

07. Mode-Selection / Cross-Validation

Data Analysis for Networks - NDA'20
Anastasios Giovanidis

Sorbonne-LIP6



November 18, 2020

Bibliography

- B.1 Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani.
“An introduction to statistical learning: with applications in R”.
Springer Texts in Statistics. ISBN 978-1-4614-7137-0
[Chapter 2](#), [Chapter 5](#)
DOI 10.1007/978-1-4614-7138-7

Recap

In the previous course (Regression) we **assumed** that the real-world model is sufficiently described by a linear model with additive noise:

$$y = \beta_1 x + \beta_0 + \epsilon = f(x) + \epsilon$$

We estimated the unknown β 's by the parameters $\hat{\beta}_1, \hat{\beta}_0$.

The following formula **predicts** for any x

$$\hat{y} = \hat{\beta}_1 x + \hat{\beta}_0 = \hat{f}(x).$$

☞ But we do not know anything about the noise !

Prediction Errors

These predictions cannot be accurate but will always have an **irreducible** error, no matter how good the choice of the predictor \hat{f} :

$$\begin{aligned}\mathbb{E}[(y - \hat{y})^2] &= \mathbb{E}[(f(x) - \hat{f}(x))^2] + \mathbb{E}[\epsilon^2] \\ &+ 2\mathbb{E}[\epsilon(f(x) - \hat{f}(x))]. \\ &= \mathbb{E}[(f(x) - \hat{f}(x))^2] + \text{Var}(\epsilon).\end{aligned}$$

Two types of errors...

Reducible vs Irreducible

- ▶ The **irreducible error**, due to the random error ϵ in the model, whose variance is unknown.
- ▶ The **reducible error**, due to errors in the estimate of the model parameters $\hat{\beta}_i$. This type of error can be reduced by using (a) larger sample sets when estimating the coefficients, or (b) different models $\hat{f}(x)$ that better describe the unknown function $f(x)$ (could be non-linear).

In practice larger intervals are used for prediction to account for both types of errors.

Accuracy

The **MSE** is a measure of model **accuracy**.

For the **available data set** $D_n = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$,

$$MSE(D_n; \hat{\beta}_1, \hat{\beta}_0) = \frac{1}{n} \sum_{i=1}^n \left(y_i - \hat{\beta}_1 x_i - \hat{\beta}_0 \right)^2. \quad (1)$$

We guarantee maximum accuracy by

$$\min_{\hat{\beta}_1, \hat{\beta}_0} MSE(D_n; \hat{\beta}_1, \hat{\beta}_0).$$

By choosing $\hat{\beta}_1, \hat{\beta}_0$ that minimize MSE we reduce the **reducible** part, but cannot change the **irreducible** part due to noise.

Train MSE

☞ The predicted response $\hat{y}(x_i) = \hat{\beta}_1 x_i + \hat{\beta}_0$ will be close to y_i , because the parameters are chosen to minimise their difference!

We say that the model is **trained** with data D_n .

$$MSE_{train} = MSE(D_n; \hat{\beta}_1, \hat{\beta}_0).$$

But! We actually want that the model predicts good values for **unknown** data, $x_o \notin D_n$.

$$\hat{y}_o = \hat{\beta}_1 x_o + \hat{\beta}_0.$$

Test MSE

We need a **different test data set** D_m^{test} with a number $m \geq 1$ of samples, to test the accuracy of our prediction model.

For this test data set, we relate the accuracy metric

$$MSE_{\text{test}} := MSE(D_m^{\text{test}}; \hat{\beta}_1, \hat{\beta}_0) \neq MSE_{\text{train}}.$$

Question 1: How good does our "minimum MSE linear predictor" behave for the test data set?

Question 2: If we use **other** prediction models $\hat{f}(x)$, e.g. non-linear, can these predict better for the same test data set?

Polynomial Regression

The linear regression model assumes a linear relationship between the response and the input (predictors).

☞ But! the true relationship may be **non-linear**.

✎ Extend the linear model, using **polynomial regression**.

For 1-D input x we write an ℓ -polynomial model:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \dots + \beta_\ell x^\ell + \epsilon = f(x) + \epsilon. \quad (2)$$

But it is still a linear model for the parameters!

☞ If we regard $x_1 := x$, $x_2 := x^2, \dots$, $x_\ell := x^\ell$ it is just a multiple linear regression.

→ Use standard linear regression software.

Polynomial Fit

A. Giovanidis 2020

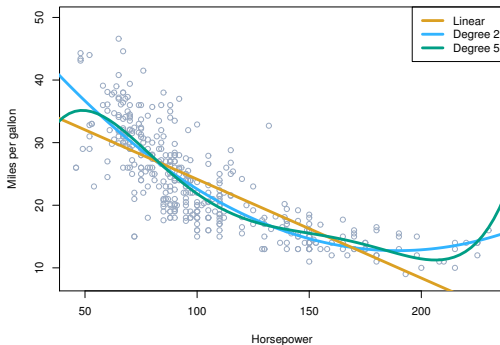


Figure: Polynomial vs Linear fit.¹

¹Source [B.1]

Flexibility VS Interpretability

Higher order polynomial models offer more flexibility.
(see Question 2)

☞ In the most extreme case we can use a model whose curve passes through every point of the train data set D_n . We can propose a polynomial fit with $\ell = \text{card}(D_n)$.

Is this a good predictive model?

☞ At the other extreme we can use a very restrictive model (simple linear regression), with $\ell = 1$.

Maybe this simple model is better?

Model Fitting I

A. Giovanidis 2020

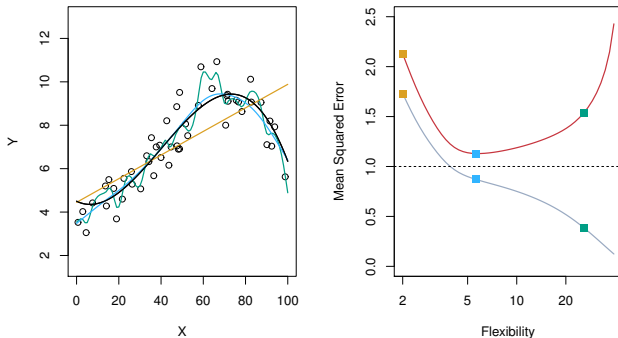


Figure: Example 1 (black curve is the real one, noise added).²

²Source [B.1]

Model Fitting II

A. Giovanidis 2020

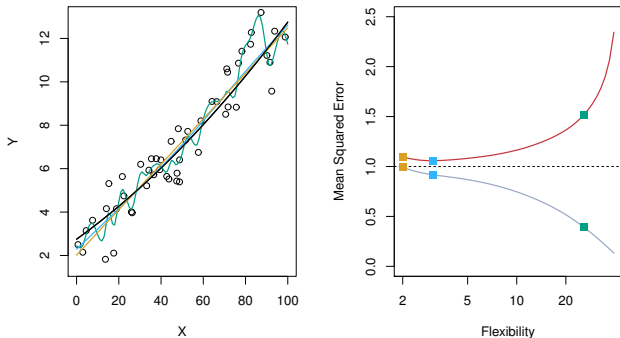


Figure: Example 2 (black curve is the real one, noise added).³

³Source [B.1]

Model Fitting III

A. Giovanidis 2020

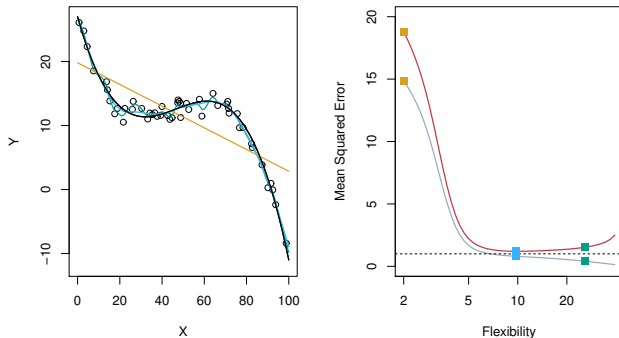


Figure: Example 3 (black curve is the real one, noise added).⁴

⁴Source [B.1]

Optimal mode selection

Answer to Question 1:

☞ The MSE_{test} is always higher than noise (irreducible error).

Answer to Question 2:

The more flexibility (higher poly-degree ℓ), the lower the MSE_{train} .

But! The MSE_{test} always has a **U** shape (fundamental property) with respect to degree (x-axis). The optimal mode is the one that **minimizes the MSE_{test}** : trade-off between flexibility vs interpretability.

☞ We call this the **Variance VS Bias trade-off**.

Overfitting / Underfitting

Overfitting: Small MSE_{train} but large MSE_{test} . The statistical learning model picks patterns that are caused by randomness rather than the true properties of the unknown $f(x)$.

☞ needs lower flexibility!

Underfitting: Large MSE_{train} and large MSE_{test} . The learning model is too rigid to accurately describe the unknown $f(x)$.

☞ needs higher flexibility!

Overfitting / Underfitting Example

A. Giovanidis 2020

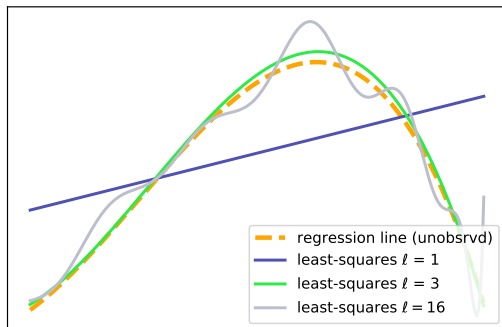


Figure: Polynomial Over/Under- fitting.

$MSE(Train) = (46.41, 25.66, 24.67)$, min train MSE for $\ell = 16$.

$MSE(Test) = (59.30, 42.94, 45.70)$, min test MSE for $\ell = 3$.

Numerical Example - Polynomial fit

Consider the numerical example with

$$D_n = \{(1, 3), (2, 4), (3, 8), (4, 14)\}.$$

- ▶ With the first three set elements do linear regression.
- ▶ Use the forth element to derive the $MSE_{test-linear}$.
- ▶ With the first three set elements do quadratic regression.
- ▶ Use the forth element to derive the $MSE_{test-quadratic}$.

Which method is best?

Numerical Example - Polynomial fit cont'd

We use the set $D_3 = \{(1, 3), (2, 4), (3, 8)\}$ for the linear regression, and we get:

- ▶ $\hat{y} = 2.50x + 0$, with an $MSE_{test-linear} = 16$.

for the quadratic regression $\beta = (\mathbf{X}\mathbf{X}^T)^{-1} \mathbf{X}^T \mathbf{y}$, and we get:

- ▶ $\hat{y} = 1.50x^2 - 3.5x + 5$, with an $MSE_{test-quadratic} = 1$.

$$MSE_{test-quadratic} < MSE_{test-linear} \Rightarrow \text{the quadratic fit is better!}$$

Resampling

In all the above, we assumed that two disjoint data-sets are available:

- ▶ a **train** data-set D_n ,
- ▶ a **test** data-set D_m^{test} ,

where $D_n \cap D_m^{test} = \emptyset$.

However, usually we only have one data-set available D_n , to both train and test the machine learning algorithm.

What should we do? a. **Cross-validation**, or b. **Bootstrapping** !

Validation set approach

Naive approach. Split the observation data set D_n in two:

- ▶ a **train** set
- ▶ a **validation** set

e.g. 80% of the elements of D_n belong to the train and the rest 20% to the test set.

- ▶ Use the **train** set to fit the model.
- ▶ Use the **validation** set to evaluate performance, e.g. MSE_{test} .

Validation set Example



Figure: n observations randomly split into a Train and Validation set.⁵

⁵Source [B.1]

Drawbacks

The method has two main problems:

1. The test error rate depends on the data split
→ MSE_{test} can be highly variable !
2. Not all available n data are used for training
→ worse performance with less observations, and MSE_{test} is overestimated !

Validation set Example

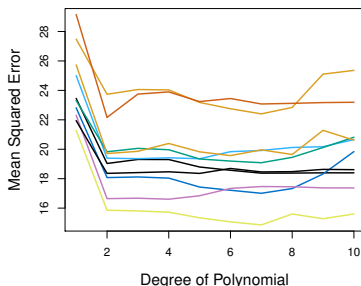
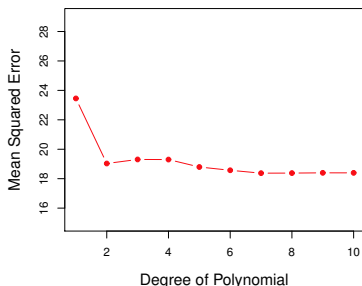


Figure: Variability of the MSE_{test} depending on the data split.⁶

⁶Source [B.1]

LOOCV

A. Giovanidis 2020

Leave-one-out-Cross-Validation. Split the data set D_n again in two:

- ▶ a validation set of a single observation (x_1, y_1)
- ▶ a train set of the rest $n - 1$ observations .

A prediction \hat{y}_1 is made only for the excluded observation using x_1 .

$$MSE_{test} = MSE_1 = (y_1 - \hat{y}_1)^2.$$

- ▶ **Problem:** The evaluation is based on a single observation \rightarrow highly variable.

LOOCV $\times n$

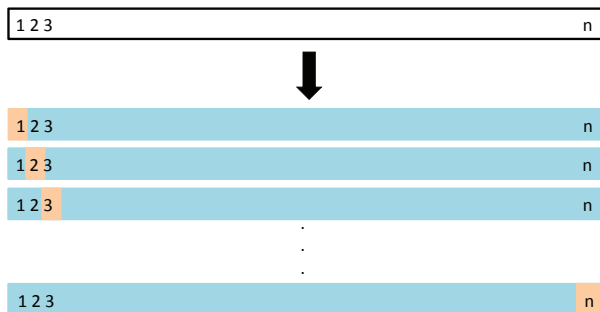


Figure: **Solution:** repeat n times, for n different splits !⁷

⁷ Source [B.1]

LOOCV $\times n$

A. Giovanidis 2020

$$\begin{array}{c|c}
 1 & 2 \ 3 \ \dots n-1 \ n \\
 2 & 1 \ 3 \ \dots n-1 \ n \\
 3 & 1 \ 2 \ \dots n-1 \ n \\
 \vdots & \vdots \\
 n & 1 \ 2 \ 3 \ \dots n-1
 \end{array}
 \begin{array}{l}
 \rightarrow MSE_1 \\
 \rightarrow MSE_2 \\
 \rightarrow MSE_3 \\
 \vdots \\
 \rightarrow MSE_n
 \end{array}$$

The LOOCV estimate for the test MSE is the **average of these n test error-estimates**

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n MSE_i.$$

☞ No randomness in the result + uses all observations !

k-fold CV

For large n LOOCV is time-consuming to calculate all MSE_i .

☞ Better use **k-fold Cross-Validation**:

- ▶ The data-set D_n is split into $1 \leq k \leq n$ folds.
- ▶ The 1st fold is treated as validation set and the rest $k - 1$ for training $\rightarrow MSE_1$ is calculated.
- ▶ Repeat k times by choosing a different fold for validation set each time $\rightarrow MSE_k$ is calculated.

The k-fold estimate is computed by **averaging**

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k MSE_i.$$

k-fold x k

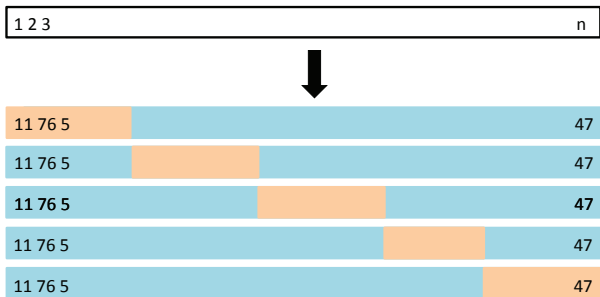


Figure: Repeat k times, for k different splits !⁸

⁸Source [B.1]

k-fold CV

Computational advantages over LOOCV:

- ▶ LOOCV is a special case of k-fold CV, for $k = n$.
- ▶ In practice $k = 5$ or $k = 10$.
- ▶ k-fold CV with $k < n$ can have lower variance in the MSE , than LOOCV.

LOOCV vs k-fold CV

A. Giovanidis 2020

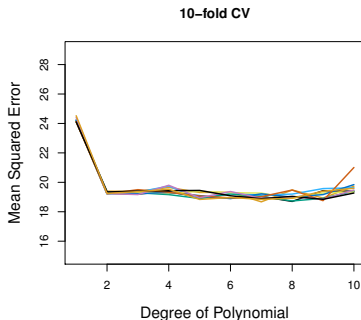
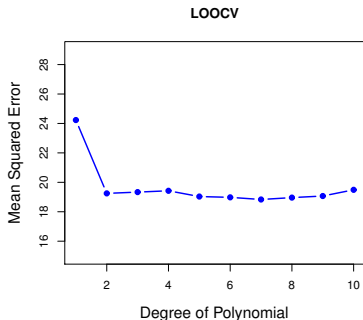


Figure: Mode selection with two CV methods⁹.

The 10-fold was run 10 times, each with a different data split.

⁹Source [B.1]

The Bootstrap

A powerful tool that can quantify the **uncertainty** associated with a given learning method.

e.g. it can estimate the standard error (SE) of $\hat{\beta}_k$, or MSE_{test}, \dots

Main idea:

Given an original data-set $Z = D_n$, create $B > 1$ new datasets of size n :

☞ each new dataset Z^{*b} results from **uniform sampling with replacement** of the set Z .

☞ for each Z^{*b} calculate the unknown $\hat{\alpha}^{*b}$.

Resampled Data Sets

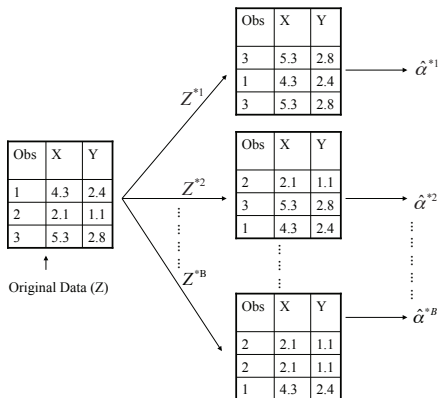


Figure: Example of Z^{*b} sets, $b = 1, 2, \dots, B$.¹⁰

¹⁰Source [B.1]

Bootstrap estimates

► The Average

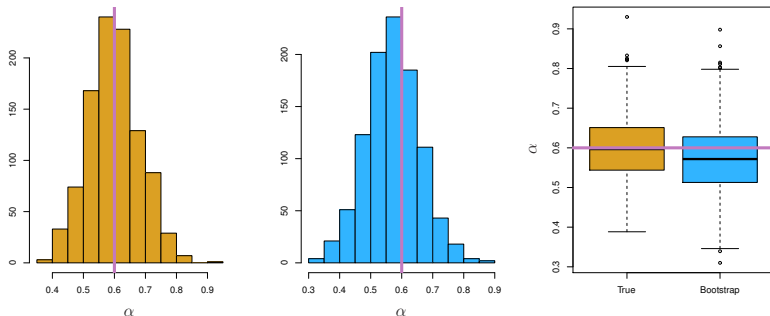
$$Av_B(\hat{\alpha}) = \bar{\alpha} = \frac{1}{B} \sum_{b=1}^B \hat{\alpha}^{*b}$$

► The Standard Error

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{(B-1)} \sum_{b=1}^B (\hat{\alpha}^{*b} - \bar{\alpha})^2}$$

Resampled stats

A. Giovanidis 2020

Figure: Real vs Bootstrapped statistics.¹¹

¹¹Source [B.1]

Numerical Example - Resampling

Consider the numerical example with $D_n = \{(1, 3), (2, 4), (3, 8), (4, 9)\}$.

Use this set and apply :

- ▶ Simple Validation.
- ▶ LOOCV.
- ▶ 2-fold CV.
- ▶ Bootstrapping.

Numerical Example - Simple Validation

Suppose first that we split the D_n in half, into a

- ▶ Train Set $\{(1, 3), (2, 4)\}$.
- ▶ Validation Set $\{(3, 8), (4, 9)\}$.

Then $\hat{y} = 1 \cdot \hat{x} + 2$.

- ▶ $MSE_{train} = 0$.
- ▶ $MSE_{test} = \frac{(5-8)^2 + (6-9)^2}{2} = 9$.

Numerical Example - LOOCV

We apply 4 times LOOCV

	Train	Validate
Fold 1	$\{(2, 4), (3, 8), (4, 9)\}$	$(1, 3)$
Fold 2	$\{(1, 3), (3, 8), (4, 9)\}$	$(2, 4)$
Fold 3	$\{(1, 3), (2, 4), (4, 9)\}$	$(3, 8)$
Fold 4	$\{(1, 3), (2, 4), (3, 8)\}$	$(4, 9)$

We get

Fold 1	$\hat{y} = 2.50x - 0.50$	$MSE_{test,1} = 1.$
Fold 2	$\hat{y} = 2.07x + 1.14$	$MSE_{test,2} = 1.638$
Fold 3	$\hat{y} = 2.07x + 0.50$	$MSE_{test,3} = 1.664$
Fold 4	$\hat{y} = 2.50x + 0.00$	$MSE_{test,4} = 1.$

$$Av(MSE_{test}) = \frac{1+1.638+1.664+1}{4} = 1.3255$$

Numerical Example - 2Folds

We apply 2-Folds (with shuffle)

	Train	Validate
Fold 1	$\{(3, 8), (2, 4)\}$	$\{(1, 3), (4, 9)\}$
Fold 2	$\{(1, 3), (4, 9)\}$	$\{(3, 8), (2, 4)\}$

We get

Fold 1	$\hat{y} = 4.00x - 4.00$	$MSE_{test,1} = 9$
Fold 2	$\hat{y} = 2.00x + 1.00$	$MSE_{test,2} = 1$

$$Av(MSE_{test}) = \frac{9+1}{2} = 5 .$$

Numerical Example - Bootstrapping

Consider the numerical example with
 $D_n = Z_1 = \{(1, 3), (2, 4), (3, 8), (4, 9)\}$.

Apply the **bootstrapping** technique for 3 more sample sets :

- ▶ $Z_2^* = \{(2, 4), (2, 4), (4, 9), (4, 9)\}$.
- ▶ $Z_3^* = \{(2, 4), (1, 3), (3, 8), (4, 9)\}$.
- ▶ $Z_4^* = \{(2, 4), (4, 9), (3, 8), (3, 8)\}$.

Numerical Example - Bootstrapping cont'd I

We get for each of the four sets,

1. $(\hat{\beta}_0, \hat{\beta}_1) = (0.5, 2.2)$.
2. $(\hat{\beta}_0, \hat{\beta}_1) = (-1.0, 2.5)$.
3. $(\hat{\beta}_0, \hat{\beta}_1) = (0.5, 2.2)$.
4. $(\hat{\beta}_0, \hat{\beta}_1) = (-0.25, 2.5)$.

Numerical Example - Bootstrapping cont'd II

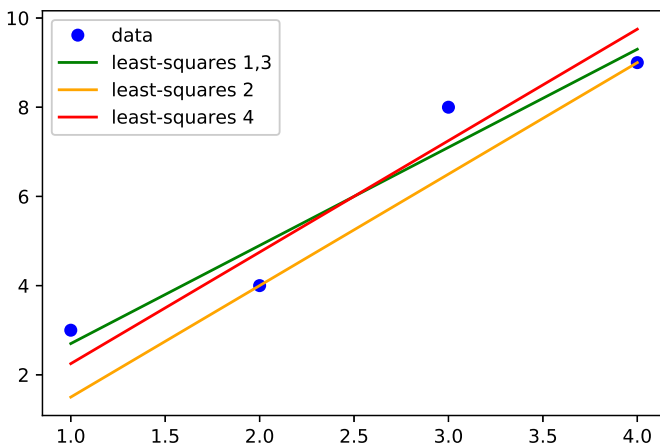


Figure: Bootstrap Regression Lines.

Numerical Example - Bootstrapping cont'd III

We have obtained the following set of estimates for the intercept and the slope

- ▶ $\hat{\beta}_0 = [0.5, -1.0, 0.5, -0.25]$.
- ▶ $\hat{\beta}_1 = [2.2, 2.5, 2.2, 2.5]$.

Then

- ▶ $\bar{\beta}_0 = Av_4(\hat{\beta}_0) = -0.0625$ and $SE_4(\hat{\beta}_0) = 0.3590$.
- ▶ $\bar{\beta}_1 = Av_4(\hat{\beta}_1) = 2.35$ and $SE_4(\hat{\beta}_1) = 0.0867$.

95% confidence intervals:

- ▶ $\beta_0 \in [-0.7806, +0.6556]$.
- ▶ $\beta_1 \in [+2.1768, +2.5232]$.

END