03-regression

nilseling

2020-05-09

Introduction

In todays session, we will focus on applied statistics in R. As an example (statistics is a huuuuuuuuge field), I will go through different regression models. In particular, these will be simple linear regression, multiple linear regression and analysis of variance. I will also try to explain the underlying mathematical derivations to some extend. This tutorial is heavily based on the instructions here.

Linear regression

In the first section, we will discuss linear regression problems using two different car-related datasets.

Simple linear regression

Here, I will give an introdcution to the most simple regression model available: the simple linear regression. I will also explain the different values returned by the model.

As an example datasets, we will use the `cars` data. `?cars` returns: The data give the speed of cars and the distances taken to stop. Note that the data were recorded in the 1920s.

```
str(cars)
```

```
## 'data.frame':    50 obs. of  2 variables:
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

In any regression problem, it is crucial to first understand the relationship between the response variable (or "dependent" variable; or target; on the y-axis) and the explanatory variable (or "independent" variable; or predictor; on the x-axis):

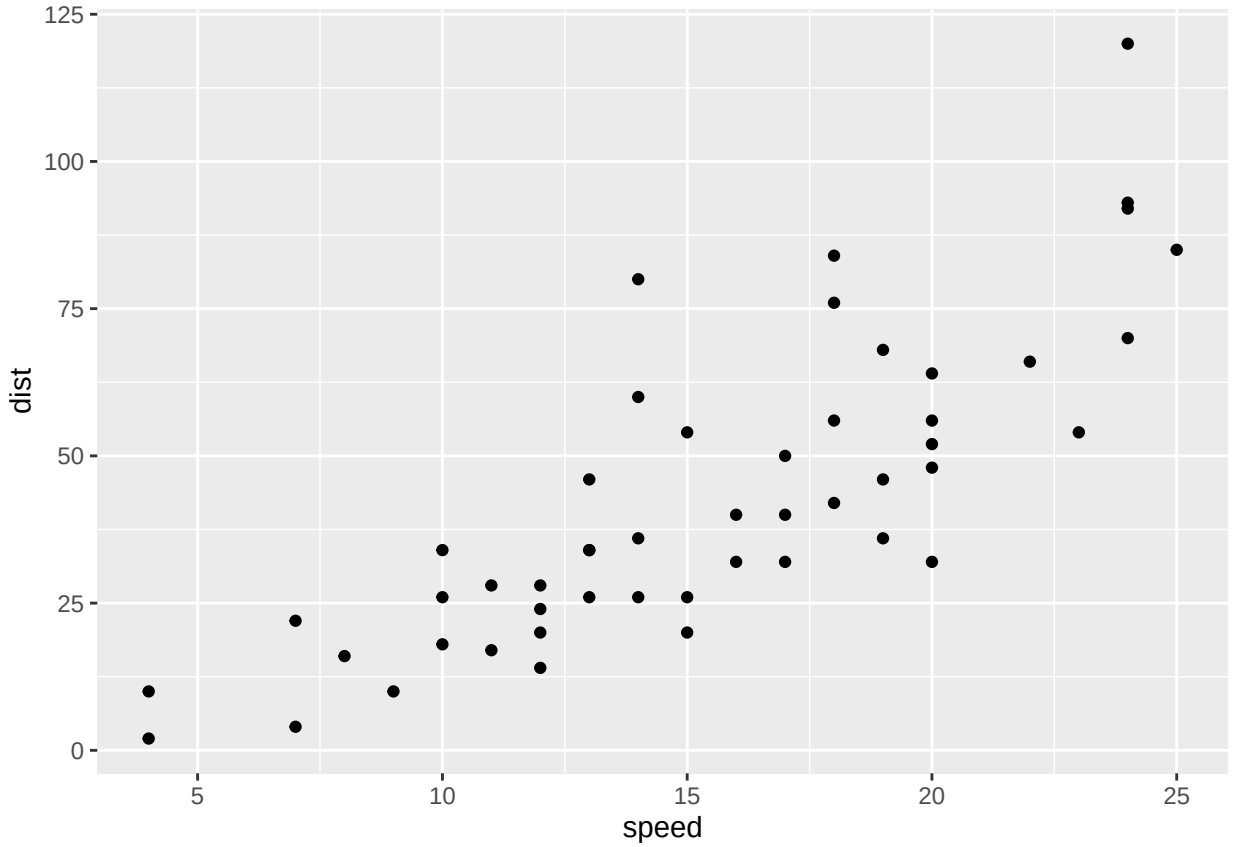
```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
cars %>%
  ggplot() + geom_point(aes(speed, dist))
```



We now want to model the relationship between the response variable `dist` and the explanatory variable `speed` in the form of:

$dist = f(speed) + \epsilon$

Here, $f(speed)$ is a function that depends on speed (or a constant) and ϵ is a "residual error" that accounts for unexplained variability in the model. By plotting the data, we know, that the data shows a linear dependence - we can therefore use a function that depends linearly on `speed` to model `dist`:

$dist = \beta_0 + \beta_1 \cdot speed + \epsilon$

In correct mathematical notation, the equation would be:

$Y = \beta_0 + \beta_1 X + \epsilon$

or:

$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$

where $\epsilon_i \sim N(0, \sigma^2)$

The last equation means that the error is a independent and identically distributed (iid) randomw variable with mean 0 and standard deviation σ^2.

To fit a linear regression, we make the following assumptions:

- Linear: the relationship between Y and x is linear
- Independent: The errors ϵ are independent
- Normal: the errors ϵ are normal distributed
- Equal variance: at each value of x, the variance of Y is equal to σ^2

Fitting a linear regression in R

To fit a linear regression in R, we can use the `lm` function of the `stats` package.

```
stop_dist_model <- lm(dist ~ speed, data = cars)
```

There are now several ways of visualizing the fitting results:

```
stop_dist_model
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Coefficients:
## (Intercept)        speed
##     -17.579        3.932
```

```
summary(stop_dist_model)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## speed         3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

Before we go into detail what these results mean, I want to introduce the `broom` package to tidying up these results:

```
library(broom)
# Tidy output of the regression parameters
broom::tidy(stop_dist_model)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -17.6      6.76      -2.60 1.23e- 2
## 2 speed           3.93     0.416      9.46 1.49e-12
```

```
# Tidy output of the fitting results
broom::glance(stop_dist_model)
```

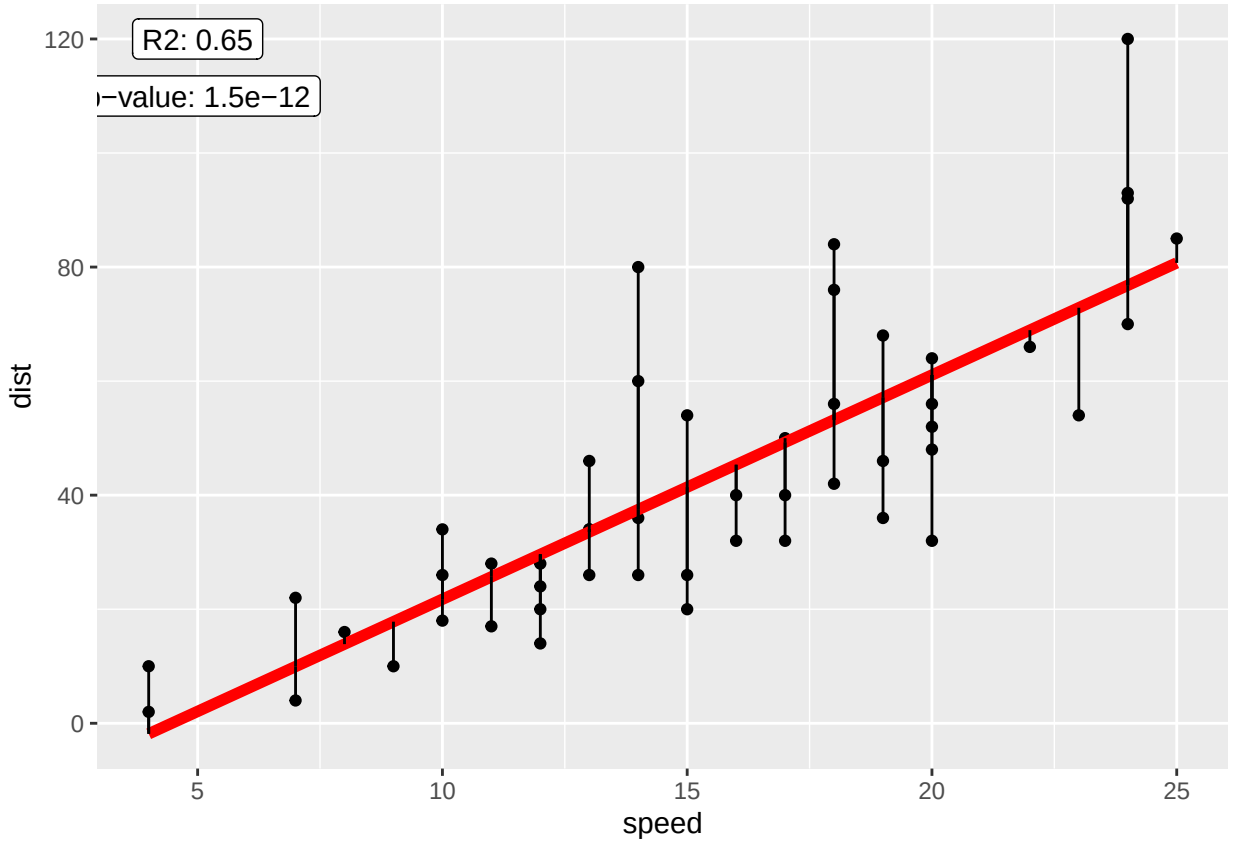
```
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
## 1     0.651         0.644  15.4      89.6 1.49e-12     2  -207.  419.  425.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```r
# Tidy output of the fitted values
head(broom::augment(stop_dist_model))
```

```
## # A tibble: 6 x 9
##    dist speed .fitted .se.fit .resid   .hat .sigma  .cooksd .std.resid
##   <dbl> <dbl>   <dbl>   <dbl>  <dbl>  <dbl>  <dbl>    <dbl>      <dbl>
## 1     2     4   -1.85    5.21   3.85 0.115    15.5 0.00459       0.266
## 2    10     4   -1.85    5.21  11.8  0.115    15.4 0.0435        0.819
## 3     4     7    9.95    4.11  -5.95 0.0715   15.5 0.00620      -0.401
## 4    22     7    9.95    4.11  12.1  0.0715   15.4 0.0255        0.813
## 5    16     8   13.9     3.77   2.12 0.0600   15.5 0.000645      0.142
## 6    10     9   17.8     3.44  -7.81 0.0499   15.5 0.00713      -0.521
```

Using the `broom` package, we can now plot the fitted line to the data:

```r
augment(stop_dist_model) %>%
  ggplot() + geom_point(aes(speed, dist)) +
  geom_line(aes(speed, .fitted), colour = "red", lwd = 2) +
  geom_segment(aes(speed, dist, xend = speed, yend = .fitted)) +
  geom_label(aes(5, 120, label = paste("R2:", round(r.squared, 2))),
             data = glance(stop_dist_model)) +
  geom_label(aes(5, 110, label = paste("p-value:", format(p.value, digits = 2))),
             data = glance(stop_dist_model))
```



Calculating the results by hand

To fully understand, what the regression results mean, we will next calculate these values by hand.

To achieve the best linear model fit, we want to minimize the sum of all squared distances between the

regression line and the data points:

$argmin \sum_{i=1}^{n}(y_i - f(x_i))^2 = argmin \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2$

Note: the error is defined as: $\epsilon_i = y_i - (\beta_0 + \beta_1 x_i)$ We basically try to minimize the sum of squared errors.

The full mathematical derivation can be found here

As a summary, we need to (i) calculate the partial derivatives of $(y_i - (\beta_0 + \beta_1 x_i))^2$ for β_0 and β_1, (ii) set them to 0 and solve the resulting set of linear equations.

The result is the following:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}$$

and

$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$

As homework, you can try to derive these equations.

Note: the "hat" on the betas means "estimate" since we do not know the true underlying regression parameters - more samples will change the estimates.

Calculating the regression coefficients in R

We define the variables x and y to be in line with the math above:

```
x <- cars$speed
y <- cars$dist

Sxy <- sum((x - mean(x)) * (y - mean(y)))
Sxx <- sum((x - mean(x)) ^ 2)

beta_hat_1 <- Sxy / Sxx
beta_hat_0 <- mean(y) - beta_hat_1 * mean(x)

c(beta_hat_0, beta_hat_1)
```

```
## [1] -17.579095   3.932409
```

```
# Compare to fitted model
stop_dist_model
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Coefficients:
## (Intercept)        speed
##     -17.579        3.932
```

Calculating the residual error in R

We can also manually calculate the residual error in R. The residuals can be obtained from the modelled fit using the `residuals` accessor function:

```
stop_dist_residuals <- residuals(stop_dist_model)

manual_residuals <- y - (beta_hat_0 + beta_hat_1 * x)
```

```r
# Test for equality
all.equal(as.numeric(stop_dist_residuals), manual_residuals)
```

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

Calculating the residual standard error

So far, we have calculated $\hat{\beta}_1$ and $\hat{\beta}_0$. Now, we want to estimate the variance σ^2 of the model. An estimate for σ^2 is defined as:

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

We can now calculate this in R:

```r
s2_e <- sum(manual_residuals^2) / (length(manual_residuals) - 2)
```

The denominatore here represents the "degrees of freedom" of the model, which in this case is defined as the number of observations minus the number of model parameters.

The "residual standard error" of the model is the standard deviation:

```r
sqrt(s2_e)
```

```
## [1] 15.37959
```

This means the estimates of the mean stopping distance are on average 15.38 meters off.

Variance decomposition

We can decompose the variation of the data points (y_i) from the sample mean ($\bar{y}$) as follows:

$y_i - \bar{y} = (y_i - \hat{y_i}) + (\hat{y_i} - \bar{y})$

By complicated calculation (see here), we can show that:

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

which we can also simply write as:

SST = SSE + SSReg

Here, SST means "sum of squares total", SSE means "sum of squares error" and SSReg means "sum of squares regression".

We can now compute the individual values:

```r
y_hat <- beta_hat_0 + beta_hat_1 * x

SST <- sum((y - mean(y)) ^ 2)
SSE <- sum((y - y_hat) ^ 2)
SSReg <- sum((y_hat - mean(y)) ^ 2)
```

We calculated these measures to derive the "coefficient of determination" also denoted as R^2.

This measure can be calculated as follows:

```r
R2 <- SSReg / SST
```

One can understand the coefficent of determination as the proportion of total variance that the model captures.

The adjusted R^2 accounts for the degrees of freedom in the model since the R^2 value by default increases when more explanatory variables are added:

$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$

where R^2 is the coefficient of determination, n is the number of samples and p is the number of explanatory variables excluding the intercept.

In R:

```
adjR2 <- 1 - (1 - R2) * ((nrow(cars) - 1) / (nrow(cars) - 1 - 1))
```

Inference of simple linear regression

Defining the regression parameters as random variables

At this point, we will not go through the mathematical derivation of the following equations. Have a look here and here for more proofs.

What we need to know is that $\hat{\beta}_0$ and$\hat{\beta}_1$ can be written as a linear combination of the response variables Y_i. We know that Y_i is normal distributed, therefore $\hat{\beta}_0$ and$\hat{\beta}_1$ are also normal distributed:

$$\hat{\beta}_1 \sim N\left(\beta_1,\ \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right).$$

and

$$\hat{\beta}_0 =\sim N\left(\beta_0,\ \frac{\sigma^2 \sum_{i=1}^{n} x_i^2}{n\sum_{i=1}^{n}(x_i - \bar{x})^2}\right).$$

We can now standardize these equations (computing the z-score):

$$\frac{\hat{\beta}_0 - \beta_0}{\mathrm{SD}[\hat{\beta}_0]} \sim N(0,1)$$

and

$$\frac{\hat{\beta}_1 - \beta_1}{\mathrm{SD}[\hat{\beta}_1]} \sim N(0,1)$$

where

$$\mathrm{SD}[\hat{\beta}_0] = \sigma\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}}}$$

and

$$\mathrm{SD}[\hat{\beta}_1] = \frac{\sigma}{\sqrt{S_{xx}}}.$$

Calculating the standard error

However, we don't know the true standard deviation σ of the model. Therefore, we will plug in the "residual standard error" as calculated above.

$$\mathrm{SE}[\hat{\beta}_0] = s_e\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}}}$$

$$\text{SE}[\hat{\beta}_1] = \frac{s_e}{\sqrt{S_{xx}}}$$

Here, $S_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2$

We can now calculate these in R:

```
Sxx <- sum((x - mean(x)) ^ 2)
s_e <- sqrt(s2_e)
n <- nrow(cars)

SE_beta_hat_0 <- s_e * sqrt(1 / n + mean(x) ^ 2 / Sxx)
SE_beta_hat_1 <- s_e / sqrt(Sxx)

c(SE_beta_hat_0, SE_beta_hat_1)
```

```
## [1] 6.7584402 0.4155128
```

```
summary(stop_dist_model)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -17.579095  6.7584402 -2.601058 1.231882e-02
## speed         3.932409  0.4155128  9.463990 1.489836e-12
```

We can proof that

$$\frac{\hat{\beta}_0 - \beta_0}{\text{SE}[\hat{\beta}_0]} \sim t_{n-2}$$

and

$$\frac{\hat{\beta}_1 - \beta_1}{\text{SE}[\hat{\beta}_1]} \sim t_{n-2}$$

follows a t distribution with $n - 2$ degrees of freedom. The t distribution is similar to a normal distribution with heavier tails.

Calculating the t-statistic

Based on the equation above, we can now calculate the t-statistic based on a given hypothesis test:

$$H_0 : \beta_0 = \beta_{00} \quad \text{vs} \quad H_1 : \beta_0 \neq \beta_{00}$$

The standard hypothesis for a linear regression model is of the form:

$$H_0 : \beta_0 = 0 \quad \text{vs} \quad H_1 : \beta_0 \neq 0$$

We can therefore calculate the t-statistic in the form:

$$t = \frac{\hat{\beta}_1 - 0}{\text{SE}[\hat{\beta}_1]}$$

And in R:

```r
t_beta_hat_0 <- (beta_hat_0 - 0) / SE_beta_hat_0
t_beta_hat_1 <- (beta_hat_1 - 0) / SE_beta_hat_1
```

Calculating the significance level

Since we know, that this measure follows a t distribution, we can now calculate the "p value". The p-value here means the probability of rejecting the null hypothesis even though the null hypothesis is true. This also considered to performing a type I error. The p-value can be calculated using the `pt` function and multiplying the value by 2 to account for a two-sided test.

```r
p_beta_hat_0 <- 2 * pt(abs(t_beta_hat_0), df = nrow(cars) - 2, lower.tail = FALSE)
p_beta_hat_0 <- 2 * pt(abs(t_beta_hat_1), df = nrow(cars) - 2, lower.tail = FALSE)
```

Multiple linear regression

ANOVA