multiple_polynomial_regression

September 8, 2019

0.1 Multiple regression

Lets look at a dataset made up of 4 variables: x, y, z, and r.

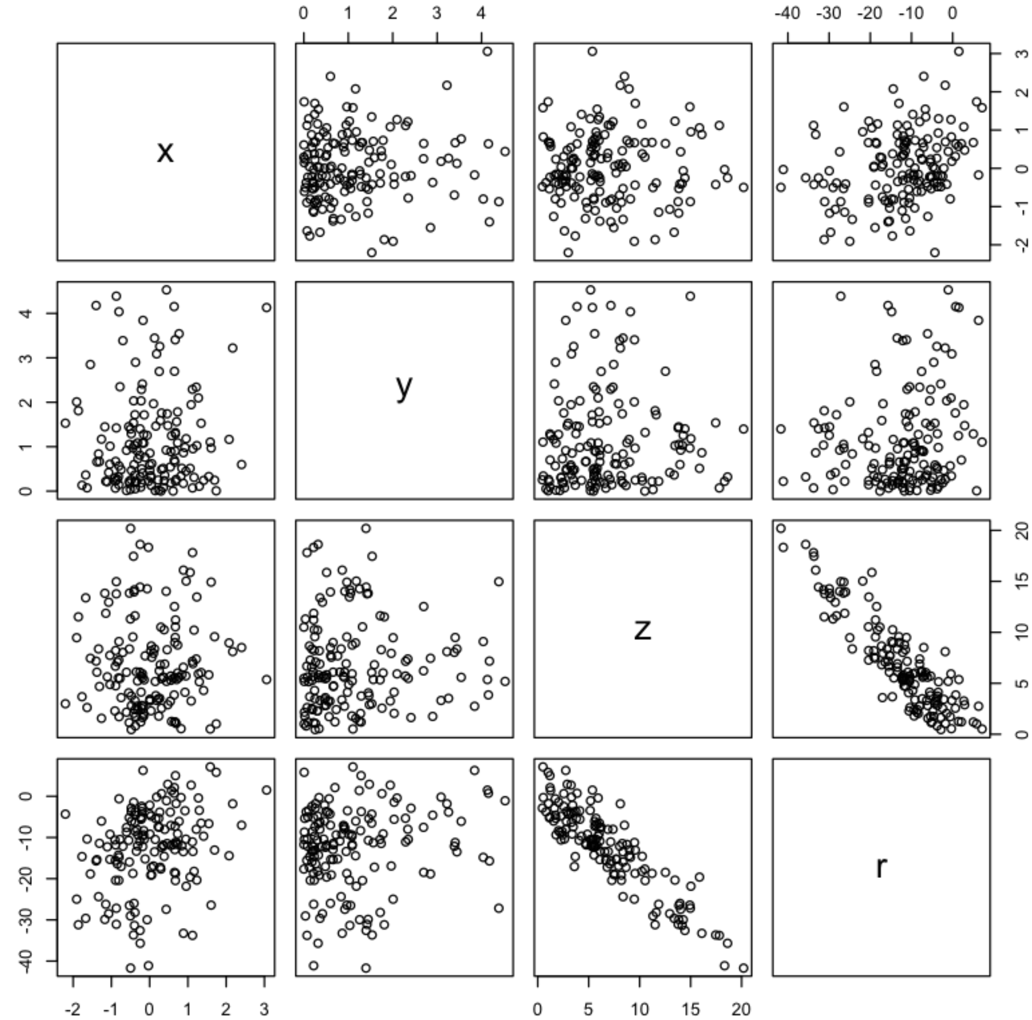
[2]:
```
dta <- read.csv('multipleData.csv')
head(dta)
```

A data.frame: 6 × 4

x	y	z	r
<dbl>	<dbl>	<dbl>	<dbl>
-0.9082393	0.30143444	2.287881	-10.61071
-0.2554406	0.80946626	6.093306	-10.86466
1.2185192	2.33952550	7.383286	-8.01159
-1.3996301	4.17670312	7.174199	-15.69763
0.9467331	0.44573692	5.719590	-10.40901
-0.2311548	0.08942789	1.201217	-5.03375

Just like simple linear regression, we can gain intuition by plotting all variables in a multiple linear regression to one another.

[75]:
```
plot(dta)
```



Sometimes this plot is call a **draftsman plot**. We notice a few interesting relationships * r and z are related negatively. Increasing values of z correspond to decreasing values of r * x is modestly related to r, y, and z positively. Increasing values of x corrspond to increasing values of y, z, and r

A **multiple linear regression (sometimes called a multivariate regression)** relates changes in $>$1 variable (the right-hand side of the equals signs) to a response variable, or variable to predict, or variable to explain (the left hand side of the equals sign). We can write this down in model form as

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots \beta_n x_{in} + \epsilon_i \tag{1}$$
$$\epsilon \sim N(0, \sigma^2), \tag{2}$$

and using matrices and vectors,

$$
\begin{aligned}
y &= X\beta + \epsilon && (3)\\
\epsilon &\sim N(0, \sigma^2), && (4)
\end{aligned}
$$

where

$$
\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} ; X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{i,1} & x_{i,2} & \cdots & x_{i,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{M,1} & x_{M,2} & \cdots & x_{M,N} \end{bmatrix}
$$

The parameters for each variable (β) are stacked into a single vector.

The Matrix X assigns one column per variable, and a column of 1s to represent the intercept.

We can also write the above multiple regression in probabilistic form

$$
y \sim N(X\beta, \sigma^2)
$$

Notice the same equation is used to represent simple linear regression and multiple linear regression. The differences between simple and multiple regression are folded into X and β.

0.2 Polynomial regression

We've discussed simple and multiple linear regression. Now lets discuss how linear regression can apply to nonlinear relationships.

Polynomial regression supposes the functional form relating y and x is a polynomial

$$
\begin{aligned}
y_i &= f(x_i|\beta) \\
&= \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots \beta_n x_i^n + \epsilon_i \\
\epsilon_i &\sim N(0, \sigma^2)
\end{aligned}
$$

We can rewrite this equation in matrix form

$$
\begin{aligned}
y_i &= f(x_i|\beta) \\
&= X\beta + \epsilon \\
\epsilon &\sim N(0, \sigma^2)
\end{aligned}
$$

where

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}; X = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_i & x_i^2 & \cdots & x_i^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_M & x_M^2 & \cdots & x_M^N \end{bmatrix}$$

Here the matrix X looks slightly different than in multiple regression. The first column of 1s is a place-holder for an intercept. The second column is the linear term x_i, the next column a quadratic term x_i^2, next column a cubic term x_i^3 and so on. The same observation x is raised to higher powers as we move across columns.

Now lets look at an example.

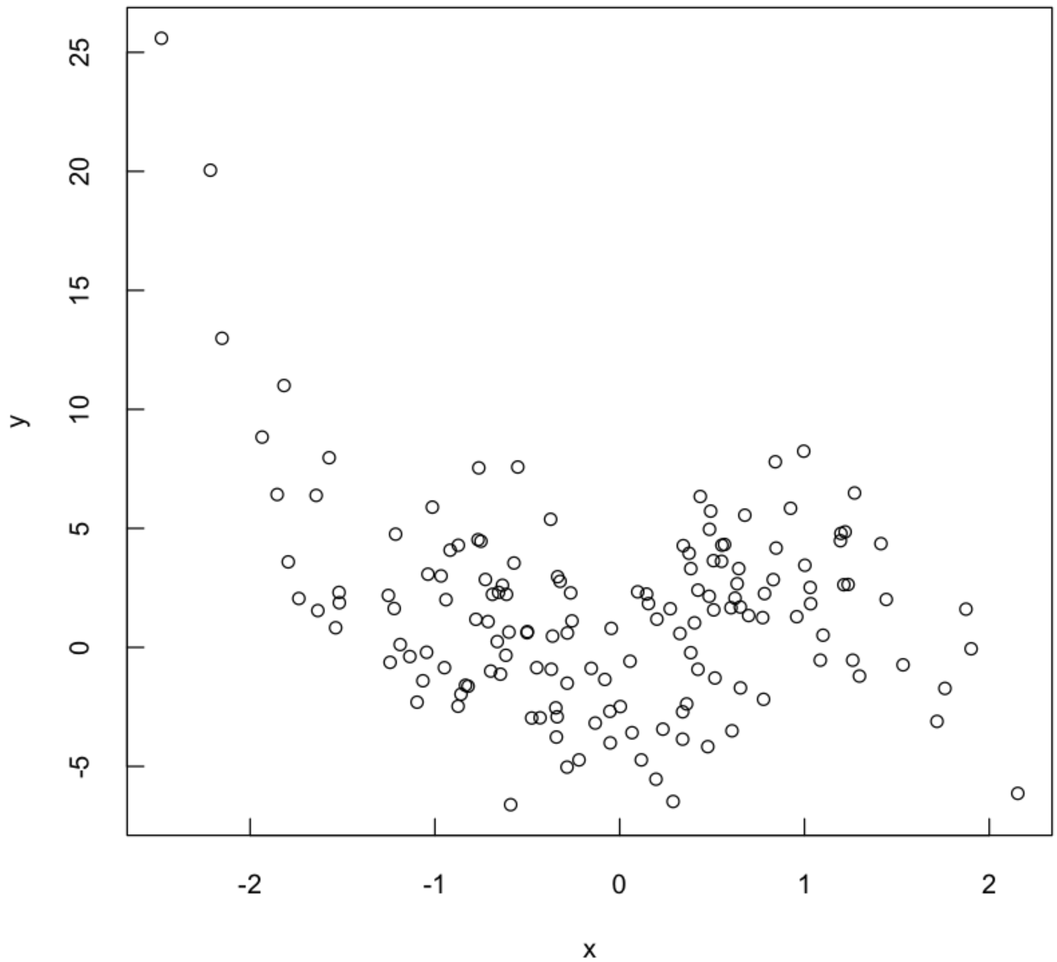
[4]:
```
data <- read.csv('polynomialData.csv')
head(data)
```

A data.frame: 6 × 2

x	y
<dbl>	<dbl>
0.9958723	8.2420054
-0.6556163	2.3114202
-0.9176787	4.0842076
0.1963727	-5.5386897
1.0309346	2.5166174
1.2610719	-0.5388713

Here we just have x values and y values. The difference, they're not related to each other linearly.

[5]:
```
plot(data$x,data$y
      ,xlab="x"
      ,ylab="y"
      ,tck=0.02
)
```



The variable x does not relate to y linearly; fitting a straight line to this data may under-represent the more complicated relationship.

Lets fit three different models: a linear model, a quadratic model, and a cubic model. We can plot each model fit and evaluate which model looks like it best represents the relationship between x and y.

[7]:
```
simpleLinearRegression <- lm(y~x,data=data)
print(simpleLinearRegression)

quadraticRegression <- lm(y~x+I(x^2),data=data)
print(quadraticRegression)

cubicRegression <- lm(y~x+I(x^2)+I(x^3),data=data)
```

```
print(cubicRegression)
```

```
Call:
lm(formula = y ~ x, data = data)

Coefficients:
(Intercept)            x
      1.453       -1.253


Call:
lm(formula = y ~ x + I(x^2), data = data)

Coefficients:
(Intercept)            x       I(x^2)
   -0.08514     -0.81522      1.72492


Call:
lm(formula = y ~ x + I(x^2) + I(x^3), data = data)

Coefficients:
(Intercept)            x       I(x^2)       I(x^3)
     0.2994       2.2877       1.0962      -1.4120
```

[8]:
```
plot(data$x,data$y
     ,xlab="x"
     ,ylab="y"
     ,tck=0.02
)

minX <- min(data$x)
maxX <- max(data$x)

#linear regression
betas <- coef(simpleLinearRegression)
beta0 <- betas[1]
beta1 <- betas[2]

xVals <- seq(minX,maxX,0.01)
linearYPredictions <- beta0 + beta1*xVals
lines(xVals,linearYPredictions,col='red')

#quadratic regression
betas <- coef(quadraticRegression)
```

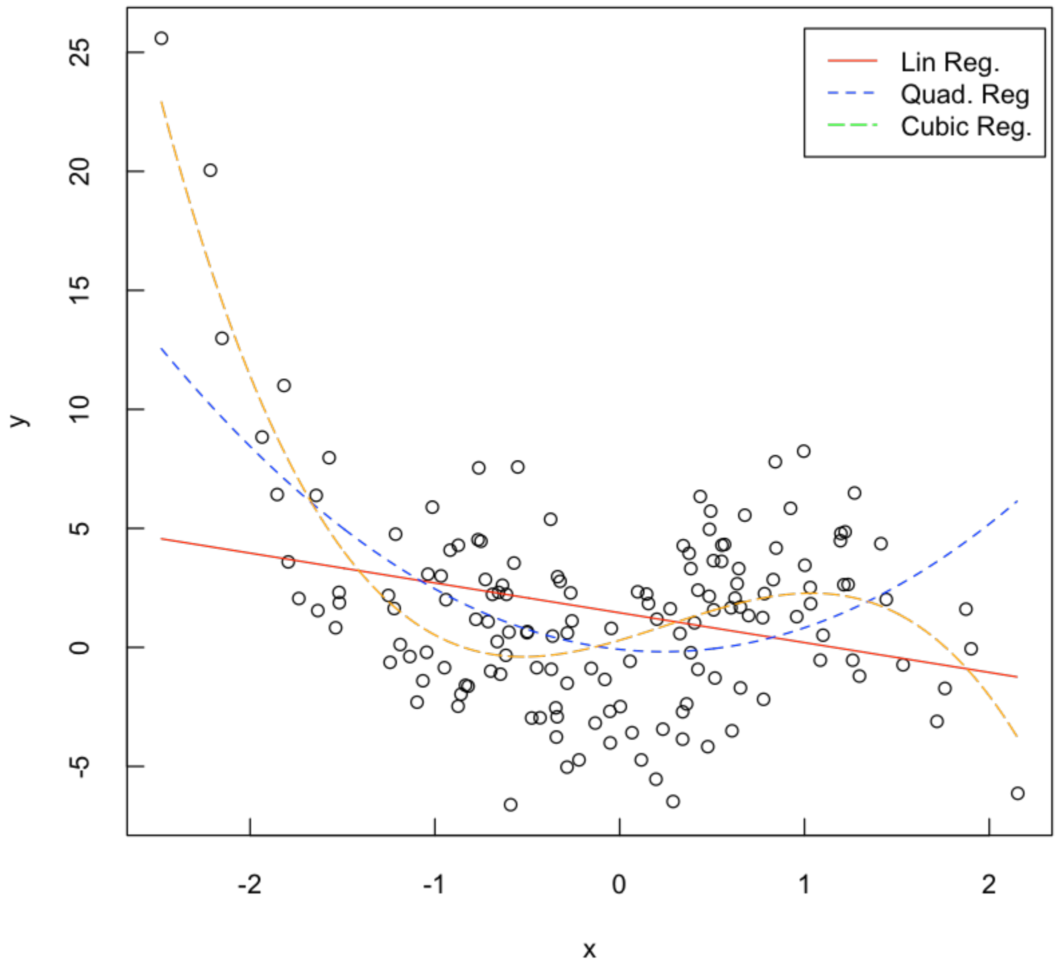
```r
beta0 <- betas[1]
beta1 <- betas[2]
beta2 <- betas[3]

xVals <- seq(minX,maxX,0.01)
quadraticYPredictions <- beta0 + beta1*xVals + beta2*xVals^2
lines(xVals,quadraticYPredictions,col='blue', lty=2)

#cubic regression
betas <- coef(cubicRegression)
beta0 <- betas[1]
beta1 <- betas[2]
beta2 <- betas[3]
beta3 <- betas[4]

xVals <- seq(minX,maxX,0.01)
cubicYPredictions <- beta0 + beta1*xVals + beta2*xVals^2 + beta3*xVals^3
lines(xVals,cubicYPredictions,col='orange',lty=5)


legend(1,26,legend=c("Lin Reg.","Quad. Reg", "Cubic Reg.")
                     ,col=c('red','blue','green')
                     ,lty=c(1,2,5)
)
```



What fit looks best?

0.3 Optimization and assessing error

We can look at the three model fits to the data and visually assess fit, but a quantitative method for evaluating model fit is likely more convincing. A quantitative fit allows us to numerically compare model fits.

The most common metrics for evaluating model fit involve: the **s**um **s**quares **e**rror (SSE), **s**um **s**quares **r**egression (SSR), and **s**um **s**quares **t**otal (SST).

Given a data set, the **SSE** sums the squared difference over all empirically collected data (y_i) and model predictions ($\hat{y}_i$)

$$\mathrm{SSE}(y, \hat{y}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2,$$

or in vector form,

$$\mathrm{SSE}(y, \hat{y}) = (y - \hat{y})'(y - \hat{y})$$

where

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{5}$$

and

$$y' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{bmatrix} \tag{6}$$

Note that all three of these relationships can be expressed using the same equation

$$y \sim N(X\beta, \sigma^2)$$

Why then is this called **linear** regression?

The **linear** in linear regression refers to the parameters. Lets look a the model form of a cubic regression.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \tag{7}$$
$$\epsilon \sim N(0, \sigma^2) \tag{8}$$

We can rewrite the above equation, so that it looks a bit more like multiple regression.

$$y = \beta_0 + \beta_1 x + \beta_2 q + \beta_3 r \tag{9}$$
$$q = x^2 \tag{10}$$
$$r = x^3 \tag{11}$$
$$\epsilon \sim N(0, \sigma^2) \tag{12}$$

Our equation now is linear in β and in reference to three variables: x, q, and r. We can do this with any functional form for x. The "linear" refers to the parameters.

```
[ ]:
```