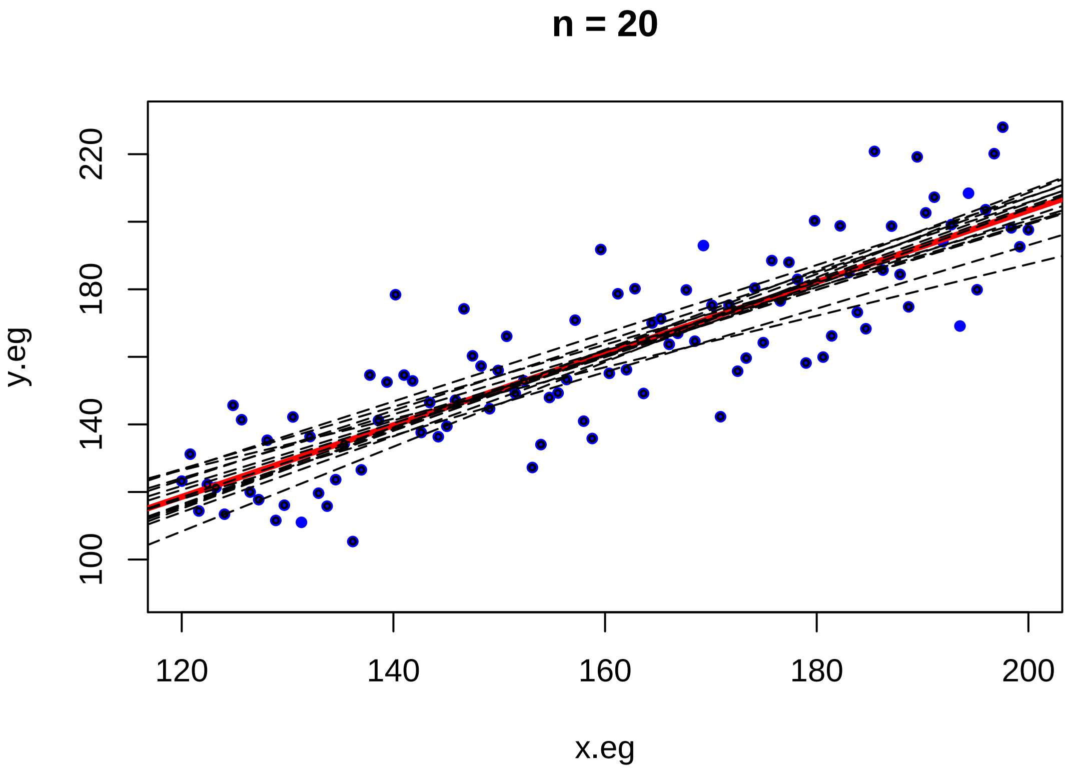

x range: 120−200
y.eg
100
140
180
220
x.eg
120
140
160
180
200

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

5.1.3 The size of the sample: the bigger the better

```
> x.eg <- seq(120, 200, length = 100)
> y.eg <- x.eg + rnorm(100, 0, 15)
> par(mfrow = c(2, 1))
> n <- 5
> plot(x.eg, y.eg, pch = 20, col = "blue", xlim = range(120, 200), ylim = range(90,
+     230), main = "n = 5")
> abline(lm(y.eg ~ x.eg), col = "red", lwd = 3)
> for (i in 1:15) {
+     index <- sample(1:100, n)
+     points(jitter(x.eg[index], 0.1), jitter(y.eg[index], 0.1), pch = 1,
+         cex = 0.4)
+     abline(lm(y.eg[index] ~ x.eg[index]), lty = 5)
+ }
> n <- 20
> plot(x.eg, y.eg, pch = 20, col = "blue", xlim = range(120, 200), ylim = range(90,
+     230), main = "n = 20")
> abline(lm(y.eg ~ x.eg), col = "red", lwd = 3)
> for (i in 1:15) {
+     index <- sample(1:100, n)
+     points(jitter(x.eg[index], 0.1), jitter(y.eg[index], 0.1), pch = 1,
+         cex = 0.4)
+     abline(lm(y.eg[index] ~ x.eg[index]), lty = 5)
+ }
```


n = 5
y.eg
100
140
180
220
x.eg
120
140
160
180
200


n = 20
y.eg
100
140
180
220
x.eg
120
140
160
180
200

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

5.1.4 Putting it all together

It turns out that these are the only 3 factors that affect the SE for the slope, which is:

(6) $$\text{slope SE} = \frac{\text{residual sd}}{\text{predictor sd} \cdot \sqrt{n-1}}$$

For our reg1 model, we have:

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

[1] 2467

> mean.squared.residuals <- sum.squared.residuals/(length(y) - 2)
> mean.squared.residuals

[1] 25.17

> residual_sd <- sqrt(mean.squared.residuals)
> residual_sd

[1] 5.017

> summary(reg1)$sigma

[1] 5.017

> (se.slope <- residual_sd/(sd(x1) * sqrt(length(y) - 1)))

[1] 0.2021

> summary(reg1)$coefficients[2, 1:2]

  Estimate Std. Error
    2.8575     0.2021
```

If the sample is big enough (100 observations definitely counts as big enough in our case), the 95% CI is obtained as we did for the mean, and also for the x_2 coefficient. We can use either the standard normal quantiles, or the t-distribution quantiles with the correct dof parameter. Since the dof parameter is high (98 dof.s), the two are very similar:

```
> (coef_x1 <- summary(reg1)$coefficients[2, 1])

[1] 2.857

> (SE_x1 <- summary(reg1)$coefficients[2, 2])

[1] 0.2021

> c(coef_x1 + qnorm(0.025) * SE_x1, coef_x1 + qnorm(0.975) * SE_x1)

[1] 2.461 3.254

> c(coef_x1 + qt(0.025, df = 98) * SE_x1, coef_x1 + qt(0.975, df = 98) *
+     SE_x1)

[1] 2.456 3.259
```

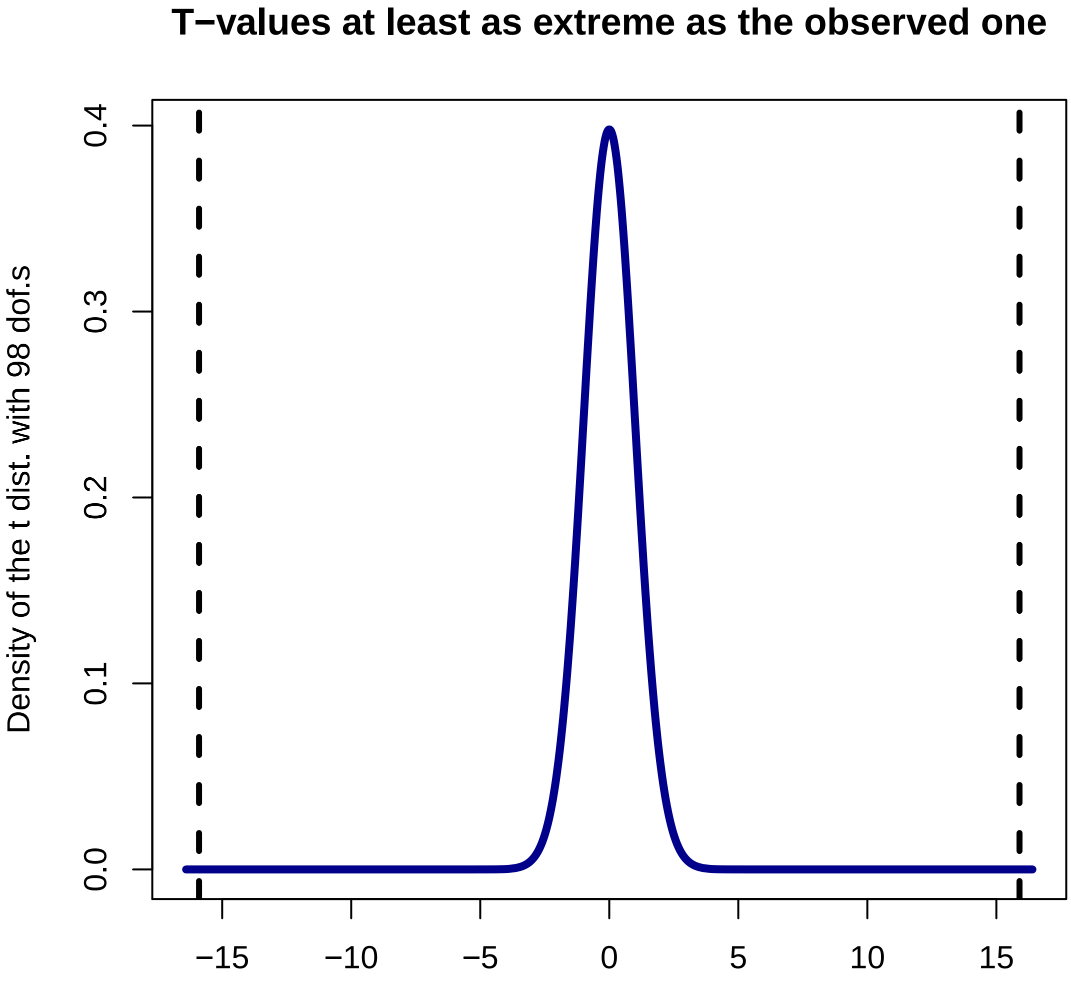
That is, the 95% CI for the population slope is roughly:

- (sample slope $-$ 2 $\cdot$ slope SE, sample slope $+$ 2 $\cdot$ slope SE)

If 0 is not in this interval, we are 95% confident that the predictor variable is statistically significant, i.e., we do significantly better at predicting y values if we take x_1 into account than if we simply take the mean of y as our model.

The probability of obtaining t-values at least as extreme as the one we observe is the p-value reported by the `reg1` output:

```
> grid_points <- seq(-(t_value_x1 + 0.5), t_value_x1 + 0.5, 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_x1, lty = 2, lwd = 3)
> abline(v = t_value_x1, lty = 2, lwd = 3)
```



```
> (t_value_x1 <- coef_x1/SE_x1)

[1] 14.14

> summary(reg1)$coef[2, 3]

[1] 14.14

> pt(-t_value_x1, df = 98)

[1] 1.065e-25

> 1 - pt(t_value_x1, df = 98)

[1] 0

> pt(-t_value_x1, df = 98) + 1 - pt(t_value_x1, df = 98)

[1] 0

> summary(reg1)$coef[2, 4]

[1] 2.13e-25

> pt(t_value_x1, df = 98)  # this value is too close to 1 for our machine precision

[1] 1

> 2 * pt(-t_value_x1, df = 98)  # alternative 1

[1] 2.13e-25

> summary(reg1)$coef[2, 4]

[1] 2.13e-25

> exp(log(2) + pt(-t_value_x1, df = 98, log.p = T))  # alternative 2 (switch to log scale)

[1] 2.13e-25

> pt(-t_value_x1, df = 98, log.p = T)  # much more precision for log probability

[1] -57.5

> log(2)  # and log(2) has enough precision too

[1] 0.6931
```

5.2 The sampling error for the intercept

Not very interesting. If the slope were 0, there would be no contribution from the x_1 predictor and the regression line would be horizontal and given by the intercept. And this would just be the mean of y, i.e., we would be back to the intercept-only / 1-mean only model, with the associated SE and 95% CI.

In general, if the slope is not 0, but the predictor (x_1) value is 0, there is no contribution from the predictor and, once again, the intercept is just the mean y.

This is why the intercept is meaningful when we have categorical predictor variables like x_2. In that case, the intercept is the mean for the $x_2 = 0$ group, i.e., the mean of the so-called reference group, or control group.

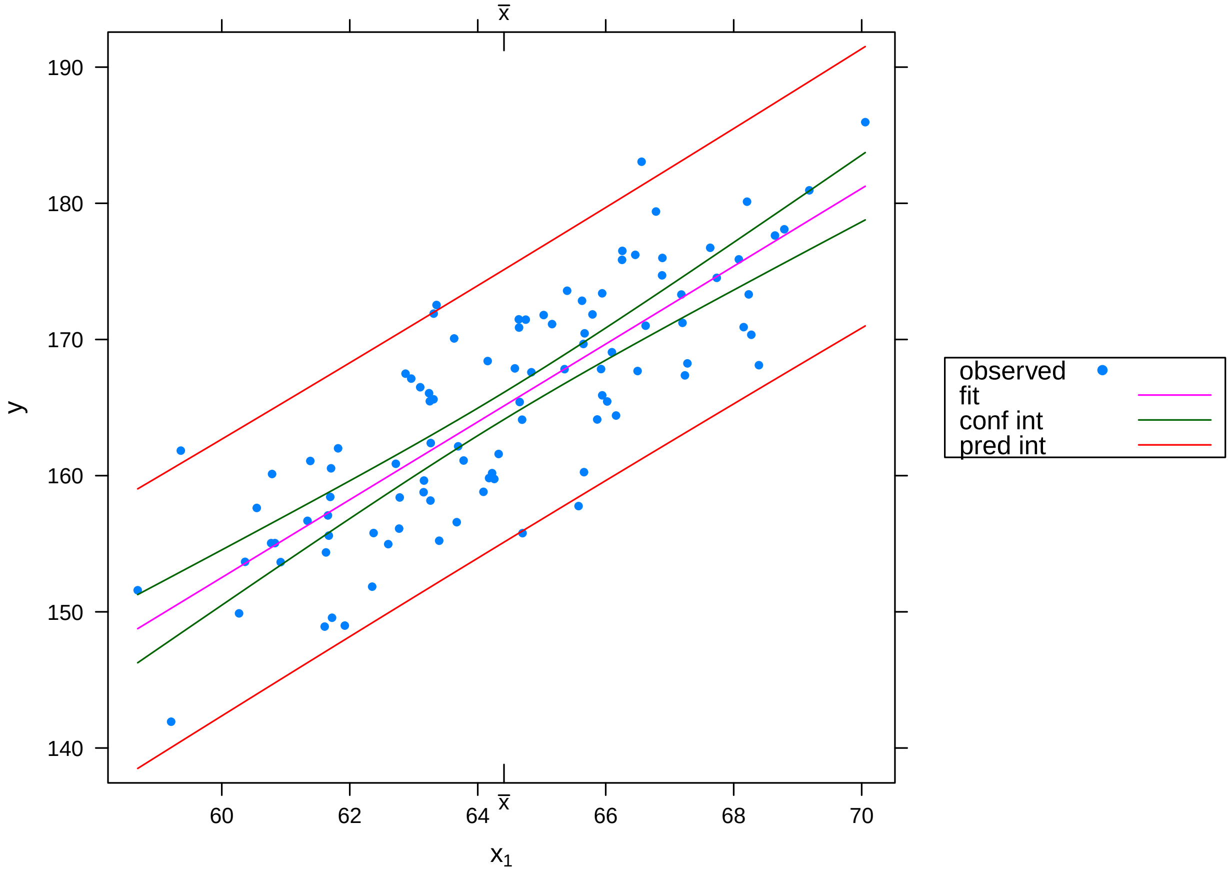
But if we have a continuous predictor and its slope is not zero (i.e., the 95% CI does not include 0), the intercept and its SE are not that meaningful. For example, in our case, the intercept would be the mean weight (y) when the mean height (x_1) is 0. But no person we will observe has a height of 0, so the intercept doesn't have a meaningful interpretation.

If we have a categorical predictor like x_2, there are observations with the 0 value for this predictor: these are all the observations in the reference / control group. So the intercept estimate and its associated SE are meaningful in such cases. And it is interesting to see how the SE is obtained; we have provided the formula for that in the previous set of lecture notes, but it is pretty complicated and we won't discuss it in more detail here.

5.3 Putting it all together: plotting predictions for linear regression models

But, whether we have continous or categorical predictors: the intercept is crucial to make correct predictions and obtain CIs for them.

```
> library("HH")
> ci.plot(lm(y ~ x1), pch = 20, main = "", lwd = 2.5, xlab = expression(x[1]))
```



Note how the 95% interval for predicted responses (red) is much wider than the 95% confidence interval for the predicted mean (green). Why?

Also note how the CIs become increasingly wider as we move away from the mean of the x_1 values in our sample (marked as $\bar{x}$ in the plot). We are less and less confident about our predictions the more they involve *extrapolation* from, rather than simply *interpolation* between, the (x_1, y) values observed in our sample.

In sum, our **model** for the population, i.e., the underlying pattern / generalization statistically inferred from the sample, is:

```
> summary(reg1)


Call:
lm(formula = y ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max
-10.684  -3.621   0.208   3.920  11.788

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -18.931     13.026   -1.45     0.15
x1             2.857      0.202   14.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.02 on 98 degrees of freedom
Multiple R-squared:  0.671,Adjusted R-squared:  0.668
F-statistic:  200 on 1 and 98 DF,  p-value: <2e-16
```

And we now know how to interpret pretty much everything in this detailed output.

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.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. L. Erlbaum Associates.

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.

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.