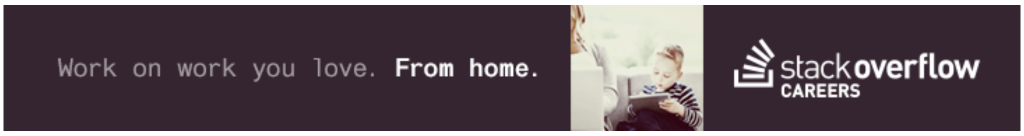
R-squared on test data



I fit a linear regression model on 75% of my data set that includes ~11000 observations and 143 variables:

```
gl.fit <- lm(y[1:ceiling(length(y)*(3/4))] ~ ., data= x[1:ceiling(length(y)*(3/4)),]) #3/4
for training
```

, and I got an R^2 of 0.43. I then tried predicting on my test data using the rest of the data:

```
ytest=y[(ceiling(length(y)*(3/4))+1):length(y)]
x.test <- cbind(1,x[(ceiling(length(y)*(3/4))+1):length(y),]) #The rest for test
yhat <- as.matrix(x.test)%*%gl.fit$coefficients  #Calculate the predicted values
```

I now would like to calculate the R^2 value on my test data. Is there any easy way to calculate that?

Thank you

r linear-regression

asked Sep 5 '14 at 17:33
H_A
97 8

See this similar question on CrossValidated. – nrussell Sep 5 '14 at 17:41

@nrussell Thanks; I used the formula in the mentioned question and got a negative number (-0.59) as my R^2 value. I have a doubt about my lm model, should I add an intercept (I assume R does it automatically)? Then why am I getting negative R^2? – H_A Sep 5 '14 at 18:06

Did you use the formula in the question or the formula in the comment below the question? Because the formula in the question is incorrect - see @Panos's comment on that question. – nrussell Sep 5 '14 at 18:41

@nrussell I used the one in comment: `errors <- (ytest - yhat)  1 - sum(errors^2)/sum((ytest-mean(ytest))^2)` and I get -0.59! – H_A Sep 5 '14 at 18:51

2 Answers

There are a couple of problems here. First, this is not a good way to use `lm(...)`. `lm(...)` is meant to be used with a data frame, with the formula expressions referencing columns in the df. So, assuming your data is in two vectors `x` and `y`,

```
set.seed(1)    # for reproducible example
x <- 1:11000
y <- 3+0.1*x + rnorm(11000,sd=1000)

df <- data.frame(x,y)
# training set
train <- sample(1:nrow(df),0.75*nrow(df))   # random sample of 75% of data

fit <- lm(y~x,data=df[train,])
```

Now `fit` has the model based on the training set. Using `lm(...)` this way allows you, for example to generate predictions without all the matrix multiplication.

The second problem is the definition of R-squared. The conventional definition is:

> 1 - SS.residuals/SS.total

For the training set, *and the training set ONLY*,

> SS.total = SS.regression + SS.residual

so

> SS.regression = SS.total - SS.residual,

and therefore

> R.sq = SS.regression/SS.total

so R.sq is the fraction of variability in the dataset that is explained by the model, and will always be between 0 and 1.

You can see this below.

```
SS.total      <- with(df[train,],sum((y-mean(y))^2))
SS.residual   <- sum(residuals(fit)^2)
SS.regression <- sum((fitted(fit)-mean(df[train,]$y))^2)
SS.total - (SS.regression+SS.residual)
# [1] 1.907349e-06
SS.regression/SS.total     # fraction of variation explained by the model
# [1] 0.08965502
1-SS.residual/SS.total     # same thing, for model frame ONLY!!!
# [1] 0.08965502
summary(fit)$r.squared     # both are = R.squared
# [1] 0.08965502
```

But this *does not* work with the test set (e.g., when you make predictions from a model).

```
test <- -train
test.pred <- predict(fit,newdata=df[test,])
test.y    <- df[test,]$y

SS.total      <- sum((test.y - mean(test.y))^2)
SS.residual   <- sum((test.y - test.pred)^2)
SS.regression <- sum((test.pred - mean(test.y))^2)
SS.total - (SS.regression+SS.residual)
# [1] 8958890

# NOT the fraction of variability explained by the model
test.rsq <- 1 - SS.residual/SS.total
test.rsq
# [1] 0.0924713

# fraction of variability explained by the model
SS.regression/SS.total
# [1] 0.08956405
```

In this contrived example there is not much difference, but it is very possible to have an R-sq. value less than 0 (when defined this way).

If, for example, the model is a very poor predictor with the test set, then the residuals can actually be larger than the total variation in test set. This is equivalent to saying that the test set is modeled better using it's mean, than using the model derived from the training set.

I noticed that you use the first three quarters of your data as the training set, rather than taking a random sample (as in this example). If the dependance of `y` on `x` is non-linear, and the `x`'s are in order, then you could get a negative R-sq with the test set.

Regarding OP's comment below, one way to assess the model with a test set is by comparing in-model to out-of-model mean squared error (MSE).

```
mse.train <- summary(fit)$sigma^2
mse.test  <- sum((test.pred - test.y)^2)/(nrow(df)-length(train)-2)
```

If we assume that the training and test set are both normally distributed with the same variance and having means which follow the same model formula, then the ratio should have an F-distribution with (n.train-2) and (n.test-2) degrees of freedom. If the MSE's are significantly different based on an F-test, then the model does *not* fit the test data well.

Have you plotted your test.y and pred.y vs x?? This alone will tell you a lot.

edited Sep 5 '14 at 22:18

answered Sep 5 '14 at 20:22
jlhoward
32.4k 3 18 45

Thank you so much for this elaborate example. In this case, what is the best way for me to assess my model on the test data set? – H_A Sep 5 '14 at 21:49

I just edited the response to bring it into alignment with the more conventional definition of R-sq, but the main conclusions are unchanged. Regarding your question, see my comments at the end. – jlhoward Sep 5 '14 at 22:17

Excellent answer, as usual. I changed my train/test set as you suggested to collect the points randomly. I am no longer getting negative R-squared for my test (assuming it has a meaning). I also calculated the training and test MSEs: 0.00056 for training, 0.00036 for test, ratio ~0.65. comparing with this: `qf(0.95,length(train)-2,length(test)-2) = 1.036603` , the model is doing something. Please correct me if I am making a mistake. – H_A Sep 5 '14 at 22:54

Well, since the out-of-model MSE is less than the MSE for the fit, it's hard to argue that the model doesn't work with the test data. Have you looked at the diagnostic plots `par(mfrow=c(2,2)); plot(fit)` ? – jlhoward Sep 5 '14 at 23:30

I did, but I am not 100% sure on how to interpret the graphs. Here is the link: dropbox.com/s/tdufnw8jqr7gfzs/diag.pdf?dl=0. Can you please comment on that? Thanks – H_A Sep 6 '14 at 0:15

If you want a function, the `miscTools` package has an `rSquared` function.

```
require(miscTools)
r2 <- rSquared(ytest, resid = ytest-yhat)
```

edited Sep 5 '14 at 19:11

answered Sep 5 '14 at 17:47
cdeterman
6,980 1 5 25

I couldn't find this package: Installing package into ‘C:/Users/Haidar/Documents/R/win-library/3.1’ (as ‘lib’ is unspecified) Warning in install.packages : package ‘micsTools’ is not available (for R version 3.1.1) – H_A Sep 5 '14 at 18:02

@H_A, typo on my part, sorry. It is `miscTools` . – cdeterman Sep 5 '14 at 19:10

Thanks, it worked, I am still getting negative value for my R^2, I suspect that there is something wrong with my regression/prediction procedure. – H_A Sep 5 '14 at 19:31