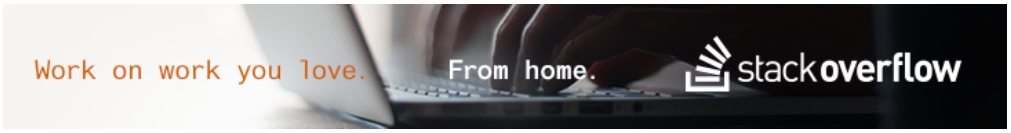
What does the R function `poly` really do?



I have read through the manual page `?poly` (which I admit I did not completely comphrehend) and also read the description of the function in book `Introduction to Statistical Learning`.

My current understanding is that a call to `poly(horsepower, 2)` should be equivalent to writing `horsepower + I(horsepower^2)`. However, this seems to be contradicted by the output of the following code.

```
library(ISLR)

summary(lm(mpg~poly(horsepower,2), data=Auto))$coef
summary(lm(mpg~horsepower+I(horsepower^2), data=Auto))$coef
```

Output:

```
                            Estimate Std. Error   t value      Pr(>|t|)
(Intercept)            23.44592  0.2209163 106.13030 2.752212e-289
poly(horsepower, 2)1 -120.13774  4.3739206 -27.46683  4.169400e-93
poly(horsepower, 2)2   44.08953  4.3739206  10.08009  2.196340e-21
                      Estimate   Std. Error   t value      Pr(>|t|)
(Intercept)     56.900099702 1.8004268063  31.60367 1.740911e-109
horsepower      -0.466189630 0.0311246171 -14.97816  2.289429e-40
I(horsepower^2)  0.001230536 0.0001220759  10.08009  2.196340e-21
```

My question is, why does the output not match, and what is `poly` really doing?

`r`

asked Oct 20 '13 at 23:22

merlin2011
29.8k 7 51 125

2 Take a look at the answer to this question: mathoverflow.net/questions/38864/… – Kieran Oct 20 '13 at 23:26

I just looked (do not fully understand), but I'd still like to know, in this context, what exactly is the closed form formula that `poly(horsepower,2)` generates? – merlin2011 Oct 20 '13 at 23:29

try `poly(horsepower, degree=2, raw=TRUE)` ; you're passing 2 as the wrong argument, and `raw` defaults to FALSE. – baptiste Oct 20 '13 at 23:51

@baptiste, your suggestion works to make `poly` generate the same output as the explicit formula, but I'd still like to know the actual form of the "orthogonal polynomial" that `poly` generates without that parameter. Also, according to the manual, I am passing 2 as degree: " Although formally 'degree' should be named (as it follows '...'), an unnamed second argument of length 1 will be interpreted as the degree." – merlin2011 Oct 20 '13 at 23:54

1 good point re `...` , still it's good practice to name arguments after it. – baptiste Oct 20 '13 at 23:57

3 Answers

When introducing polynomial terms in a statistical model the usual motivation is to determine whether the response is "curved" and whether the curvature is "significant" when that term is added in. The upshot of throwing in an `+I(x^2)` terms is that minor deviations may get "magnified" by the fitting process depending on their location, and misinterpreted as due to the

curvature term when they were just fluctuations at one end or other of the range of data. This results in inappropriate assignment of declarations of "significance".

If you just throw in a squared term with `I(x^2)`, of necessity it's also going to be highly correlated with x at least in the domain where `x > 0`. Using instead: `poly(x,2)` creates a "curved" set of variables where the linear term is not so highly correlated with x, and where the curvature is roughly the same across the range of data. (If you want to read up on the statistical theory, search on "orthogonal polynomials".) Just type `poly(1:10, 2)` and look at the two columns.

```
poly(1:10, 2)
                1           2
 [1,] -0.49543369  0.52223297
 [2,] -0.38533732  0.17407766
 [3,] -0.27524094 -0.08703883
 [4,] -0.16514456 -0.26111648
 [5,] -0.05504819 -0.34815531
 [6,]  0.05504819 -0.34815531
 [7,]  0.16514456 -0.26111648
 [8,]  0.27524094 -0.08703883
 [9,]  0.38533732  0.17407766
[10,]  0.49543369  0.52223297
attr(,"degree")
[1] 1 2
attr(,"coefs")
attr(,"coefs")$alpha
[1] 5.5 5.5

attr(,"coefs")$norm2
[1]   1.0  10.0  82.5 528.0

attr(,"class")
[1] "poly"   "matrix"
```

The "quadratic" term is centered on 5.5 and the linear term has been shifted down so it is 0 at the same x-point (with the implicit `(Intercept)` term in the model being depended upon for shifting everything back at the time predictions are requested.)

edited Dec 19 '14 at 20:47

answered Oct 20 '13 at 23:59

42-

160k 6 134 265

To get ordinary polynomials as in the question use `raw = TRUE`. Unfortunately there is an undesirable aspect with ordinary polynomials in regression. If we fit a quadratic, say, and then a cubic the lower order coefficients of the cubic are all different than for the quadratic:

```
> library(ISLR)
> fm2raw <- lm(mpg ~ poly(horsepower, 2, raw = TRUE), Auto)
> cbind(coef(fm2raw))
                                        [,1]
(Intercept)                     56.900099702
poly(horsepower, 2, raw = TRUE)1 -0.466189630
poly(horsepower, 2, raw = TRUE)2  0.001230536
>
> fm3raw <- lm(mpg ~ poly(horsepower, 3, raw = TRUE), Auto)
> cbind(coef(fm3raw))
                         [,1]
(Intercept)                       6.068478e+01
poly(horsepower, 3, raw = TRUE)1 -5.688501e-01
poly(horsepower, 3, raw = TRUE)2  2.079011e-03
poly(horsepower, 3, raw = TRUE)3 -2.146626e-06
```

What we would really like is to add the cubic term in such a way that the lower order coefficients stay the same. To do this take linear combinations of the columns of `poly(horsepower, 2, raw = TRUE)` and `poly(horsepower, 3, raw = TRUE)` such that the columns in each are orthogonal to each other. That is sufficient to guarantee that the lower order coefficients won't change when we add higher order coefficients. Note how the first three coefficients are now the same in the two sets below (whereas above they differed):

```
> fm2 <- lm(mpg ~ poly(horsepower, 2), Auto)
> cbind(coef(fm2))
                            [,1]
(Intercept)             23.44592
poly(horsepower, 2)1 -120.13774
poly(horsepower, 2)2   44.08953
> fm3 <- lm(mpg ~ poly(horsepower, 3), Auto)
> cbind(coef(fm3))
                             [,1]
```

```
(Intercept)            23.445918
poly(horsepower, 3)1 -120.137744
poly(horsepower, 3)2   44.089528
poly(horsepower, 3)3   -3.948849
```

Importantly, since the columns of `poly(horsepwer, 2)` are just linear combinations of the columnns of `poly(horsepower, 2, raw = TRUE)` the two represent the same models except for parameterization and therefore give rise to the same predictions. For example, the fitted values are the same:

```
> all.equal(fitted(fm2), fitted(fm2raw))
[1] TRUE
```

We can also verify that the polynomials do have orthogonal columns which are also orthogonal to the intercept:

```
> nr <- nrow(Auto)
> e <- rep(1, nr) / sqrt(nr) # constant vector of unit length
> p <- cbind(e, poly(Auto$horsepower, 2))
> round(crossprod(p), 2)
    1 2
  1 0 0
1 0 1 0
2 0 0 1
```

edited Oct 21 '13 at 13:20

answered Oct 21 '13 at 0:35

G. Grothendieck
80.6k 3 66 142

A quick answer is that `poly` of a vector is `x` essentially equivalent to the QR decomposition of the matrix whose columns are powers of `x` (after centering). For example:

```
> x<-rnorm(50)
> x0<-sapply(1:5,function(z) x^z)
> x0<-apply(x0,2,function(z) z-mean(z))
> x0<-qr.Q(qr(x0))
> cor(x0,poly(x,5))
                1             2             3             4             5
[1,] -1.000000e+00 -1.113975e-16 -3.666033e-17  7.605615e-17 -1.395624e-17
[2,] -3.812474e-17  1.000000e+00  1.173755e-16 -1.262333e-17 -3.988085e-17
[3,] -7.543077e-17 -7.778452e-17  1.000000e+00  3.104693e-16 -8.472204e-17
[4,]  1.722929e-17 -1.952572e-16  1.013803e-16 -1.000000e+00 -1.611815e-16
[5,] -5.973583e-17 -1.623762e-18  9.163891e-17 -3.037121e-16  1.000000e+00
```

answered Oct 21 '13 at 0:52

mrip
8,458 1 13 29
