Fitting & Interpreting Linear Models in R

by yhat

May 18, 2013

`R` makes it easy to fit a linear model to your data. The hard part is knowing whether the model you've built is worth keeping and, if so, figuring out what to do next.

This is a post about linear models in `R`, how to interpret `lm` results, and common rules of thumb to help side-step the most common mistakes.

```
>>> fit <- lm(weight ~ log(length) + height + how.it.moved, data=dinos)
>>> summary(fit)

Call:
lm(formula = weight ~ log(length) + height + how.it.moved, data = dinos)

Residuals:
   Min     1Q Median     3Q    Max
 -8997  -4369  -1448   2489  49683

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                            -7380       5100   -1.45    0.151
```

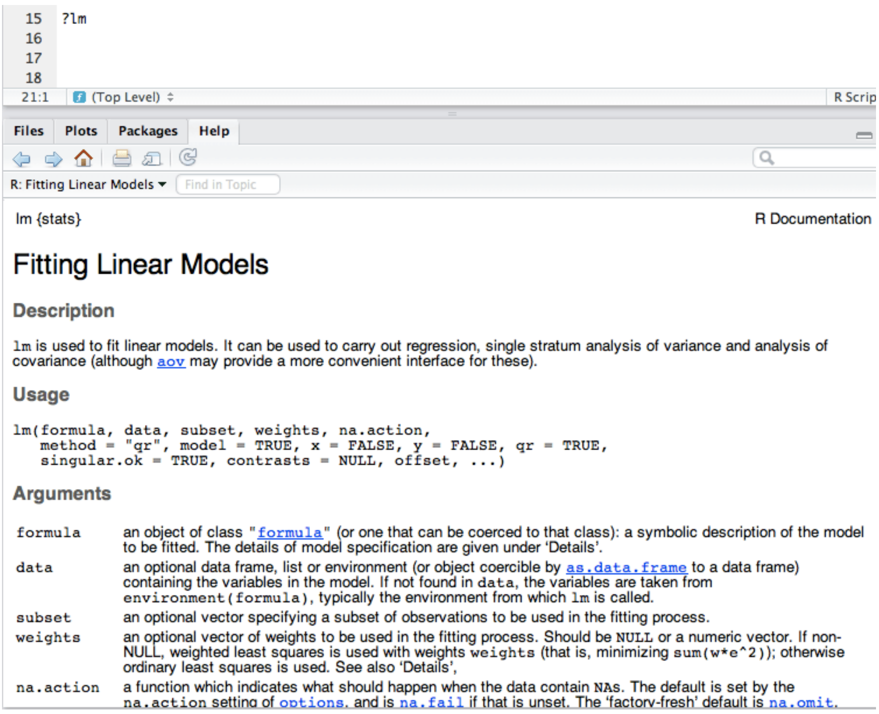
```
log(length)                        3228      1846    1.75    0.083 .
height                             1492       808    1.85    0.068 .
how.it.movedon 2 legs              1058      4925    0.21    0.830
how.it.movedon 2 legs and by flying 9319      7831    1.19    0.237
how.it.movedon 2 or 4 legs         -564      5171   -0.11    0.913
how.it.movedon 4 legs              7264      5052    1.44    0.153
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9440 on 107 degrees of freedom
Multiple R-squared:  0.391, Adjusted R-squared:  0.357
F-statistic: 11.5 on 6 and 107 DF,  p-value: 7.22e-10
```

Building a linear model in R

`R` makes building linear models really easy. Things like dummy variables, categorical features, interactions, and multiple regression all come very naturally. The centerpiece for linear regression in `R` is the `lm` function.

`lm` comes with base `R`, so you don't have to install any packages or import anything special. The documentation for `lm` is very extensive, so if you have any questions about using it, just type `?lm` into the `R` console.



Introduction to `lm`

For our example linear model, I'm going to use data from the original, or at least one of the earliest, linear regression models. The dataset consists of heights of children and their parents. The origin of the term "regression" stems from a 19th century statistician's observation that children's heights tended to "regress" towards the population mean in relation to their parent's heights.

```
galton <- read.csv("http://blog.yhathq.com/static/misc/galton.csv",
                    header=TRUE, stringsAsFactors=FALSE)
summary(galton)
#     child           parent
# Min.   :61.7   Min.   :64.0
# 1st Qu.:66.2   1st Qu.:67.5
# Median :68.2   Median :68.5
# Mean   :68.1   Mean   :68.3
# 3rd Qu.:70.2   3rd Qu.:69.5
# Max.   :73.7   Max.   :73.0

head(galton)
#  child parent
#1  61.7   70.5
#2  61.7   68.5
#3  61.7   65.5
#4  61.7   64.5
#5  61.7   64.0
#6  62.2   67.5
```

intro_to_lm_summary.R hosted with ❤ by **GitHub** **view raw**

Fit the model to the data by creating a formula and passing it to the `lm` function. In our case we want to use the parent's height to predict the child's height, so we make the formula `(child ~ parent)`. In other words, we're representing the relationship between parents' heights (X) and children's heights (y).

We then set the data being used to `galton` so `lm` knows what data frame to associate "child" and "parent" to.

```
fit <- lm(child ~ parent, data=galton)
fit
#Call:
#lm(formula = child ~ parent, data = galton)
#
#Coefficients:
#(Intercept)       parent
#     23.942        0.646
```

model_lm_summary.R hosted with ❤ by **GitHub** **view raw**

NOTE: Formulas in `R` take the form `(y ~ x)`. To add more predictor variables, just use the `+` sign. i.e. `(y ~ x + z)`.

Calling `summary`

We fit a model to our data. That's great! But the important question is, *is it any good?*

There are lots of ways to evaluate model fit. `lm` consolidates some of the most popular ways into the `summary` function. You can invoke the `summary` function on any model you've fit with `lm` and get some metrics indicating the quality of the fit.

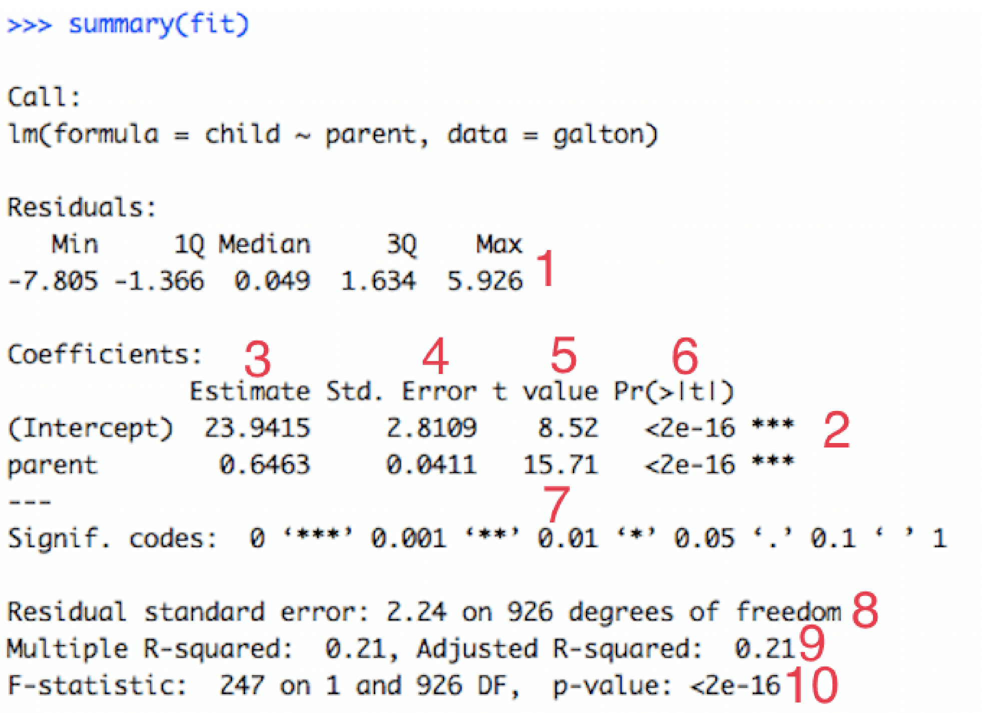
```
summary(fit)

#Call:
#lm(formula = child ~ parent, data = galton)
#
#Residuals:
#   Min     1Q Median     3Q    Max
#-7.805 -1.366  0.049  1.634  5.926
#
#Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
#(Intercept)  23.9415     2.8109    8.52   <2e-16 ***
#parent        0.6463     0.0411   15.71   <2e-16 ***
#---
#Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
#Residual standard error: 2.24 on 926 degrees of freedom
#Multiple R-squared:  0.21,    Adjusted R-squared:  0.21
#F-statistic:  247 on 1 and 926 DF,  p-value: <2e-16
```

summary_lm_summary.R hosted with ❤ by **GitHub** **view raw**

So if you're like I was at first, your reaction was probably something like "Whoa this is cool...what does it mean?"

Interpreting the output



```
>>> summary(fit)

Call:
lm(formula = child ~ parent, data = galton)

Residuals:
   Min     1Q Median     3Q    Max
-7.805 -1.366  0.049  1.634  5.926 1

Coefficients:  3          4        5      6
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  23.9415     2.8109    8.52   <2e-16 ***  2
parent        0.6463     0.0411   15.71   <2e-16 ***
---                               7
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.24 on 926 degrees of freedom 8
Multiple R-squared:  0.21, Adjusted R-squared:  0.21 9
F-statistic:  247 on 1 and 926 DF,  p-value: <2e-16 10
```

#	Name	Description
1	**Residuals**	The residuals are the difference between the actual values of the variable you're predicting and predicted values from your regression-- `y - ŷ` . For most regressions you want your residuals to look like a normal distribution when plotted. If our residuals are normally distributed, this indicates the mean of the difference between our predictions and the actual values is close to 0 (good) and that when we miss, we're missing both short and long of the actual value, and the likelihood of a miss being far from the actual value gets smaller as the distance from the actual value gets larger. Think of it like a dartboard. A good model is going to hit the bullseye some of the time (but not everytime). When it doesn't hit the bullseye, it's missing in all of the other buckets evenly (i.e. not just missing in the 16 bin) and it also misses closer to the bullseye as opposed to on the outer edges of the dartboard.
2	**Significance Stars**	The stars are shorthand for significance levels, with the number of asterisks displayed according to the p-value computed. `***` for high significance and `*` for low significance. In this case, `***` indicates that it's unlikely that no relationship exists b/w heights of parents and heights of their children.
3	**Estimated Coeffecient**	The estimated coefficient is the value of slope calculated by the regression. It might seem a little confusing that the Intercept also has a value, but just think of it as a slope that is always multiplied by 1. This number will obviously vary based on the magnitude of the variable you're inputting into the regression, but it's always good to spot check this number to make sure it seems reasonable.
4	**Standard Error of the Coefficient Estimate**	Measure of the variability in the estimate for the coefficient. Lower means better but this number is relative to the value of the coefficient. As a rule of thumb, you'd like this value to be at least an order of magnitude less than the coefficient estimate. In our example, the std error or the parent variable is 0.04 which is 16x less than the estimate of the coefficient (or 1.6 orders of magnitude greater).
5	**t-value of the Coefficient Estimate**	Score that measures whether or not the coefficient for this variable is meaningful for the model. You probably won't use this value itself, but know that it is used to calculate the p-value and the significance levels.
6	**Variable p-value**	Probability the variable is *NOT* relevant. You want this number to be as small as possible. If the number is *really* small, `R` will display it in scientific notation. In or example 2e-16 means that the odds that parent is meaningless is about $\frac{1}{5000000000000000}$
7	**Significance Legend**	The more punctuation there is next to your variables, the better. Blank=bad, Dots=pretty good, Stars=good, More Stars=very good
8	**Residual Std Error / Degrees of Freedom**	The Residual Std Error is just the standard deviation of your residuals. You'd like this number to be proportional to the quantiles of the residuals in #1. For a normal distribution, the 1st and 3rd quantiles should be 1.5 +/- the std error. The Degrees of Freedom is the difference between the number of observations included in your training sample and the number of variables used in your model (intercept counts as a variable).
9	**R-squared**	Metric for evaluating the goodness of fit of your model. Higher is better with 1 being the best. Corresponds with the amount of variability in what you're predicting that is explained by the model. In this instance, ~21% of the cause for a child's height is due to the height their parent. WARNING: While a high R-squared indicates good correlation, correlation does *not* always imply causation.
10	**F-statistic & resulting p-value**	Performs an F-test on the model. This takes the parameters of our model (in our case we only have 1) and compares it to a model that has fewer parameters. In theory the model with more parameters should fit better. If the model with more parameters (your model) doesn't perform better than the model with fewer parameters, the F-test will have a high p-value (probability *NOT* significant boost). If the model with more parameters is better than the model with fewer parameters, you will have a lower p-value. The DF, or degrees of freedom, pertains to how many variables are in the model. In our case there is one variable so there is one degree of freedom.

Categorical Variables

People often wonder how they can include categorical variables in their regression models. With `R` this is extremely easy. Just include the categorical variable in your regression formula and `R` will take care of the rest. `R` calls categorical variables `factors`. A `factor` has a set of levels, or possible values. These levels will show up as variables in the model summary.

Dummy Variable Trap

One very important thing to note is that one of your levels will not appear in the output. This is because when fitting a regression with a categorical variable, one option must be left out to avoid overfitting the model. This is often referred to as the dummy variable trap. In our model, Africa is left out of the summary but *it is still accounted for in the model.*

```r
library(reshape2)

phones <- melt(WorldPhones)
names(phones) <- c("year", "continent", "n_phones")
head(phones)
fit <- lm(n_phones ~ year + continent, data=phones)
summary(fit)
```

phones_lm_fit.R hosted with ❤ by **GitHub** **view raw**

```
>>> summary(fit)

Call:
lm(formula = n_phones ~ year + continent, data = phones)

Residuals:
   Min     1Q Median     3Q    Max
-14637  -1615   -476   1263   9655

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      -1877797     372405   -5.04  9.8e-06 ***
year                  960        190    5.05  9.7e-06 ***
continentAsia        4745       2181    2.18    0.035 *
continentEurope     32859       2181   15.07  < 2e-16 ***
continentMid.Amer    -642       2181   -0.29    0.770
continentN.Amer     65264       2181   29.92  < 2e-16 ***
continentOceania     1141       2181    0.52    0.604
continentS.Amer      1288       2181    0.59    0.558
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4080 on 41 degrees of freedom
Multiple R-squared:  0.975, Adjusted R-squared:  0.971
F-statistic:  232 on 7 and 41 DF,  p-value: <2e-16
```

Conclusion

It's often tricker to spot a bad model rather than pick out a good model. Be sure to rigorously evaluate models--don't just take the easy way out and spot check the R-squared value! R provides you with tons of different ways to check your models. For more information check out these resources:

- Advanced Interpretation of R models
- Residuals in R
- Multiple Regression in R
- Plotting lm and glm models with ggplot

BACK TO BLOG
