Bentley University MA252 in pure R

Content extracted from the How to Data Website

This PDF generated on 25 January 2023

Contents

MA252 is an undergraduate statistics course at Bentley University that focuses on model building using regression. The description from the course catalog can be found here.

It covers simple linear regression, multivariate linear regression, logistic linear regression, model building, transformations, and interactions.

Content last modified on 07 December 2021.

How to fit a linear model to two columns of data

Description

Let's say we have two columns of data, one for a single independent variable x and the other for a single dependent variable y. How can I find the best fit linear model that predicts y based on x?

In other words, what are the model coefficients β_0 and β_1 that give me the best linear model $\hat{y} = \beta_0 + \beta_1 x$ based on my data?

Related tasks:

- How to compute R-squared for a simple linear model
- How to fit a multivariate linear model
- How to predict the response variable in a linear model

Solution in pure R

This solution uses fake example data. When using this code, replace our fake data with your real data.

```r
# Here is the fake data you should replace with your real data.
xs <- c( 393, 453, 553, 679, 729, 748, 817 )
ys <- c(  24,  25,  27,  36,  55,  68,  84 )

# If you need the model coefficients stored in variables for later use, do:
model <- lm( ys ~ xs )
beta0 = model$coefficients[1]
beta1 = model$coefficients[2]

# If you just need to see the coefficients, do this alone:
lm( ys ~ xs )
```

```
Call:
lm(formula = ys ~ xs)

Coefficients:
(Intercept)           xs
   -37.3214       0.1327
```

The linear model in this example is approximately $y = 0.133x - 37.32$.

Content last modified on 28 May 2021.

See a problem? Tell us or edit the source.

How to compute a confidence interval for the expected value of a response variable

Description

If we have a simple linear regression model, $y = \beta_0 + \beta_1 x + \epsilon$, where ϵ is some random error, then given any x input, y can be veiwed as a random variable because of ϵ. Let's consider its expected value. How do we construct a confidence interval for that expected value, given a value for the predictor x?

Related tasks:

- How to compute a confidence interval for a mean difference (matched pairs) (on website)
- How to compute a confidence interval for a regression coefficient
- How to compute a confidence interval for a population mean (on website)
- How to compute a confidence interval for a single population variance (on website)
- How to compute a confidence interval for the difference between two means when both population variances are known (on website)
- How to compute a confidence interval for the difference between two means when population variances are unknown (on website)
- How to compute a confidence interval for the difference between two proportions (on website)
- How to compute a confidence interval for the population proportion (on website)
- How to compute a confidence interval for the ratio of two population variances (on website)

Solution in pure R

Let's assume that you already have a linear model. We construct an example one here from some fabricated data.

```r
# Make the linear model
x <- c(34, 9, 78, 60, 22, 45, 83, 59, 25)
y <- c(126, 347, 298, 309, 450, 187, 266, 385, 400)
model <- lm(y ~ x)
```

Construct a data frame containing just one entry, the value of the independent variable for which you want to compute the confidence interval. That data frame can then be passed to R's `predict` function to get a confidence interval for the expected value of y.

```r
# Use your chosen value of x below:
data <- data.frame(x=40)
# Compute the confidence interval for y:
predict(model, data, interval="confidence", level=0.95) # or choose a different confidence level; here we use 0
```

```
  fit      lwr     upr
1 313.7217 226.648 400.7954
```

Our 95% confidence interval is $[226.648, 400.7954]$. We can be 95% confident that the true average value of y, given that x is 40, is between 226.648 and 400.7954.

Content last modified on 09 September 2021.

See a problem? Tell us or edit the source.

How to compute R-squared for a simple linear model

Description

Let's say we have fit a linear model to two columns of data, one for a single independent variable x and the other for a single dependent variable y. How can we compute R^2 for that model, to measure its goodness of fit?

Related tasks:

- How to fit a linear model to two columns of data
- How to compute adjusted R-squared

Solution in pure R

We assume you have already fit a linear model to the data, as in the code below, which is explained fully in a separate task, how to fit a linear model to two columns of data.

```
xs <- c( 393, 453, 553, 679, 729, 748, 817 )
ys <- c(  24,  25,  27,  36,  55,  68,  84 )
model <- lm( ys ~ xs )
```

You can get a lot of information about your model from its summary.

```
summary( model )
```

```
Call:
lm(formula = ys ~ xs)

Residuals:
      1       2       3       4       5       6       7 
  9.163   2.199  -9.072 -16.795  -4.431   6.047  12.890 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)   
(Intercept) -37.32142   18.99544  -1.965  0.10664   
xs            0.13272    0.02959   4.485  0.00649 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.62 on 5 degrees of freedom
Multiple R-squared:  0.8009,    Adjusted R-squared:  0.7611 
F-statistic: 20.12 on 1 and 5 DF,  p-value: 0.006486
```

In particular, it contains the R^2 value.

```
summary( model )$r.squared
```

```
[1] 0.8009488
```

Content last modified on 01 June 2021.

How to predict the response variable in a linear model

Description

If we have a linear model and a value for each explanatory variable, how do we predict the corresponding value of the response variable?

Related tasks:

- How to fit a linear model to two columns of data
- How to fit a multivariate linear model

Solution in pure R

Let's assume that you've already built a linear model. We do an example below with fake data, but you can use your own actual data. For more information on the following code, see how to fit a multivariate linear model.

```
x1 <- c( 2,  7,  4,  3, 11, 18,   6, 15,   9,  12)
x2 <- c( 4,  6, 10,  1, 18, 11,   8, 20,  16,  13)
x3 <- c(11, 16, 20,  6, 14,  8,   5, 23,  13,  10)
y  <- c(24, 60, 32, 29, 90, 45, 130, 76, 100, 120)
model <- lm(y ~ x1 + x2 + x3)
```

Let's say we want to estimate y given that $x_1 = 5$, $x_2 = 12$, and $x_3 = 50$. We can use R's `predict()` function as shown below.

```
predict(model, newdata = data.frame(x1 = 5, x2 = 12, x3 = 50))
```

```
        1 
-91.71014 
```

For the given values of the explanatory variables, our predicted response variable is -91.71014.

Note that if you want to compute the predicted values for all the data on which the model was trained, simply call `predict(model)` with no new data, and it defaults to using the training data.

```
predict(model)
```

```
        1         2         3         4         5         6         7         8 
 47.57012  24.35988  42.21531  47.27614 110.86526  70.03098  95.12690  70.91291 
        9        10 
106.52987  91.11264 
```

Content last modified on 07 December 2021.

See a problem? Tell us or edit the source.

How to fit a multivariate linear model

Description

Let's say we have several independent variables, $x_1, x_2, \ldots, x_k$, and a dependent variable y. How can I fit a linear model that uses these independent variables to best predict the dependent variable?

In other words, what are the model coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ that give me the best linear model $\hat{y} = \beta_0 + \beta_1 x + \beta_2 x + \cdots + \beta_k x$ based on my data?

Related tasks:

- How to fit a linear model to two columns of data
- How to predict the response variable in a linear model

Solution in pure R

We're going to use fake data here for illustrative purposes. You can replace our fake data with your real data in the code below.

```r
# Replace this fake data with your real data
x1 <- c(2, 7, 4, 3, 11, 18, 6, 15, 9, 12)
x2 <- c(4, 6, 10, 1, 18, 11, 8, 20, 16, 13)
x3 <- c(11, 16, 20, 6, 14, 8, 5, 23, 13, 10)
y <- c(24, 60, 32, 29, 90, 45, 130, 76, 100, 120)

# If you'll need the model coefficients later, store them as variables like this:
model <- lm(y ~ x1 + x2 + x3)
beta0 <- model$coefficients[1]
beta1 <- model$coefficients[2]
beta2 <- model$coefficients[3]
beta3 <- model$coefficients[4]

# To see the model summary, which includes the coefficients and much more, do this:
summary(model)
```

```
Call:
lm(formula = y ~ x1 + x2 + x3)

Residuals:
    Min      1Q  Median      3Q     Max 
-25.031 -20.218  -8.373  22.937  35.640 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   77.244     27.366   2.823   0.0302 *
x1            -2.701      2.855  -0.946   0.3806  
x2             7.299      2.875   2.539   0.0441 *
x3            -4.861      2.187  -2.223   0.0679 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 30.13 on 6 degrees of freedom
Multiple R-squared:  0.5936,    Adjusted R-squared:  0.3904 
F-statistic: 2.921 on 3 and 6 DF,  p-value: 0.1222
```

The coefficients and intercept appear on the left hand side of the output, about half way down, under the heading “Estimate.”

Thus the multivariate linear model from the example data is $\hat{y} = 77.244 - 2.701x_1 + 7.299x_2 - 4.861x_3$.

Content last modified on 07 December 2021.

See a problem? Tell us or edit the source.

How to add an interaction term to a model

Description

Sometimes, a simple linear model isn't sufficient for our data, and we need more complex terms or transformed variables in the model to make adequate predictions. How do we include these complex and transformed terms in a regression model?

Related tasks:

- How to add a polynomial term to a model
- How to add a transformed term to a model

Solution in pure R

We're going to use the `ToothGrowth` dataset in R as example data. It contains observations of tooth growth for guinea pigs who received various doses of various supplements. You would use your own data instead.

```r
df <- ToothGrowth
```

Let's model tooth length (`len`) based on the product of two predictors, the supplement given (`supp`) and its dosage (`dose`). We simply use the ordinary multiplication operator in R, written `*`, to express the product of these two factors when creating the model, as shown below.

Note that `supp` is a categorical variable with two values, so the model will include a binary variable for whether the supplement was equal to "VC."

```r
# Build the model
model <- lm(len ~ supp*dose, data = df)
summary(model)
```

```
Call:
lm(formula = len ~ supp * dose, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.2264 -2.8462  0.0504  2.2893  7.9386 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   11.550      1.581   7.304 1.09e-09 ***
suppVC        -8.255      2.236  -3.691 0.000507 ***
dose           7.811      1.195   6.534 2.03e-08 ***
suppVC:dose    3.904      1.691   2.309 0.024631 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.083 on 56 degrees of freedom
Multiple R-squared:  0.7296,	Adjusted R-squared:  0.7151 
F-statistic: 50.36 on 3 and 56 DF,  p-value: 6.521e-16
```

Now we have a model of the form $\hat{L} = 11.55 - 8.255s + 7.811d + 3.904sd$, where L stands for tooth length, s for whether the VC supplement was given, and d for the dose given.

How to add a polynomial term to a model

Description

Sometimes, a simple linear model isn't sufficient to describe the data. How can we include a higher-order term in a regression model, such as the square or cube of one of the predictors?

Related tasks:

- How to add a transformed term to a model
- How to add an interaction term to a model

Solution in pure R

We're going to use the `Pressure` dataset in R's `ggplot` library as example data. It contains observations of pressure and temperature. You would use your own data instead.

```
# install.packages( "ggplot2" ) # if you haven't done this already
library(ggplot2)
data("pressure")
```

Let's model temperature as the dependent variable with pressure squared as the independent variable. To place the "pressure squared" term in the model, we use R's `poly` function, as shown below. It automatically includes a pressure term as well (not squared).

```
# Build the model
model <- lm(temperature ~ poly(pressure, 2), data = pressure)
summary(model)
```

```
Call:
lm(formula = temperature ~ poly(pressure, 2), data = pressure)

Residuals:
     Min       1Q   Median       3Q      Max 
-113.095  -44.543    6.157   50.459   75.791 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)          180.00      14.31  12.581 1.03e-09 ***
poly(pressure, 2)1   361.84      62.36   5.802 2.70e-05 ***
poly(pressure, 2)2  -186.66      62.36  -2.993   0.0086 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 62.36 on 16 degrees of freedom
Multiple R-squared:  0.7271,	Adjusted R-squared:  0.693 
F-statistic: 21.31 on 2 and 16 DF,  p-value: 3.079e-05
```

Now we have a model of the form $\hat{t} = 180 + 361.84p - 186.66p^2$, where t stands for temperature and p for pressure.

You can change the number in the `poly` function. For example, if we wanted to create a third-degree polynomial term then we would have specified `poly(pressure, 3)`, and it would have included pressure, pressure squared, and pressure cubed.

How to add a transformed term to a model

Description

Sometimes, a simple linear model isn't sufficient for our data, and we need more complex terms or transformed variables in the model to make adequate predictions. How do we include these complex and transformed terms in a regression model?

Related tasks:

- How to add a polynomial term to a model
- How to add an interaction term to a model

Solution in pure R

We're going to use the `Pressure` dataset in R's `ggplot` library as example data. It contains observations of pressure and temperature. You would use your own data instead.

```r
# install.packages( "ggplot2" ) # if you haven't done this already
library(ggplot2)
data("pressure")
```

Let's model temperature as the dependent variable with the logarithm of pressure as the independent variable. To place the "log of pressure" term in the model, we use R's `log` function, as shown below. It uses the naturarl logarithm (base e).

```r
# Build the model
model.log <- lm(temperature ~ log(pressure), data = pressure)
summary(model.log)
```

```
Call:
lm(formula = temperature ~ log(pressure), data = pressure)

Residuals:
   Min     1Q Median     3Q    Max 
-28.60 -22.30 -10.13  20.00  48.61 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    153.970      6.330   24.32 1.20e-14 ***
log(pressure)   23.784      1.372   17.33 3.07e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 26.81 on 17 degrees of freedom
Multiple R-squared:  0.9464,	Adjusted R-squared:  0.9433 
F-statistic: 300.3 on 1 and 17 DF,  p-value: 3.07e-12
```

The model is $\hat{t} = 153.97 + 23.784 \log p$, where t stands for temperature and p for pressure.

Another example transformation is the square root transformation. As with `log`, just apply the `sqrt` function to the appropriate term when defining the model.

```r
# Build the model
model.sqrt <- lm(temperature ~ sqrt(pressure), data = pressure)
summary(model.sqrt)
```

```
Call:
lm(formula = temperature ~ sqrt(pressure), data = pressure)

Residuals:
   Min     1Q Median     3Q    Max 
-98.72 -34.74  11.53  42.75  56.59 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)      98.561     15.244   6.465 5.81e-06 ***
sqrt(pressure)   11.446      1.367   8.372 1.95e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 51.16 on 17 degrees of freedom
Multiple R-squared:  0.8048,	Adjusted R-squared:  0.7933 
F-statistic:  70.1 on 1 and 17 DF,  p-value: 1.953e-07
```

The model is $\hat{t} = 98.561 + 11.446\sqrt{p}$, with t and p having the same meanings as above.

Content last modified on 16 September 2021.

See a problem? Tell us or edit the source.

How to compute a confidence interval for a regression coefficient

Description

Say we have a linear regression model, either single variable or multivariate. How do we compute a confidence interval for the coefficient of one of the explanatory variables in the model?

Related tasks:

- How to compute a confidence interval for a mean difference (matched pairs) (on website)
- How to compute a confidence interval for a population mean (on website)
- How to compute a confidence interval for a single population variance (on website)
- How to compute a confidence interval for the difference between two means when both population variances are known (on website)
- How to compute a confidence interval for the difference between two means when population variances are unknown (on website)
- How to compute a confidence interval for the difference between two proportions (on website)
- How to compute a confidence interval for the expected value of a response variable
- How to compute a confidence interval for the population proportion (on website)
- How to compute a confidence interval for the ratio of two population variances (on website)

Solution in pure R

We'll assume that you have fit a single linear model to your data, as in the code below, which uses fake example data. You can replace it with your actual data.

```r
x <- c(34, 9, 78, 60, 22, 45, 83, 59, 25)
y <- c(126, 347, 298, 309, 450, 187, 266, 385, 400)
model <- lm(y ~ x)
```

We can use R's `confint()` function to find the confidence interval for the model coefficients. You can change the `level` parameter to specify a different confidence level. Note that if you have a multiple regression model, it will make confidence intervals for all of the coefficient values.

```r
confint(model, level = 0.95)   # or choose any confidence level; here we use 0.95
```

```
                2.5 %     97.5 %
(Intercept) 172.638075 535.526421
x            -4.491961   2.473935
```

The 95% confidence interval for the regression coefficient is $[-4.491961, 2.473935]$.

Content last modified on 09 September 2021.

See a problem? Tell us or edit the source.

How to compute adjusted R-squared

Description

If we have fit a multivariate linear model, how can we compute the Adjusted R^2 for that model, to measure its goodness of fit?

Related tasks:

- How to compute R-squared for a simple linear model

Solution in pure R

We assume you have already fit a multivariate linear model to the data, as in the code below. (If you're unfamiliar with how to do so, see how to fit a multivariate linear model.) The data shown below is fake, and we assume you will replace it with your own real data if you use this code.

```r
x1 <- c(2, 7, 4, 3, 11, 18, 6, 15, 9, 12)
x2 <- c(4, 6, 10, 1, 18, 11, 8, 20, 16, 13)
x3 <- c(11, 16, 20, 6, 14, 8, 5, 23, 13, 10)
y <- c(24, 60, 32, 29, 90, 45, 130, 76, 100, 120)
model <- lm(y ~ x1 + x2 + x3)
```

You can get a lot of information about your model from its summary.

```r
summary(model)
```

```
Call:
lm(formula = y ~ x1 + x2 + x3)

Residuals:
    Min      1Q  Median      3Q     Max 
-25.031 -20.218  -8.373  22.937  35.640 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   77.244     27.366   2.823   0.0302 *
x1            -2.701      2.855  -0.946   0.3806  
x2             7.299      2.875   2.539   0.0441 *
x3            -4.861      2.187  -2.223   0.0679 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 30.13 on 6 degrees of freedom
Multiple R-squared:  0.5936,	Adjusted R-squared:  0.3904 
F-statistic: 2.921 on 3 and 6 DF,  p-value: 0.1222
```

In particular, that printout contains the Adjusted R^2 value; it is the second value in the right-hand column, near the top.

You can also obtain it directly, as follows:

```
summary(model)$adj.r.squared
```

```
[1] 0.3903924
```

In this case, the Adjusted R^2 is 0.3904.

How to compute covariance and correlation coefficients

Description

Covariance is a measure of how much two variables "change together." It is positive when the variables tend to increase or decrease together, and negative when they upward motion of one variable is correlated with downward motion of the other. Correlation normalizes covariance to the interval $[-1, 1]$.

Solution in pure R

We will construct some random data here, but when applying this, you would use your own data, of course.

```r
# Create a dataframe with random values between 0 and 1
set.seed(1)
df <- as.data.frame(matrix(runif(n=50,min=0,max=1),nrow = 10))
names(df) <- c('col1','col2','col3','col4','col5')
head(df)
```

```
       col1      col2      col3      col4      col5
1 0.2655087 0.2059746 0.9347052 0.4820801 0.8209463
2 0.3721239 0.1765568 0.2121425 0.5995658 0.6470602
3 0.5728534 0.6870228 0.6516738 0.4935413 0.7829328
4 0.9082078 0.3841037 0.1255551 0.1862176 0.5530363
5 0.2016819 0.7698414 0.2672207 0.8273733 0.5297196
6 0.8983897 0.4976992 0.3861141 0.6684667 0.7893562
```

In R, we can use the `cov()` function to calculate the covariance between two variables. The default method is Pearson.

```r
cov( df$col1, df$col2 )
```

```
[1] 0.0004115864
```

You can also compare all of a DataFrame's columns among one another, each as a separate variable.

```r
cov(df)
```

```
             col1          col2          col3          col4         col5
col1  0.0996382947  0.0004115864 -0.0287090091 -0.0052485522 -0.029944309
col2  0.0004115864  0.0731549057 -0.0255386673 -0.0112688616 -0.026535785
col3 -0.0287090091 -0.0255386673  0.0942522913  0.0009465216  0.050640298
col4 -0.0052485522 -0.0112688616  0.0009465216  0.0593140088 -0.008714775
col5 -0.0299443088 -0.0265357850  0.0506402980 -0.0087147752  0.055665077
```

The Pearson correlation coefficient can be computed with `cor()` in place of `cov()`.

```r
cor(df$col1,df$col2)
```

```
[1] 0.004820878
```

And you can compute correlation coefficients for all numeric columns in a DataFrame.

```
cor(df)
```

```
     col1         col2         col3        col4        col5
col1  1.000000000  0.004820878 -0.29625051 -0.06827280 -0.4020775
col2  0.004820878  1.000000000 -0.30756049 -0.17107229 -0.4158329
col3 -0.296250506 -0.307560491  1.00000000  0.01265919  0.6991315
col4 -0.068272803 -0.171072293  0.01265919  1.00000000 -0.1516653
col5 -0.402077472 -0.415832858  0.69913152 -0.15166527  1.0000000
```

Content last modified on 10 November 2022.

See a problem? Tell us or edit the source.

How to compute the standard error of the estimate for a model

Description

One measure of the goodness of fit of a model is the standard error of its estimates. If the actual values are y_i and the estimates are $\hat{y}_i$, the definition of this quantity is as follows, for n data points.

$$\sigma_{\text{est}} = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{n}}$$

If we've fit a linear model, how do we compute the standard error of its estimates?

Solution in pure R

Let's assume that you already fit the linear model, as shown in the code below. This one uses a small amount of fake data, but it's just an example. See also how to fit a linear model to two columns of data.

```r
x <- c(34, 9, 78, 60, 22, 45, 83, 59, 25)
y <- c(126, 347, 298, 309, 450, 187, 266, 385, 400)
model <- lm(y ~ x)
```

The standard error for each estimate is shown as part of the model summary, reported by R's built-in `summary` function. See the column entitled "Std. Error" in the output below.

```r
summary(model)
```

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

Residuals:
     Min       1Q   Median       3Q      Max 
-193.776   -4.334   15.459   71.143  118.116 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)  354.082     76.733   4.614  0.00244 **
x             -1.009      1.473  -0.685  0.51536   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 107.1 on 7 degrees of freedom
Multiple R-squared:  0.06283,	Adjusted R-squared:  -0.07106 
F-statistic: 0.4693 on 1 and 7 DF,  p-value: 0.5154
```

If we need to extract just the model coefficients table, or even just the "Std. Error" column of it, we can use code like the following.

```r
coef(summary(model))
coef(summary(model))[,2]
```

```
            Estimate   Std. Error t value    Pr(>|t|)
(Intercept) 354.082248 76.732772   4.6144853 0.002441995
x            -1.009013  1.472939  -0.6850334 0.515358250




(Intercept)           x
  76.732772    1.472939
```

The standard error of the estimate for the intercept is is 76.733 and the standard error of the estimate for the slope is 1.473.

Content last modified on 08 November 2022.

See a problem? Tell us or edit the source.

How to do a hypothesis test of a coefficient's significance

Description

Let's say we have a linear model, either one variable or many. How do we conduct a test of significance for the coefficient of a single explanatory variable in the model? Similarly, how can we determine if an explanatory variable has a significant impact on the response variable?

Related tasks:

- How to compute a confidence interval for the difference between two proportions (on website)
- How to do a hypothesis test for a mean difference (matched pairs) (on website)
- How to do a hypothesis test for a population proportion (on website)
- How to do a hypothesis test for population variance (on website)
- How to do a hypothesis test for the difference between means when both population variances are known (on website)
- How to do a hypothesis test for the difference between two proportions (on website)
- How to do a hypothesis test for the mean with known standard deviation (on website)
- How to do a hypothesis test for the ratio of two population variances (on website)
- How to do a one-sided hypothesis test for two sample means (on website)
- How to do a two-sided hypothesis test for a sample mean (on website)
- How to do a two-sided hypothesis test for two sample means (on website)

Solution in pure R

We will use the fake data shown below with a single variable model. You can use a model created from your own actual data instead.

```r
x <- c( 34,   9,  78,  60,  22,  45,  83,  59,  25)
y <- c(126, 347, 298, 309, 450, 187, 266, 385, 400)
model <- lm(y ~ x)
```

We can test whether a coefficient is zero by using that as our null hypothesis, $H_0 : \beta_i = 0$. We can use any value $0 \le \alpha \le 1$ as our Type 1 error rate; we will set α to be 0.05 here.

The answer to our hypothesis test can be obtained by looking at just the coefficients portion of the model summary:

```r
summary(model)$coef
```

```
              Estimate   Std. Error t value    Pr(>|t|)
(Intercept) 354.082248 76.732772   4.6144853 0.002441995
x            -1.009013  1.472939  -0.6850334 0.515358250
```

The final column of output shows p-values for each β_i. The p-value associated with the x row is therefore for β_1, the coefficient on x. Because it is 0.515358250, which is greater than α, we cannot reject the null hypothesis, and we should continue to assume that $\beta_1 = 0$ and there is no significant relationship between the explanatory and response variable in this situation.

Content last modified on 24 October 2021.

See a problem? Tell us or edit the source.

How to do a test of joint significance

Description

If we have a multivariate linear model, how do we test the joint significance of all the variables in the model? In other words, how do we test the overall significance of the regression model?

Solution in pure R

Let's assume that you already made your multiple regression model, similar to the one shown below. You can visit this task, , to see how to construct a multivariate linear model.

Let's assume that you already made your multivariate linear model, similar to the one shown below. If you still need to create one, first see how to fit a multivariate linear model.

We use example data here, but you would use your own data instead.

```r
x1 <- c( 2,  7,  4,  3, 11, 18,   6, 15,   9,  12)
x2 <- c( 4,  6, 10,  1, 18, 11,   8, 20,  16,  13)
x3 <- c(11, 16, 20,  6, 14,  8,   5, 23,  13,  10)
y  <- c(24, 60, 32, 29, 90, 45, 130, 76, 100, 120)
model <- lm(y ~ x1 + x2 + x3)
```

Now we want to test whether the model is significant. We will use a null hypothesis that states that all of the model's coefficients are equal to zero, that is, they are not jointly significant in predicting y. We can write $H_0 : \beta_0 = \beta_1 = \beta 2 = \beta_3 = 0$.

We also choose a value $0 \leq \alpha \leq 1$ as our Type 1 error rate. Herer we'll use $\alpha = 0.05$.

The summary output for the model will give us both the F-statistic and the p-value.

```r
summary(model)
```

```
Call:
lm(formula = y ~ x1 + x2 + x3)

Residuals:
    Min      1Q  Median      3Q     Max 
-25.031 -20.218  -8.373  22.937  35.640 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   77.244     27.366   2.823   0.0302 *
x1            -2.701      2.855  -0.946   0.3806  
x2             7.299      2.875   2.539   0.0441 *
x3            -4.861      2.187  -2.223   0.0679 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 30.13 on 6 degrees of freedom
Multiple R-squared:  0.5936,	Adjusted R-squared:  0.3904 
F-statistic: 2.921 on 3 and 6 DF,  p-value: 0.1222
```

In the final line of the output, we can see that the F-statistic is 2.921. The corresponding p-value in the same line is 0.1222, which is greater than α, so we do not have sufficient evidence to reject the null hypothesis.

We cannot conclude that the independent variables in our model are jointly significant in predicting the response variable.

How to do a Spearman rank correlation test

Description

When we want to determine whether there is a relationship between two variables, but our samples do not come from normally distributed populations, we can use the Spearman Rank Correlation Test. How do we conduct it?

Solution in pure R

We will use some fake data about height and weight measurements for people. You can replace it with your real data.

Our data should be stored in R vectors, as shown below.

```r
heights <- c(60, 76, 57, 68, 70, 62, 63)
weights <- c(145, 178, 120, 143, 174, 130, 137)
```

Let's say we want to test the correlation between height (inches) and weight (pounds). Our null hypothesis would state that the Pearson correlation coefficient is equal to zero, or that there is no relationship between height and weight, $H_0 : \rho_s = 0$. We choose α, or the Type I error rate, to be 0.05 and carry out the Spearman Rank Correlation Test to get the test-statistic and p-value.

```r
# Run the Spearman Rank Correlation Test to get the test-statistic and p-value
cor.test(heights, weights, alternative = "two.sided", method = "spearman")
```

```
	Spearman's rank correlation rho

data:  heights and weights
S = 12, p-value = 0.04802
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho 
0.7857143 
```

Our p-value is 0.04802, which is less than $\alpha = 0.05$, so we reject the null hypothesis. There does appear to be a relationship between height and weight.

(This p-value is different than the one computed in the solution using Python, because different approximation methods are used by the two software packages when the sample size is small.)

Note that for a right-tailed test, you can replace "two.sided" with "greater" and for a left-tailed test, you can replace "two.sided" with "less".

Content last modified on 05 October 2021.

See a problem? Tell us or edit the source.

How to compute the residuals of a linear model

Description

If a model has been fit to a dataset, the *residuals* are the differences between the actual data points and the results the model would predict. Given a linear model and a dataset, how can we compute those residuals?

Solution in pure R

Let's assume that you've already built a linear model similar to the one below. This one uses a small amount of fake data, but it's just an example. See also how to fit a linear model to two columns of data.

```
xs <- c( 393, 453, 553, 679, 729, 748, 817 )
ys <- c(  24,  25,  27,  36,  55,  68,  84 )
model <- lm(ys ~ xs)
```

We can extract the residuals of the model in either of two ways.

R has a built-in `residuals()` function for this purpose.

```
residuals(model)
```

```
         1          2          3          4          5          6          7 
  9.162630   2.199457  -9.072500 -16.795165  -4.431143   6.047185  12.889535 
```

The model itself has a `$residuals` attribute.

```
model$residuals
```

```
         1          2          3          4          5          6          7 
  9.162630   2.199457  -9.072500 -16.795165  -4.431143   6.047185  12.889535 
```

Content last modified on 14 September 2021.

See a problem? Tell us or edit the source.