simple_multiple_polynomial regression

September 5, 2019

1 Simple, multiple, and polynomial Linear Regression

Suppose you're asked by a physician whether or not aging, increased body mass index, and a history of smoking increase a patient's chances of a myocardial infarction (heart attack). An ecologist asks you how tree density (trees per square mile) and amount of fresh water are associated with the number of deer. A stock broker, looking to capitalize on their investment, asks you to relate historical stock price data to predict future prices.

All of these examples relate one random variable to a set of others. Questions like this can begin to be answered with **regression**

Regression determines the functional relationship between one random variable (denoted y) and a set of others (denoted X). Then we can frame our regression problem as finding a function that relates X and y to one another through a function f and list of parameters β, or

$$y = f(X|\beta).$$

While some more advanced methods do not specify what f should look like, the methods we'll explore in this class restrict the functional form between X and y and find optimal values for a small set of parameters that best explain our data.

In what follows we'll explore simple, multiple, and polynomial linear regression. Our goal is to understand how the following equation,

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

can represent all three regression models, how to fit and interpret these models, and how to evaluate how well these models fit our data.

1.1 Simple Linear regression

Suppose you've collected N pairs (x, y) and want to explore how changes in x impact changes in y. One plausible model relates x to y linearly. Our function

$$\begin{aligned} y &= f(x|\beta) &(1)\\ &= \beta_0 + x_i * \beta_1 &(2) \end{aligned}$$

or in matrix notation

$$y = X\beta$$

where

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

β is a vector containing β_0 and β_1,

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix},$$

and

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

We also assume that the above relationship only applies to the mean of y, and
since we've collected empirical data that may have inherent measurement error, or knowing our linear relationship may not perfectly fit our data, we assume our data is normally distributed with variance σ^2. Our model is now a **probabilstic** model, one that admits the data will not perfectly fit on a line, but will on average with a quantifiable error around this average.

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

or

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

A possible model in hand, we also need to assess how likely a linear relationship exists between x and y. If y and x were not related linearly, not related like

$$y = \beta_0 + \beta_1 x$$

then a small difference would exist between the above model and a simple average of the data, and we can represent the average by

$$y = \beta_0,$$

it does not depend on x.

Lets take a look at some sample data, and apply these concepts.

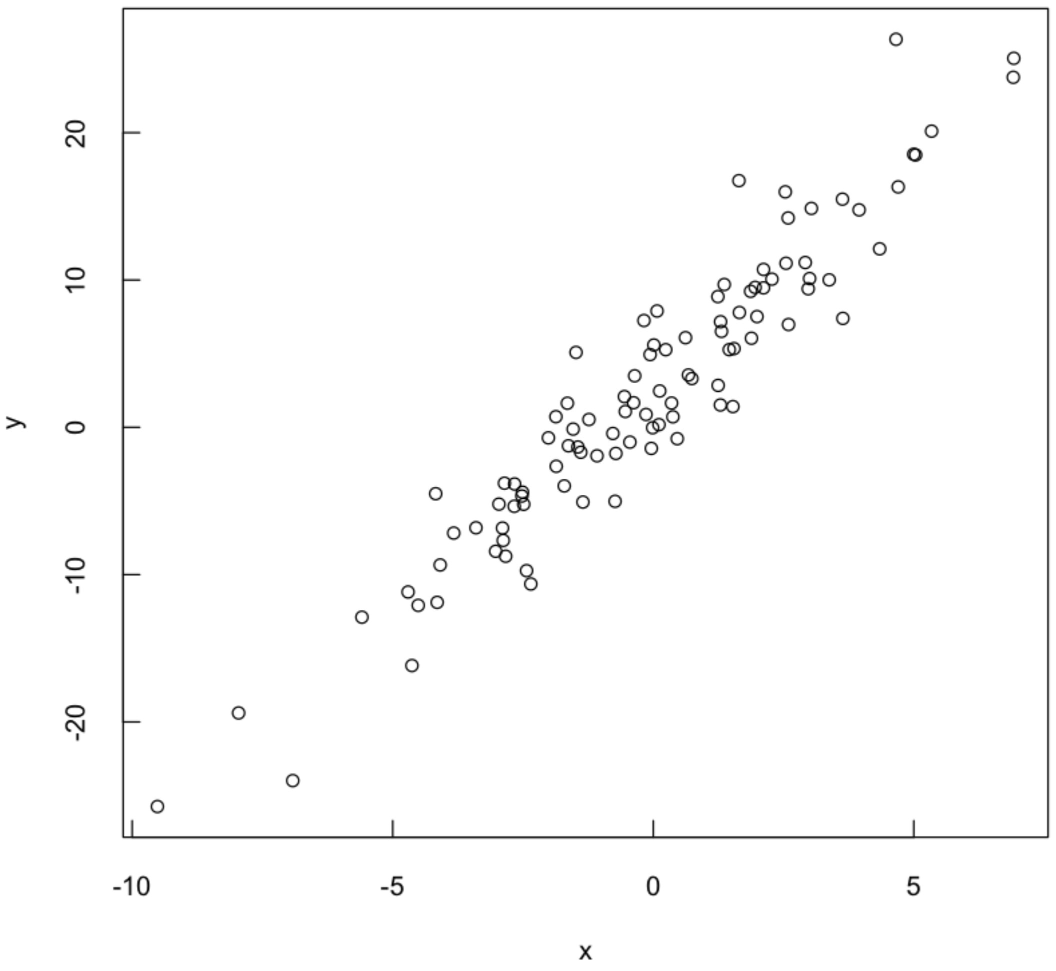
```
[68]: data<-read.csv('./ourSimpleLinReg.csv')
      head(data)
```

A data.frame: 6 × 2

x <dbl>	y <dbl>
6.9187782	25.049450
2.5956088	6.977310
-2.5106578	-4.403495
0.6182831	6.083902
-2.8554528	-3.794021
4.6587200	26.338585

We can gain some intuition about the relationship between x and y by plotting x against y

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



From the plot of x against y, it looks like x and y are related **linearly**, that the functional form should look something like $y = \beta_0 + \beta_1 x$. We can fit a linear model in R and plot this over the scatterplot.

[70]:
```
simpleLinearRegression <- lm(y~x,data=data)
print(simpleLinearRegression)
```

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

Coefficients:
(Intercept)            x
      2.614        3.086
```

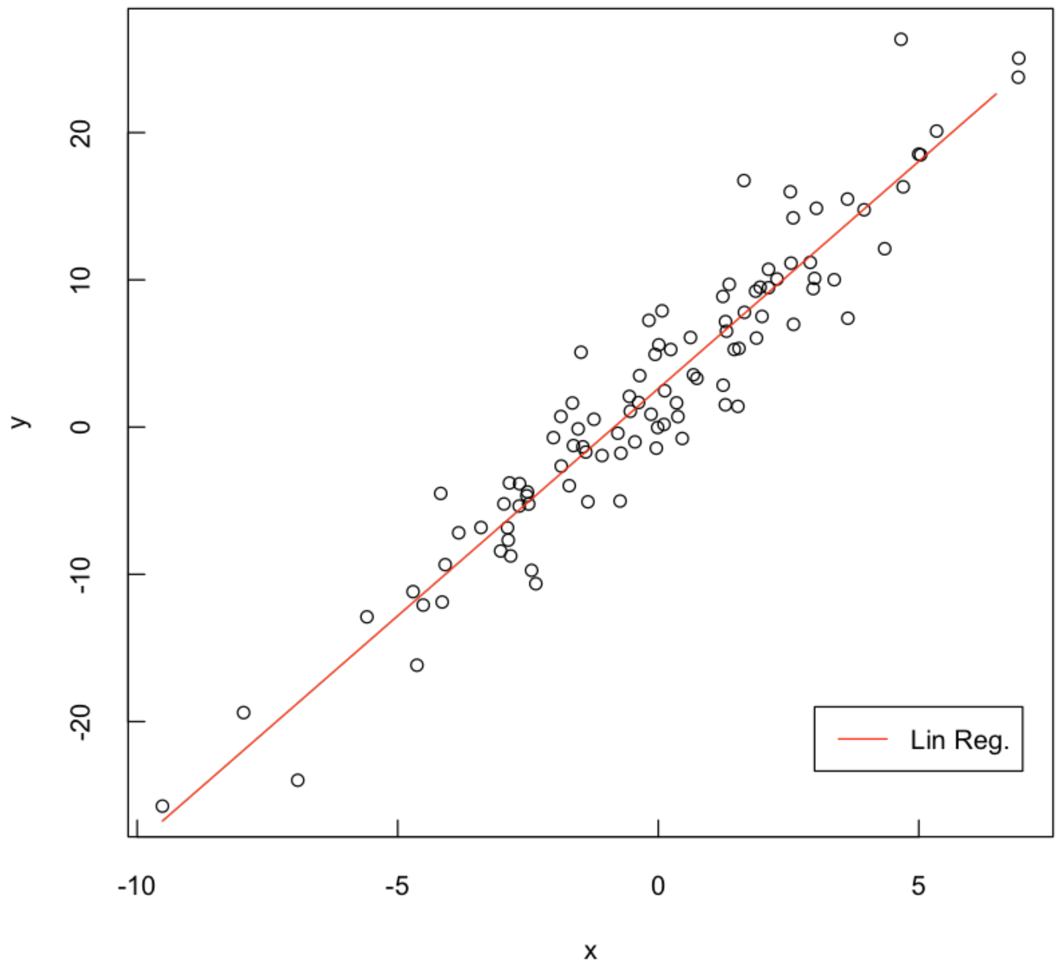
Assuming x is linearly related to y, $y = \beta_0 + \beta_1 x$, the best fit to the data is when $\beta_0 = 2.614$ and $\beta_1 = 3.086$. Let's evaluate this solution visually.

[71]:
```
plot(data$x,data$y
     ,xlab="x"
     ,ylab="y"
     ,tck=0.02
)
betas <- coef(simpleLinearRegression)
beta0 <- betas[1]
beta1 <- betas[2]

xVals <- seq(min(data$x),max(data$x),1)
yPredictions <- beta0 + beta1*xVals

lines(xVals,yPredictions,col='red')

legend(3,-19,legend="Lin Reg.",col='red',lty=1)
```



We can also compare this to a simple average over the model, a model that does not include x.

[72]:
```
plot(data$x,data$y
      ,xlab="x"
      ,ylab="y"
      ,tck=0.02
)

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

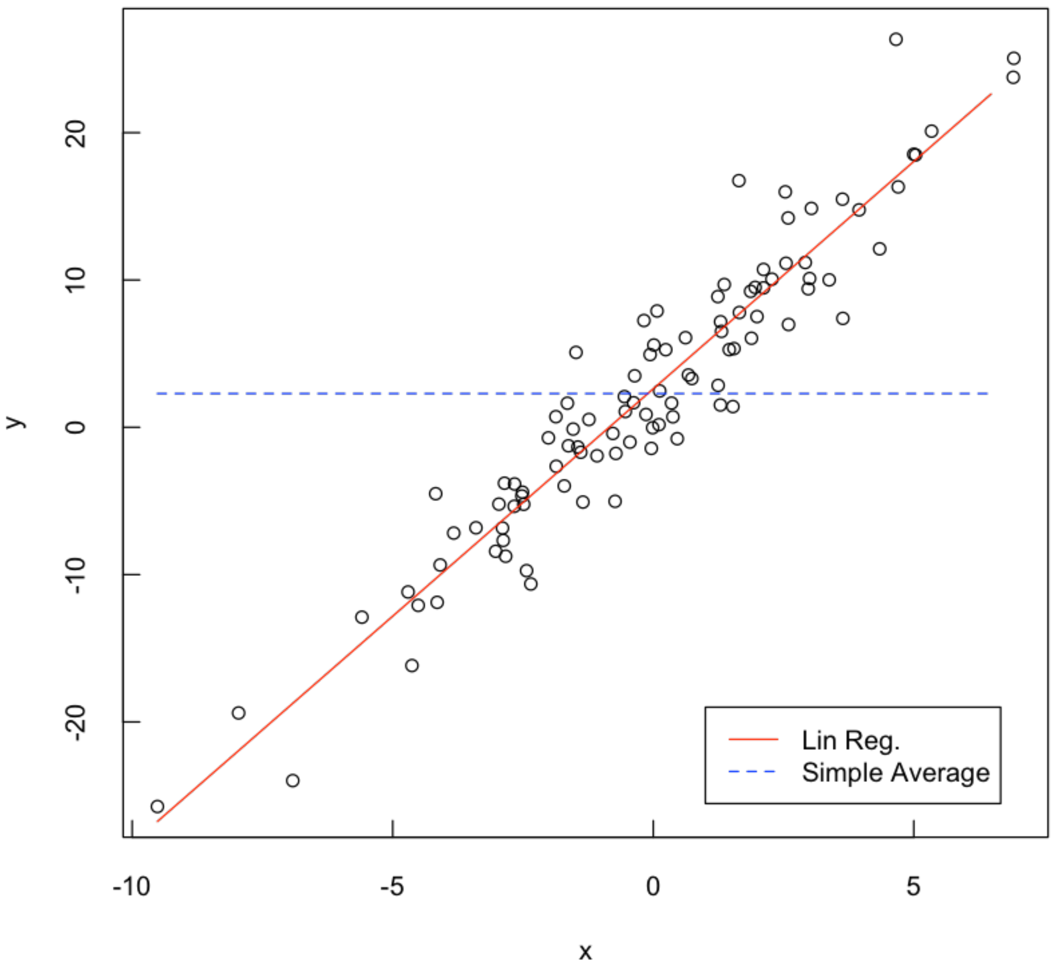
```r
xVals <- seq(min(data$x),max(data$x),1)
yPredictions <- beta0 + beta1*xVals

lines(xVals,yPredictions,col='red')


#simple average
simpleAverage <- mean(data$y)

lines(xVals, rep(simpleAverage,length(xVals)),col='blue',lty=2)

legend(1,-19
       ,legend=c("Lin Reg.","Simple Average")
       ,col=c('red','blue')
       ,lty=c(1,2)
)
```



Intuitively, we see that a simple average does a much worse job of explaining y. But lets rigorously test this.

We can assume that our parameters, β_0 and β_1, are themselves random variables. If we were able to gather a second, third, etc. set of data, these β parameters would likely change.

β_1, the slope, can be tested statistically with the following hypothesis test

$$H_{\text{null}} : \beta_1 = 0 \tag{5}$$
$$H_{\text{alternative}} : \beta_1 \neq 0 \tag{6}$$
$$\tag{7}$$

and computed in R

[80]:
```
summary(simpleLinearRegression)
confint(simpleLinearRegression)
```

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

Residuals:
    Min      1Q  Median      3Q     Max 
-8.8025 -2.8617 -0.0525  2.3649 21.0301 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1.4531     0.3379   4.300 3.09e-05 ***
x            -1.2529     0.3541  -3.538 0.000538 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.126 on 148 degrees of freedom
Multiple R-squared:  0.078,Adjusted R-squared:  0.07177 
F-statistic: 12.52 on 1 and 148 DF,  p-value: 0.0005382
```

A matrix: 2 × 2 of type dbl

	2.5 %	97.5 %
(Intercept)	0.785322	2.1209501
x	-1.952690	-0.5532032

Here we see two important pieces of information: the pvalue for the slope and the 95% confidence interval. The pvalue asks the following question "If our null hypothesis was correct (the slope equalled 0) what's the probability of seeing our estimated slope, or an even more extreme example?"

The 95% confidence interval estimates, over an infinite number of resampled finite datasets, where the slope would fall 95% of the time.

There is a correspondence between the 95% confidence interval and pvalue. A pvalue will be <0.05 if the 95% confidence interval does not include 0 (our null hypothesis value for the slope).

1.2 multiple regression

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

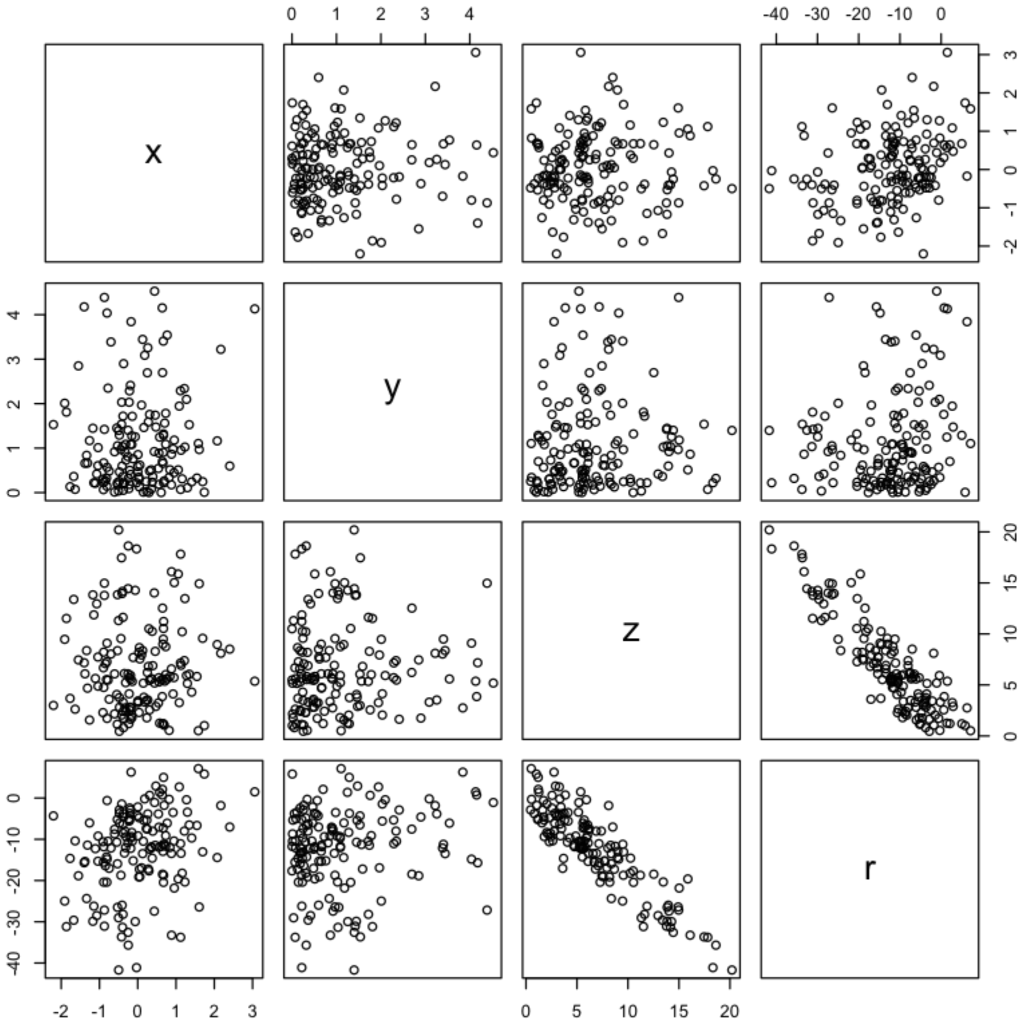
[74]:
```
dta <- read.csv('multipleData.csv')
head(dta)
```

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 in simple linear regression, relating one variable to one other variable, 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 **mutliple 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

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots \beta_n x_{in} + \epsilon_i \tag{8}$$
$$\epsilon \sim N(0, \sigma^2), \tag{9}$$

and using matrices and vectors,

$$y = X\beta + \epsilon \tag{10}$$
$$\epsilon \sim N(0, \sigma^2), \tag{11}$$

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 regresion in in probabilstic 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.

1.3 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

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

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.

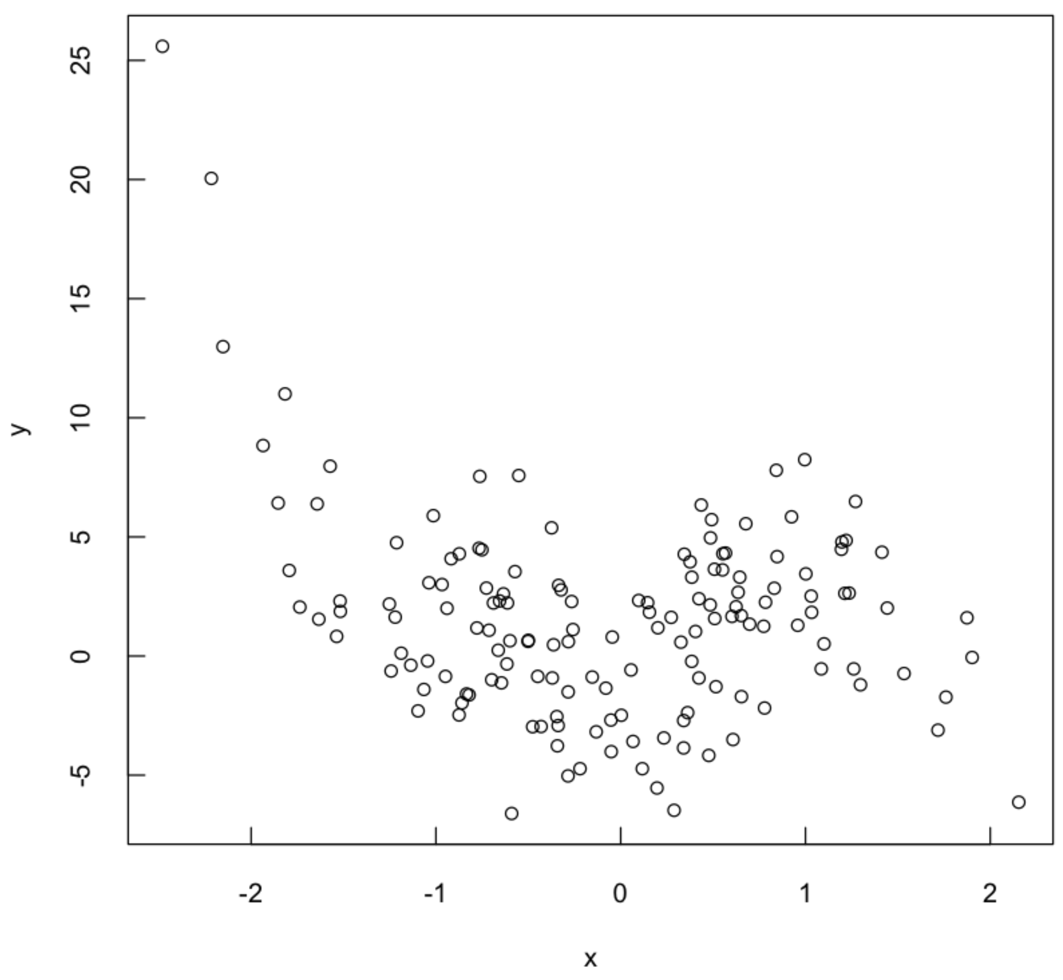
[76]:
```
data <- read.csv('polynomialData.csv')
head(data)
```

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.

[77]:
```
plot(data$x,data$y
     ,xlab="x"
     ,ylab="y"
     ,tck=0.02
)
```



Lets fit three different models: a linear model, a quadratic model, and a cubic model.

[78]:
```
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
```

[79]:
```
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)
beta0 <- betas[1]
beta1 <- betas[2]
```

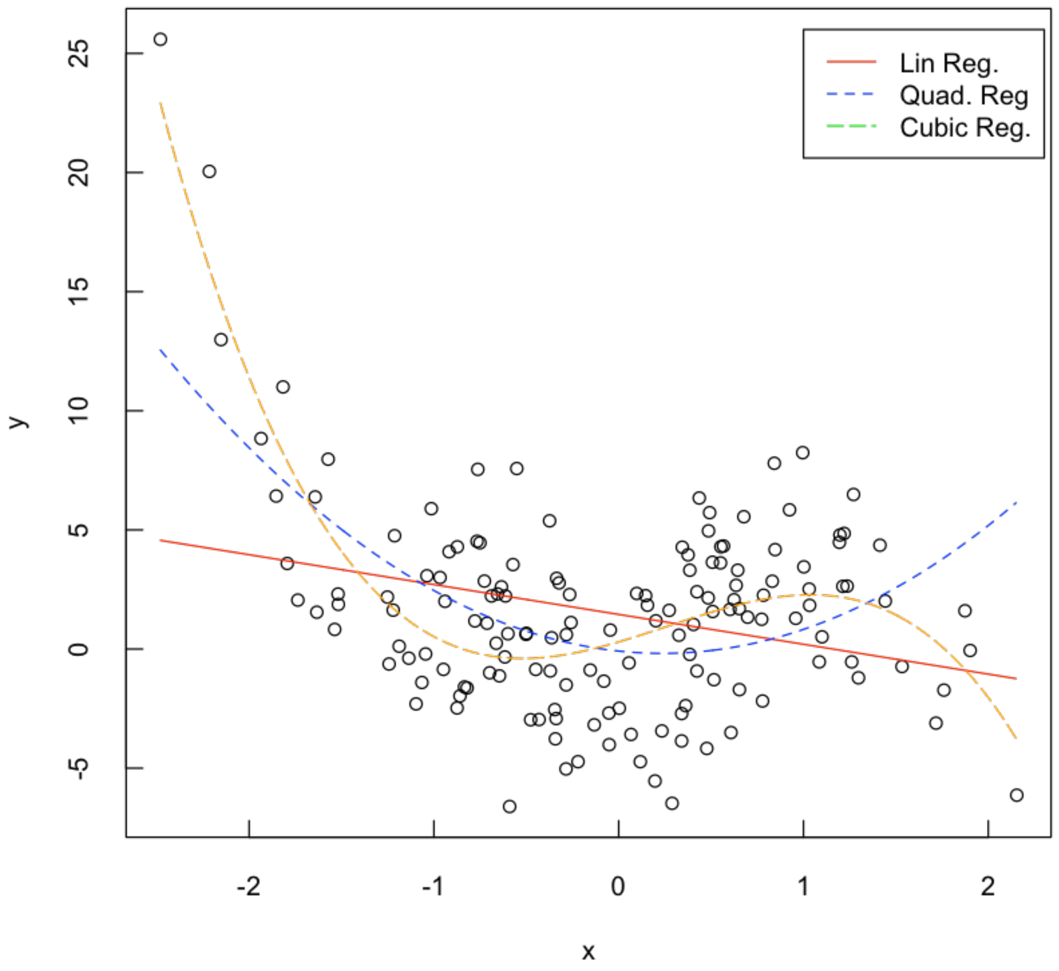
```
beta2 <- betas[3]

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)
)
```



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.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \tag{12}$$

$$\epsilon \sim N(0, \sigma^2) \tag{13}$$

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

$$y = \beta_0 + \beta_1 x + \beta_2 q + \beta_3 r \tag{14}$$
$$q = x^2 \tag{15}$$
$$r = x^3 \tag{16}$$
$$\epsilon \sim N(0, \sigma^2) \tag{17}$$

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 the parameters.

```
[ ]:
```