Non-linear least square regression

author: Maria Paniw date: 10.02.2016

What if the assumptions of linear regression or ANOVA are violated?

We can do two things:

Transform the data. You can use a logarithmic transformation to transfrom X-Y relationships that are curves into lines. You can transform the Y variable with `log(Y)` or the X variable `log(X)` or both. Of course, logging X is only an option in regression. The log transformation usually reduces the variation in the data if the mean and the variance are positively correlated.

Note though that a value for `log(0)` does not exist and you will get an error message of `Inf` as a response. One way around this problem is to add 1 to all observations you would like to log.

If you have **count data**, you can try the **square-root transformation**, where $Y^* = \sqrt{Y}$. Count data typically show a strong positive correlation between mean and variance and this transformation removes this correlation. Again, there is no $\sqrt{0}$, so add a small number (e.g., 0.5) to your observations.

For **proportions** (data bounded betwenn 0 and 1), you can use the **arcsine transformation** ($Y^* = arcsine\sqrt{Y}$).

Finally, the **Box-Cox transformation** is a generalized form of the other transformations. This transformation takes Y to the power of λ, where λ is chosen so that the resulting model has the best fit (compared with other values of λ) to the data: $Y^* = (Y^\lambda - 1)/\lambda$.

Your second option if you don't want to transform the data:

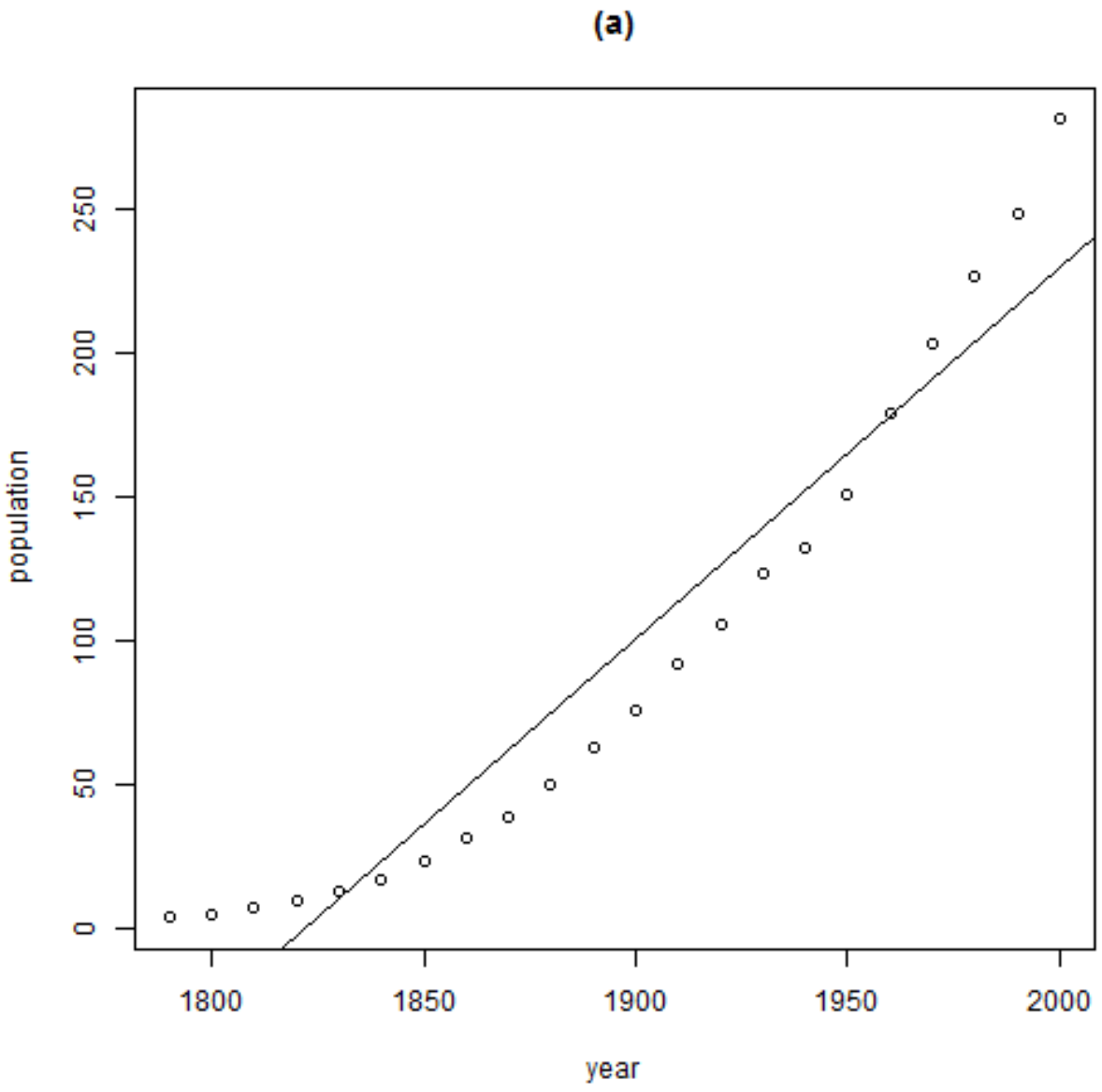
Run a nonlinear least squares regression. If you are working with a complex power (some form of $Y = aX^b$) or exponential (some form of $Y = e^{aX}$) function, a simple algebraic transformation may not be feasible. You can instead estimate the parameters of such functions directly using iterative methods. These methods, given initial values of parameters, iterate through numerical algorithms until they generate parameters that minimize the least squares. One function that does this in R is the `nls()` function.

As always, let's look at a concrete example

In order to run this example, we need to install the package `car` in R

```
#I comment the following command out because I already have the package
#install.packages("car")

library(car)
#This pacakge contains the data frame USPop that has decennial U. S. Census
#population for the United States (in millions), from 1790 through 2000.
```

```
plot(population ~ year, data=USPop, main="(a)")
abline(lm(population ~ year, data=USPop))
```

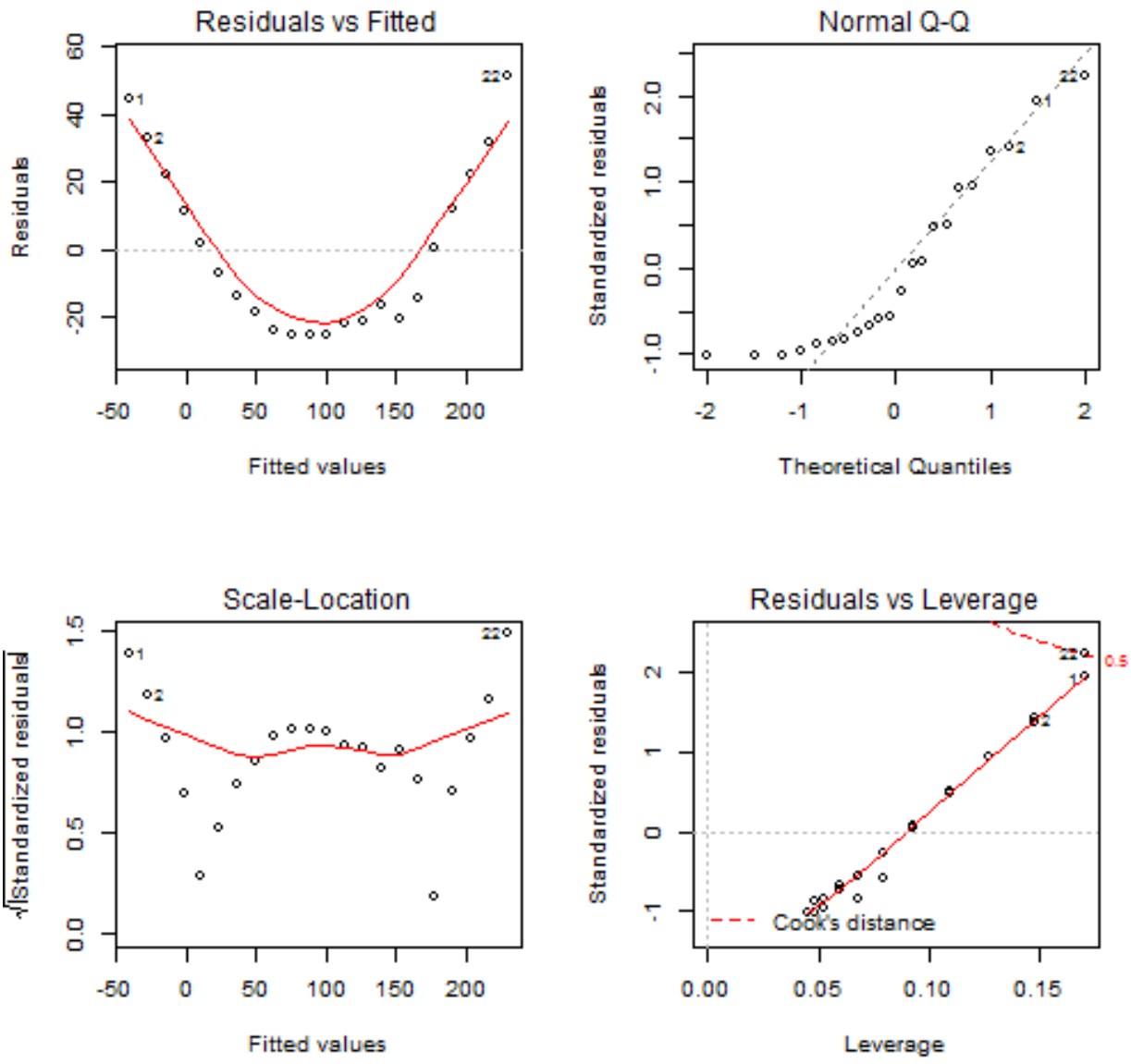


What can you tell about the data immediately?

Is the relationship between X and Y linear?

And then from looking at the `plot.lm`

```
#This allows you to split you screen and plot all 4 plots in one window
par(mfrow=c(2,2))

plot(lm(population ~ year, data=USPop))

#To go back to the single graph per window, use the following command
par(mfrow=c(1,1))
```



Yip, the residuals are clearly not normal, nor is the the variance homoscedastic.

So, let's try to tranform the data using the Box-Cox general transformation.

Box-Cox transformation

using the `powerTransform` function in the `car` package

```
p1 = powerTransform(population ~ year, data=USPop)
summary(p1)
```

```
bcPower Transformation to Normality
```

```
   Est.Power Std.Err. Wald Lower Bound Wald Upper Bound
Y1    0.3449   0.0184           0.3088           0.3811

Likelihood ratio tests about transformation parameters
                           LRT df         pval
LR test, lambda = (0) 65.14400  1 6.661338e-16
LR test, lambda = (1) 94.82891  1 0.000000e+00
```

The LR tests you see are likelihood ration tests that compare the λ found at the maximum likelihood estimate to hypothesized values (0, 1) - in our case, $\lambda = 0.334$ has a higher loglikelihood!

Fit linear model with transformed response

```
coef(p1, round=TRUE)
```

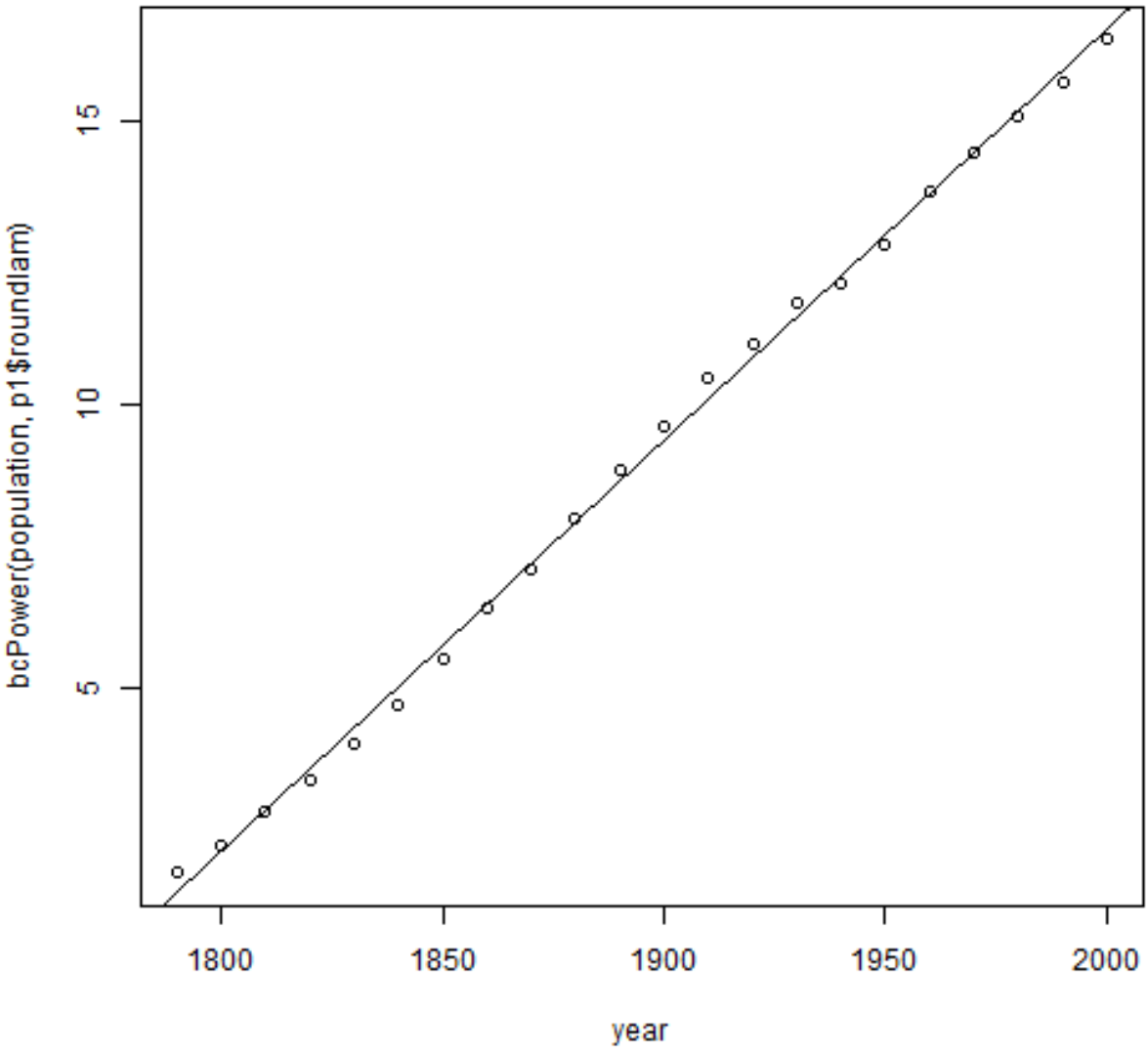
```
  Y1
0.33
```

```
m1 <- lm(bcPower(population, p1$roundlam) ~ year, data=USPop)
summary(m1)
```

```
Call:
lm(formula = bcPower(population, p1$roundlam) ~ year, data = USPop)

Residuals:
     Min       1Q   Median       3Q      Max
-0.32720 -0.15411 -0.05197  0.18519  0.36011

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.283e+02  1.370e+00  -93.64   <2e-16 ***
year         7.244e-02  7.225e-04  100.27   <2e-16 ***

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.215 on 20 degrees of freedom
Multiple R-squared:  0.998, Adjusted R-squared:  0.9979
F-statistic: 1.005e+04 on 1 and 20 DF,  p-value: < 2.2e-16
```

Again, let's check the results

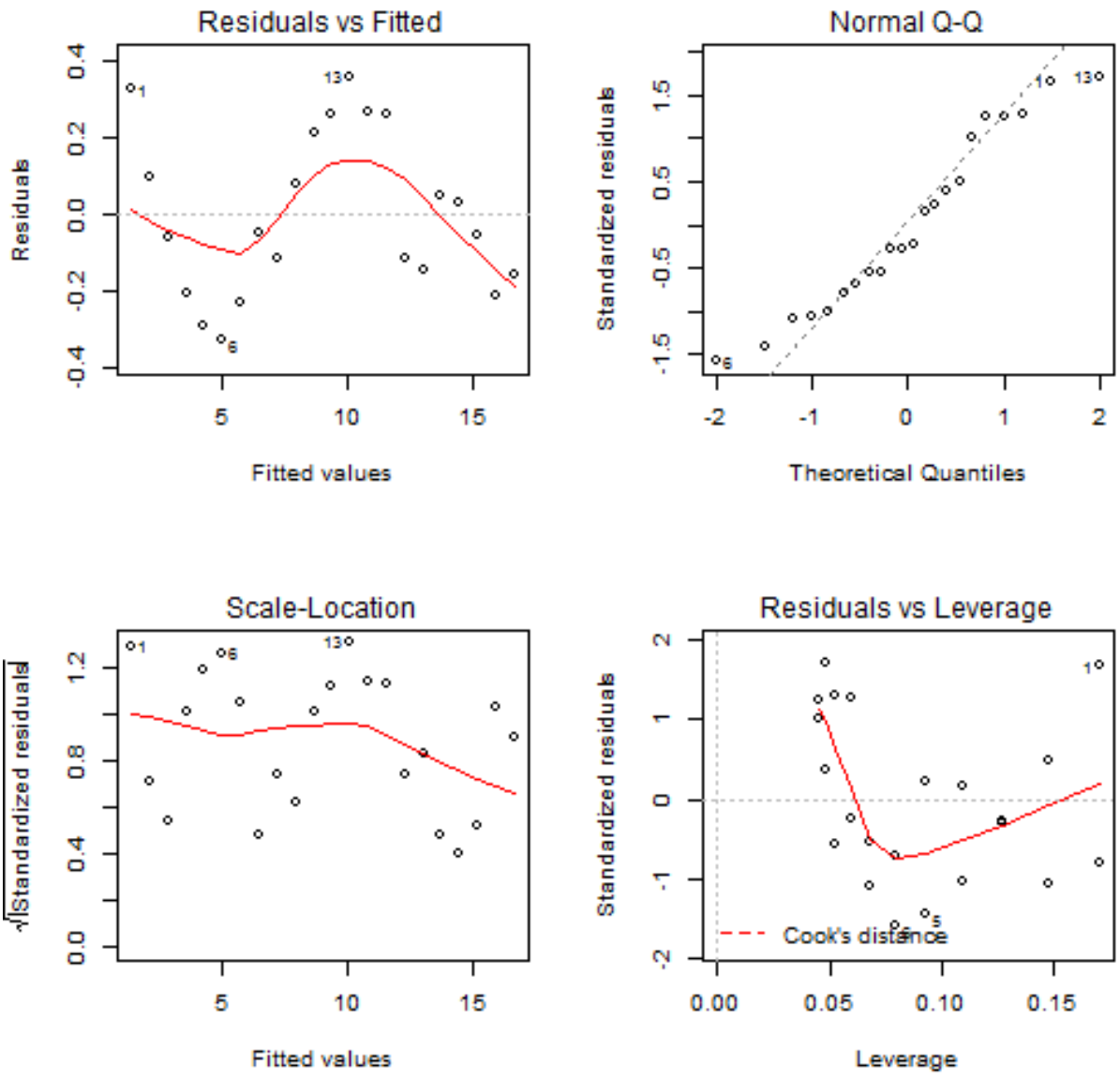
```
plot(bcPower(population, p1$roundlam) ~ year, data=USPop)
abline(m1)
```



Did we fix it?

Well, the plot with the transformed Y as a response looks normal. But be careful! Take a look at the residual plots! The residual variance clearly is not random and homoscedastic!

```
par(mfrow=c(2,2))

plot(m1)

#To go back to the single graph per window, use the following command
par(mfrow=c(1,1))
```



In fact, if we run some of our formal tests on the residuals, we may be able to pinpoint the problem:

```
#Normality
shapiro.test(as.numeric(residuals(m1))) # OK, normality is good to go
```

```
    Shapiro-Wilk normality test

data:  as.numeric(residuals(m1))
W = 0.9439, p-value = 0.2382
```

```
#Test for constant variance of residuals

ncvTest(m1) #also constant varaince
```

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 0.6720274    Df = 1     p = 0.4123457
```

```
# Test for Autocorrelated Errors
durbinWatsonTest(m1) # ATTENTION. Our residuals are autocorrelated
```

```
 lag Autocorrelation D-W Statistic p-value
   1       0.6887357     0.4800467       0
 Alternative hypothesis: rho != 0
```

An alternative: nls

A common simple model for population growth is the *logistic growth model*, expressed as

$$y = \frac{\theta_1}{1 + exp[-(\theta_2 + \theta_3 x)]} + \epsilon$$

Here is what an `nls` would look like:

```
mod.nls=nls(formula=population ~ theta1/(1 + exp(-(theta2 + theta3*year))),
             start=list(theta1=440, theta2=-49, theta3=0.025),
            algorithm = "default" ,data=USPop,trace=T)
```

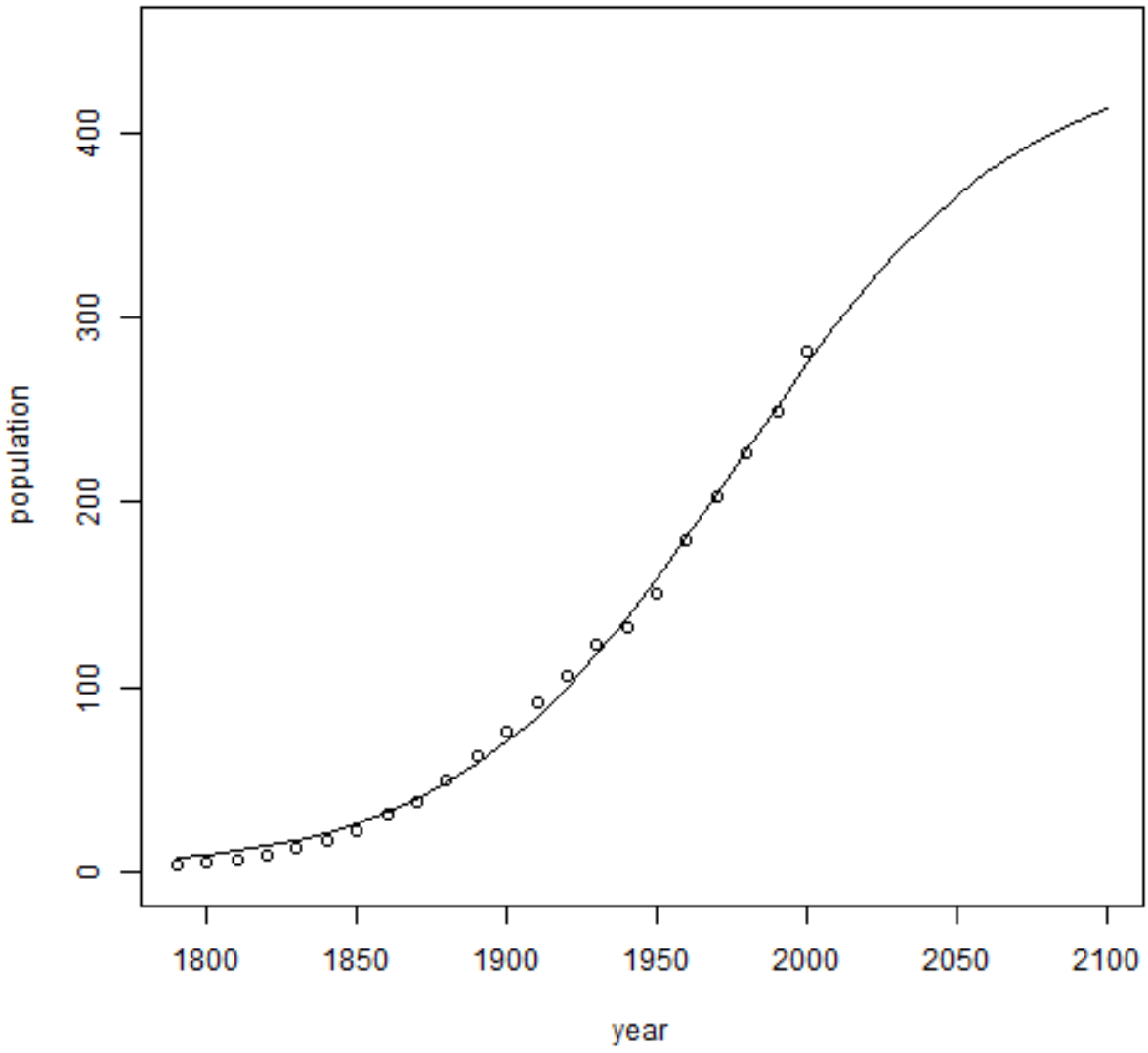
```
13446.15 :  440.000 -49.000   0.025
474.7592 :  426.06199153 -42.91623295   0.02174679
458.1232 :  438.7153636 -42.7747539   0.0216443
457.806 :  440.77631246 -42.70686103   0.02160596
457.8056 :  440.82549717 -42.70735543   0.02160612
457.8056 :  440.8334940 -42.7069540   0.0216059
```

The key here are the starting values! For the `nls` to converge (i.e., find the best-fit parameters), we need to specify values that the algorithms should start with - and these starting values must be reasonable!!! In the case of the logistic growth models, getting these starting values is not trivial, and this is perhaps the biggest drawback to the non-linear least-square regression. I got this example from this tutorial, and you can check how they derived their intial values.

Let's take a look at the predictions of population sizes

```
#get predicted values:
pred = predict(mod.nls, newdata=data.frame(year=seq(1790, 2100, by=10)))
```

```
plot(population ~ year, USPop, xlim=c(1790, 2100), ylim=c(0,450))

lines(seq(1790, 2100, by=10),pred)
```


population
0
100
200
300
400
1800
1850
1900
1950
2000
2050
2100
year