Bentley University GB213 in Julia

Content extracted from the How to Data Website

This PDF generated on 05 November 2022

Contents

GB213 is an undergraduate Business Statistics course at Bentley University. The description from the course catalog can be found here. Topics included in the course are listed as tasks (on website) below.

Mathematical topics include random variables, discrete and continuous probability distributions, confidence intervals, hypothesis testing, single-variable linear models, and optionally ANOVA and/or χ^2 tests, time permitting.

Basics

- How to do basic mathematical computations
- How to quickly load some sample data
- How to compute summary statistics

Random variables and probability distributions

- How to generate random values from a distribution
- How to compute probabilities from a distribution
- How to plot continuous probability distributions
- How to plot discrete probability distributions

Confidence intervals and hypothesis testing

- How to find critical values and p-values from the t-distribution
- How to find critical values and p-values from the normal distribution
- How to compute a confidence interval for a population mean
- How to do a two-sided hypothesis test for a sample mean
- How to do a two-sided hypothesis test for two sample means

Linear modeling, time permitting

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

Content last modified on 07 December 2021.

How to do basic mathematical computations

Description

How do we write the most common mathematical operations in a given piece of software? For example, how do we write multiplication, or exponentiation, or logarithms, in Python vs. R vs. Excel, and so on?

Solution in Julia

Mathematical notation	Julia code
$x + y$	`x+y`
$x - y$	`x-y`
xy	`x*y`
$\frac{x}{y}$	`x/y` (or `y\x`)
$\left\lfloor \frac{x}{y} \right\rfloor$	`x÷y`
remainder of $x \div y$	`x%y`
x^y	`x^y`
$\lvert x \rvert$	`abs(x)`
$\ln x$	`log(x)`
$\log_a b$	`log(a,b)`
e^x	`exp(x)`
π	`pi`
$\sin x$	`sin(x)`
$\sin^{-1} x$	`asin(x)`
$\sqrt{x}$	`sqrt(x)`

Other trigonometric functions are also available besides just `sin` including `cos`, `tan`, etc.

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to quickly load some sample data

Description

Sometimes you just need to try out a new piece of code, whether it be data manipulation, statistical computation, plotting, or whatever. And it's handy to be able to quickly load some example data to work with. There is a lot of freely available sample data out there. What's the easiest way to load it?

Solution in Julia

The R programming language comes with many free datasets built in. To make these same datasets available to Julia programmers as well, you can install and import the `RDatasets` package.

First, ensure that you have it installed, by running the Julia commands `using Pkg` and then `Pkg.add( "RDatasets" )`. Then you can get access to many datasets as follows:

```
using RDatasets
iris = dataset( "datasets", "iris" )
first( iris, 5 ) # just show the first 5 rows
```

{5×5 DataFrame}

Row	SepalLength Float64	SepalWidth Float64	PetalLength Float64	PetalWidth Float64	Species Cat…
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa

But what datasets are available? There are many! You can find a full list in the package itself.

```
RDatasets.packages()
```

{34×2 DataFrame}

{9 rows omitted}

Row	Package String15	Title String
1	COUNT	Functions, data and code for count data.
2	Ecdat	Data sets for econometrics
3	HSAUR	A Handbook of Statistical Analyses Using R (1st Edition)
4	HistData	Data sets from the history of statistics and data visualization
5	ISLR	Data for An Introduction to Statistical Learning with Applications in R
6	KMsurv	Data sets from Klein and Moeschberger (1997), Survival Analysis
7	MASS	Support Functions and Datasets for Venables and Ripley's MASS
8	SASmixed	Data sets from "SAS System for Mixed Models"
9	Zelig	Everyone's Statistical Software
10	adehabitatLT	Analysis of Animal Movements
11	boot	Bootstrap Functions (Originally by Angelo Canty for S)
12	car	Companion to Applied Regression

Row	Package String15	Title String
13	cluster	Cluster Analysis Extended Rousseeuw et al.
23	plm	Linear Models for Panel Data
24	plyr	Tools for splitting, applying and combining data
25	pscl	Political Science Computational Laboratory, Stanford University
26	psych	Procedures for Psychological, Psychometric, and Personality Research
27	quantreg	Quantile Regression
28	reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package.
29	robustbase	Basic Robust Statistics
30	rpart	Recursive Partitioning and Regression Trees
31	sandwich	Robust Covariance Matrix Estimators
32	sem	Structural Equation Models
33	survival	Survival Analysis
34	vcd	Visualizing Categorical Data

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to compute summary statistics

Description

The phrase "summary statistics" usually refers to a common set of simple computations that can be done about any dataset, including mean, median, variance, and some of the others shown below.

Related tasks:

- How to summarize a column (on website)
- How to summarize and compare data by groups (on website)

Solution in Julia

We first load a famous dataset, Fisher's irises, just to have some example data to use in the code that follows. (See how to quickly load some sample data.)

```
using RDatasets
iris = dataset( "datasets", "iris" );
```

How big is the dataset? The output shows number of rows then number of columns.

```
size( iris )
```

```
(150, 5)
```

What are the columns and their data types? The following command shows the first 5 rows, plus the column names and types.

```
first( iris, 5 )
```

{5×5 DataFrame}

Row	SepalLength Float64	SepalWidth Float64	PetalLength Float64	PetalWidth Float64	Species Cat…
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa

Are any values missing? The following command answers that question, plus provides summary statistics, and the same data type information from above.

```
describe( iris )
```

{5×7 DataFrame}

Row	variable Symbol	mean Union…	min Any	median Union…	max Any	nmissing Int64	eltype DataType
1	SepalLength	5.84333	4.3	5.8	7.9	0	Float64
2	SepalWidth	3.05733	2.0	3.0	4.4	0	Float64
3	PetalLength	3.758	1.0	4.35	6.9	0	Float64
4	PetalWidth	1.19933	0.1	1.3	2.5	0	Float64
5	Species		setosa		virginica	0	CategoricalValue{String, UInt8}

The individual statistics are the column headings, and the numeric columns from the original dataset are listed under the "Symbol" heading.

We can also compute these statistics (and others) one at a time for any given set of data points. Here, we let `xs` be one column from the above DataFrame, but you could use any array or DataFrame instead.

```
xs = iris."SepalLength"

using Statistics

mean( xs )            # mean, or average, or center of mass
median( xs )          # 50th percentile
quantile!( xs, 0.25 ) # compute any percentile, such as the 25th
var( xs )             # variance
std( xs )             # standard deviation, the square root of the variance
sort( xs )            # data in increasing order
sum( xs )             # sum, or total
```

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to generate random values from a distribution

Description

There are many famous continuous probability distributions, such as the normal and exponential distributions. How can we get access to them in software, to generate random values from a chosen distribution?

Related tasks:

- How to compute probabilities from a distribution
- How to plot continuous probability distributions
- How to plot discrete probability distributions

Solution in Julia

You can import many different random variables from Julia's `Distributions` package. The full list of them is online here.

If you don't have that package installed, first run `using Pkg` and then `Pkg.add( "Distributions" )` from within Julia.

Regardless of whether the distribution is discrete or continuous, the appropriate function to call is `rand`. Here are two examples.

Using a **normal distribution:**

```
using Distributions
X = Normal( 5, 3 )
rand( X, 10 )
```

```
10-element Vector{Float64}:
  3.2984814947222136
  3.8868925858958496
 11.054518475291347
  2.151820027715916
  6.288103699884755
  4.207026557133198
  2.957553317891313
  7.192469043513352
  4.440336885361706
  4.809597919062985
```

Using a **uniform distribution:**

In this example, we generate the random values in one line of code, without giving the random variable a name.

```
using Distributions
rand( Uniform( 100, 200 ), 5 )
```

```
5-element Vector{Float64}:
 147.77637653948742
 139.5897785099046
 154.48524776533654
 141.81989321071512
 116.27699388606632
```

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to compute probabilities from a distribution

Description

There are many famous continuous probability distributions, such as the normal and exponential distributions. How can we get access to them in software, to compute the probability of a value/values occurring?

Related tasks:

- How to generate random values from a distribution
- How to plot continuous probability distributions
- How to plot discrete probability distributions

Solution in Julia

You can import many different random variables from Julia's `Distributions` package. The full list of them is online here.

If you don't have that package installed, first run `using Pkg` and then `Pkg.add( "Distributions" )` from within Julia.

To compute a probability from a **discrete** distribution, create a random variable, then use the `pdf` function. (This is a slight misnomer, because PDF stands for Probability Density Function, which is a concept related to continuous random variables, but it's the function Julia uses.)

```
using Distributions

# Create a binomial random variable with 10 trials
# and probability 0.5 of success on each trial
X = Binomial( 10, 0.5 )

# What is the probability of exactly 3 successes?
pdf( X, 3 )
```

```
0.1171875000000004
```

To compute a probability from a **continuous** distribution, create a random variable, then use its Cumulative Density Function, `cdf`. You can only compute the probability that a random value will fall in an interval $[a, b]$, not the probability that it will equal a specific value.

```
using Distributions

# Create a normal random variable with mean μ=10 and standard deviation σ=5
X = Normal( 10, 5 )

# What is the probability of the value lying in the interval [12,13]?
cdf( X, 13 ) - cdf( X, 12 )
```

```
0.07032514063960227
```

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to plot continuous probability distributions

Description

There are many famous continuous probability distributions, such as the normal and exponential distributions. How can we get access to them in software, to plot the distribution as a curve?

Related tasks:

- How to generate random values from a distribution
- How to compute probabilities from a distribution
- How to plot discrete probability distributions

Solution in Julia

You can import many different random variables from Julia's `Distributions` package. The full list of them is online here.

If you don't have that package installed, first run `using Pkg` and then `Pkg.add( "Distributions" )` from within Julia.

The challenge with plotting a random variable is knowing the appropriate sample space, because some random variables have sample spaces of infinite width, which cannot be plotted.

But we can just ask Julia to show us the central 99.98% of a continuous distribution, which is almost always indistinguishable to the human eye from the entire distribution.

We style the plot below so that it is clear the sample space is continuous.

```
using Distributions
X = Normal( 10, 5 )                  # use a normal distribution with μ=10 and σ=5

xmin = quantile( X, 0.0001 )         # compute min x as the 0.0001 quantile
xmax = quantile( X, 0.9999 )         # compute max x as the 0.9999 quantile
xs = range( xmin, xmax, length=100 )  # create 100 x values in that range

using Plots
plot( xs, pdf.( X, xs ) )            # plot the shape of the distribution
```

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to plot discrete probability distributions

Description

There are many famous discrete probability distributions, such as the binomial and geometric distributions. How can we get access to them in software, to plot the distribution as a series of points?

Related tasks:

- How to generate random values from a distribution
- How to compute probabilities from a distribution
- How to plot continuous probability distributions

Solution in Julia

You can import many different random variables from Julia's `Distributions` package. The full list of them is online here.

If you don't have that package installed, first run `using Pkg` and then `Pkg.add( "Distributions" )` from within Julia.

The challenge with plotting a random variable is knowing the appropriate sample space, because some random variables have sample spaces of infinite width, which cannot be plotted.

The example below uses a geometric distribution, whose sample space is $\{1, 2, 3, ...\}$. We specify that we just want to use x values in the set $\{1, 2, ..., 10\}$. (In some software, the geometric distribution's sample space begins at 0, but not in SciPy.)

We style the plot below so that it is clear the sample space is discrete.

```
using Distributions
X = Geometric( 0.5 )      # use a geometric distribution with p=0.5
xs = 1:10                 # specify the range to be 1,2,3,...,10

using Plots
bar( xs, pdf.( X, xs ) )  # plot the shape of the distribution
```

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to find critical values and p-values from the t-distribution

Description

If we have a test statistic and need to find the corresponding p-value from the t-distribution, how do we do that? If we need to find a p-value from the t distribution, given that we know the significance level and degrees of freedom, how do we do that?

Related tasks:

- How to find critical values and p-values from the normal distribution

Solution in Julia

If we choose a value $0 \leq \alpha \leq 1$ as our Type 1 error rate, then we can find the critical value from the normal distribution using the `quantile()` function in Julia's Distributions package.

If you don't have that package installed, first run `using Pkg` and then `Pkg.add( "Distributions" )` from within Julia.

The code below shows how to do this for left-tailed, right-tailed, and two-tailed hypothesis tests.

```
using Distributions
alpha = 0.05                    # Replace with your alpha value
n = 68                          # Replace with your sample size
tdist = TDist( n - 1 )
quantile( tdist, alpha )        # Critical value for a left-tailed test
```

```
-1.6679161141074252
```

```
quantile( tdist, 1 - alpha )  # Critical value for a right-tailed test
```

```
1.6679161141074246
```

```
quantile( tdist, alpha / 2 )  # Critical value for a two-tailed test
```

```
-1.996008354025297
```

We can also compute p-values from the normal distribution to compare to a test statistic. As an example, we'll use a test statistic of 2.67, but you can substitute your test statistic's value instead.

We can find the p-value for this test statistic using the `cdf()` function in Julia's Distributions package. Again, we show code for left-tailed, right-tailed, and two-tailed tests.

```
test_statistic = 2.67                    # Replace with your test statistic
cdf( tdist, test_statistic )             # p-value for a left-tailed test
```

```
0.9952454518351646
```

```
1 - cdf( tdist, test_statistic )          # p-value for a right-tailed test
```

```
0.004754548164835448
```

```
2 * ( 1 - cdf( tdist, test_statistic ) )  # p-value for a two-tailed test
```

```
0.009509096329670896
```

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to find critical values and p-values from the normal distribution

Description

Some statistical techniques require computing critical values or p-values from the normal distribution. For example, we need to do this when constructing a confidence interval or conducting a hypothesis test. How do we compute such values?

Related tasks:

- How to find critical values and p-values from the t-distribution

Solution in Julia

If we choose a value $0 \leq \alpha \leq 1$ as our Type 1 error rate, then we can find the critical value from the normal distribution using the `quantile()` function in Julia's Distributions package.

If you don't have that package installed, first run `using Pkg` and then `Pkg.add( "Distributions" )` from within Julia.

The code below shows how to do this for left-tailed, right-tailed, and two-tailed hypothesis tests.

```
using Distributions
alpha = 0.05                            # Replace with your alpha value
standard_normal = Normal( 0, 1 )
quantile( standard_normal, alpha )      # Critical value for a left-tailed test
```

```
-1.6448536269514724
```

```
quantile( standard_normal, 1 - alpha )  # Critical value for a right-tailed test
```

```
1.6448536269514717
```

```
quantile( standard_normal, alpha / 2 )  # Critical value for a two-tailed test
```

```
-1.9599639845400592
```

We can also compute p-values from the normal distribution to compare to a test statistic. As an example, we'll use a test statistic of 2.67, but you can substitute your test statistic's value instead.

We can find the p-value for this test statistic using the `cdf()` function in Julia's Distributions package. Again, we show code for left-tailed, right-tailed, and two-tailed tests.

```
test_statistic = 2.67                           # Replace with your test statistic
cdf( standard_normal, test_statistic )          # p-value for a left-tailed test
```

```
0.9962074376523146
```

```
1 - cdf( standard_normal, test_statistic )      # p-value for a right-tailed test
```

```
0.0037925623476854353
```

```
2 * ( 1 - cdf( standard_normal, test_statistic ) )  # p-value for a two-tailed test
```

```
0.007585124695370871
```

Content last modified on 04 November 2022.

See a problem? Tell us or edit the source.

How to compute a confidence interval for a population mean

Description

If we have a set of data that seems normally distributed, how can we compute a confidence interval for the mean? Assume we have some confidence level already chosen, such as $\alpha = 0.05$.

We will use the t-distribution because we have not assumed that we know the population standard deviation, and we have not assumed anything about our sample size. If you know the population standard deviation or have a large sample size (typically at least 30), then you can use z-scores instead; see how to compute a confidence interval for a population mean using z-scores (on website).

Related tasks:

- How to compute a confidence interval for a population mean using z-scores (on website)
- How to do a two-sided hypothesis test for a sample mean
- How to do a two-sided hypothesis test for two sample means
- How to compute a confidence interval for a mean difference (matched pairs) (on website)
- How to compute a confidence interval for a regression coefficient (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 (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 Julia

When applying this technique, you would have a series of data values for which you needed to compute a confidence interval for the mean. But in order to provide code that runs independently, we create some fake data below. When using this code, replace our fake data with your real data.

```
alpha = 0.05       # replace with your chosen alpha (here, a 95% confidence level)
data = [ 435,542,435,4,54,43,5,43,543,5,432,43,36,7,876,65,5 ] # fake

# Compute the confidence interval:
using HypothesisTests
confint( OneSampleTTest( data ), level=1-alpha, tail=:both )
```

```
(70.2984781107082, 350.05446306576243)
```

Note: The solution above assumes that the population is normally distributed, which is a common assumption in introductory statistics courses, but we have not verified that assumption here.

Content last modified on 05 November 2022.

See a problem? Tell us or edit the source.

How to do a two-sided hypothesis test for a sample mean

Description

Say we have a population whose mean μ is known. We take a sample $x_1, \ldots, x_n$ and compute its mean, $\bar{x}$. We then ask whether this sample is significantly different from the population at large, that is, is $\mu = \bar{x}$?

Related tasks:

- How to compute a confidence interval for a population mean
- How to do a two-sided hypothesis test for two sample means
- How to do a one-sided hypothesis test for two sample means (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)

Solution in Julia

This is a two-sided test with the null hypothesis $H_0 : \mu = \bar{x}$. We choose a value $0 \leq \alpha \leq 1$ as the probability of a Type I error (false positive, finding we should reject H_0 when it's actually true).

```
# Replace these first three lines with the values from your situation.
alpha = 0.05
pop_mean = 10
sample = [ 9, 12, 14, 8, 13 ]

# The following code runs the test for your chosen alpha:
using HypothesisTests
p_value = pvalue( OneSampleTTest( sample, pop_mean ) )
reject_H0 = p_value < alpha
alpha, p_value, reject_H0
```

```
(0.05, 0.35845634462296455, false)
```

In this case, the p-value was larger than α, so the sample does not give us enough information to reject the null hypothesis. We would continue to assume that the sample is like the population, $\mu = \bar{x}$.

When you are using the most common value for α, which is 0.05 for the 95% confidence interval, you can simply print out the test itself and get a detailed printout with all the information you need, thus saving a few lines of code.

```
OneSampleTTest( sample, pop_mean )
```

```
One sample t-test
-----------------
Population details:
    parameter of interest:   Mean
    value under h_0:         10
    point estimate:          11.2
    95% confidence interval: (7.986, 14.41)

Test summary:
    outcome with 95% confidence: fail to reject h_0
    two-sided p-value:           0.3585

Details:
    number of observations:   5
    t-statistic:              1.0366421106976316
    degrees of freedom:       4
    empirical standard error: 1.1575836902790224
```

Content last modified on 05 November 2022.

See a problem? Tell us or edit the source.

How to do a two-sided hypothesis test for two sample means

Description

If we have two samples, $x_1, \ldots, x_n$ and $x'_1, \ldots, x'_m$, and we compute the mean of each one, we might want to ask whether the two means seem approximately equal. Or more precisely, is their difference statistically significant at a given level?

Related tasks:

- How to compute a confidence interval for a population mean
- How to do a two-sided hypothesis test for a sample mean
- How to do a one-way analysis of variance (ANOVA) (on website)
- How to do a one-sided hypothesis test for two sample means (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)

Solution in Julia

If we call the mean of the first sample $\bar{x}_1$ and the mean of the second sample $\bar{x}_2$, then this is a two-sided test with the null hypothesis $H_0 : \bar{x}_1 = \bar{x}_2$. We choose a value $0 \leq \alpha \leq 1$ as the probability of a Type I error (false positive, finding we should reject H_0 when it's actually true).

```
# Replace these first three lines with the values from your situation.
alpha = 0.10
sample1 = [ 6, 9, 7, 10, 10, 9 ]
sample2 = [ 12, 14, 10, 17, 9 ]

# Run a one-sample t-test and print out alpha, the p value,
# and whether the comparison says to reject the null hypothesis.
using HypothesisTests
p_value = pvalue( UnequalVarianceTTest( sample1, sample2 ) )
reject_H0 = p_value < alpha
alpha, p_value, reject_H0
```

```
(0.1, 0.05097283741847691, true)
```

In this case, the p-value was less than α, so the sample gives us enough evidence to reject the null hypothesis at the $\alpha = 0.10$ level. The data suggest that $\bar{x}_1 \neq \bar{x}_2$.

When you are using the most common value for α, which is 0.05 for the 95% confidence interval, you can simply print out the test itself and get a detailed printout with all the information you need, thus saving a few lines of code. Note that this gives a different answer below than the one above, because above we chose to use $\alpha = 0.10$, but the default below is $\alpha = 0.05$.

```
UnequalVarianceTTest( sample1, sample2 )
```

```
Two sample t-test (unequal variance)
------------------------------------
Population details:
    parameter of interest:   Mean difference
    value under h_0:         0
    point estimate:          -3.9
    95% confidence interval: (-7.823, 0.02309)

Test summary:
    outcome with 95% confidence: fail to reject h_0
    two-sided p-value:           0.0510

Details:
    number of observations:   [6,5]
    t-statistic:              -2.4616581720814326
    degrees of freedom:       5.720083530052662
    empirical standard error: 1.584297951775486
```

Here we did not assume that the two samples had equal variance. If in your case they do, you can use `EqualVarianceTTest()` instead of `UnequalVarianceTTest()`.

Content last modified on 05 November 2022.

See a problem? Tell us or edit the source.

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 (on website)
- How to predict the response variable in a linear model (on website)

Solution in Julia

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

```
# Here is the fake data you should replace with your real data.
xs = [ 393, 453, 553, 679, 729, 748, 817 ]
ys = [  24,  25,  27,  36,  55,  68,  84 ]

# Place the data into a DataFrame, because that's what Julia's modeling tools expect:
using DataFrames
data = DataFrame( xs=xs, ys=ys )  # Or you can name the columns whatever you like

# Create the linear model:
using GLM
lm( @formula( ys ~ xs ), data )
```

```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Float64, LinearAlgeb

ys ~ 1 + xs

Coefficients:
──────────────────────────────────────────────────────────────────────────
                 Coef.  Std. Error      t  Pr(>|t|)    Lower 95%  Upper 95%
──────────────────────────────────────────────────────────────────────────
(Intercept)  -37.3214    18.9954    -1.96    0.1066  -86.1508      11.5079
xs             0.13272    0.029589   4.49    0.0065    0.0566587    0.20878
──────────────────────────────────────────────────────────────────────────
```

The linear model in this example is approximately $y = 0.13272x - 37.3214$.

Content last modified on 05 November 2022.

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 (on website)

Solution in Julia

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.

```
using GLM, DataFrames
xs = [ 393, 453, 553, 679, 729, 748, 817 ]
ys = [  24,  25,  27,  36,  55,  68,  84 ]
data = DataFrame( xs=xs, ys=ys )
model = lm( @formula( ys ~ xs ), data )
```

```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Float64, LinearAlgeb

ys ~ 1 + xs

Coefficients:
──────────────────────────────────────────────────────────────────────────
                 Coef.  Std. Error      t  Pr(>|t|)    Lower 95%  Upper 95%
──────────────────────────────────────────────────────────────────────────
(Intercept)  -37.3214    18.9954    -1.96    0.1066  -86.1508      11.5079
xs             0.13272    0.029589   4.49    0.0065    0.0566587    0.20878
──────────────────────────────────────────────────────────────────────────
```

You can get the R^2 value from your model using the `r2` function in the GLM package.

```
r2( model )
```

```
0.8009488239830588
```

Content last modified on 05 November 2022.

See a problem? Tell us or edit the source.