Homework 05

Anika Cartas

Wednesday, April 08, 2015

### 0. Read Data

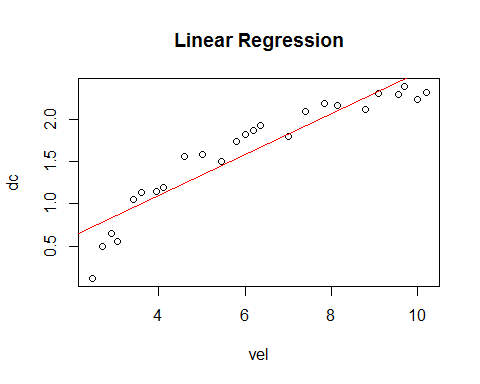
The table lists measurements of DC output at different wind velocity.

* Dependent variable: DC
* Independent variable: Wind Velocity

data <- data.frame(  
 vel = c(5,6,3.4,2.7,10,9.7,9.55,3.05,8.15,6.2,2.9,6.35,4.6,5.8,7.4,3.6,7.85,8.8,7,5.45,9.1,10.2,4.1,3.95,2.45),  
 dc = c(1.582,1.822,1.057,0.5,2.236,2.386,2.294,0.558,2.166,1.866,0.653,1.93,1.562,1.737,2.088,1.137,2.179,2.112,1.8,1.501,2.303,2.310,1.194,1.144,0.123))  
  
data <- data[with(data,order(vel)),]  
  
attach(data)

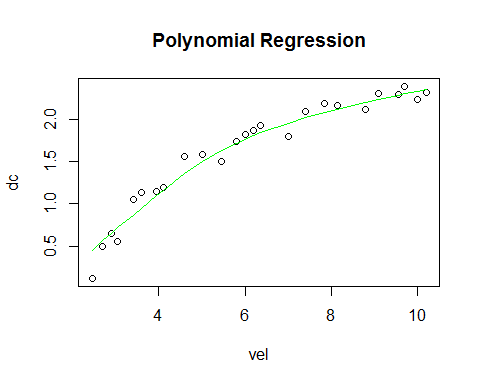
### 1. Linear Regression

reg.linear <- lm(dc~vel)  
  
plot(dc~vel, main="Linear Regression")  
abline(reg.linear, col="red")



### 2. Polynomial Regression

plot(dc~vel, main="Polynomial Regression")  
lines(lowess(dc~vel), col="green")



Using lowess, we visualize this to be a 4th degree polynomial. However, let's see if we can get more details.

fit2 <- lm(dc ~ poly(vel, 2, raw=TRUE))  
fit3 <- lm(dc ~ poly(vel, 3, raw=TRUE))  
fit4 <- lm(dc ~ poly(vel, 4, raw=TRUE))  
fit5 <- lm(dc ~ poly(vel, 5, raw=TRUE))  
summary(fit2)

##   
## Call:  
## lm(formula = dc ~ poly(vel, 2, raw = TRUE))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.26347 -0.02537 0.01264 0.03908 0.19903   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.155898 0.174650 -6.618 1.18e-06 \*\*\*  
## poly(vel, 2, raw = TRUE)1 0.722936 0.061425 11.769 5.77e-11 \*\*\*  
## poly(vel, 2, raw = TRUE)2 -0.038121 0.004797 -7.947 6.59e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1227 on 22 degrees of freedom  
## Multiple R-squared: 0.9676, Adjusted R-squared: 0.9646   
## F-statistic: 328.3 on 2 and 22 DF, p-value: < 2.2e-16

summary(fit3)

##   
## Call:  
## lm(formula = dc ~ poly(vel, 3, raw = TRUE))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.199016 -0.054316 0.008116 0.064812 0.164210   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.358550 0.438464 -5.379 2.46e-05 \*\*\*  
## poly(vel, 3, raw = TRUE)1 1.426866 0.246758 5.782 9.70e-06 \*\*\*  
## poly(vel, 3, raw = TRUE)2 -0.160366 0.042056 -3.813 0.00101 \*\*   
## poly(vel, 3, raw = TRUE)3 0.006458 0.002211 2.921 0.00817 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1059 on 21 degrees of freedom  
## Multiple R-squared: 0.9769, Adjusted R-squared: 0.9737   
## F-statistic: 296.6 on 3 and 21 DF, p-value: < 2.2e-16

summary(fit4)

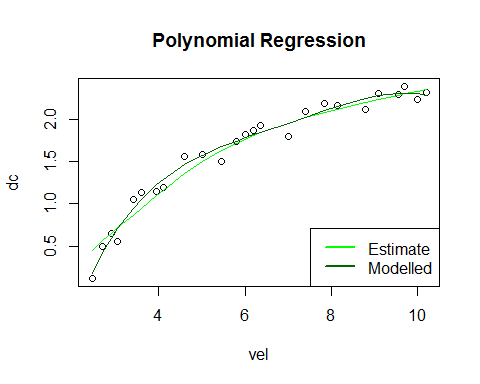
##   
## Call:  
## lm(formula = dc ~ poly(vel, 4, raw = TRUE))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.17091 -0.06373 0.02672 0.07621 0.11434   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -4.7151534 1.0997243 -4.288 0.000359 \*\*\*  
## poly(vel, 4, raw = TRUE)1 3.2773039 0.8353441 3.923 0.000842 \*\*\*  
## poly(vel, 4, raw = TRUE)2 -0.6620619 0.2214626 -2.989 0.007245 \*\*   
## poly(vel, 4, raw = TRUE)3 0.0627544 0.0245587 2.555 0.018861 \*   
## poly(vel, 4, raw = TRUE)4 -0.0022306 0.0009698 -2.300 0.032342 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.09648 on 20 degrees of freedom  
## Multiple R-squared: 0.9818, Adjusted R-squared: 0.9781   
## F-statistic: 269.2 on 4 and 20 DF, p-value: < 2.2e-16

summary(fit5)

##   
## Call:  
## lm(formula = dc ~ poly(vel, 5, raw = TRUE))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.16744 -0.06573 0.02651 0.06511 0.11238   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -6.4743544 3.1589991 -2.049 0.0545 .  
## poly(vel, 5, raw = TRUE)1 5.0279731 3.0604169 1.643 0.1168   
## poly(vel, 5, raw = TRUE)2 -1.3154225 1.1201753 -1.174 0.2548   
## poly(vel, 5, raw = TRUE)3 0.1775419 0.1943954 0.913 0.3725   
## poly(vel, 5, raw = TRUE)4 -0.0117847 0.0160763 -0.733 0.4725   
## poly(vel, 5, raw = TRUE)5 0.0003034 0.0005096 0.595 0.5586   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.09808 on 19 degrees of freedom  
## Multiple R-squared: 0.9821, Adjusted R-squared: 0.9774   
## F-statistic: 208.5 on 5 and 19 DF, p-value: 6.523e-16

Depending on how specific we want to get, it looks like we are adding significant variables up to 4th degree. I am using the significance levels of each added degree of the variable to determine my cut-off point, because due to overfitting errors the R^2 and RMSE value are not good indicators of which model is best.

plot(dc~vel, main="Polynomial Regression")  
lines(lowess(dc~vel), col="green")  
reg.poly <- lm(dc ~ poly(vel, 4, raw=TRUE))  
pred.poly <- predict(reg.poly)  
lines(vel, pred.poly,col="darkgreen")   
legend("bottomright", c("Estimate","Modelled"), col=c("green","darkgreen"), lwd=2)



### 3. Transformation Based Regression

The Box-Cox Power Transformation identifies an appropriate exponent (lambda) to transform the data into a normal shape. The Lambda value indicates the power to which all data should be raised.

When lambda is plotted against log likelihood, the best lambda is the one that maximizes the log likelihood.

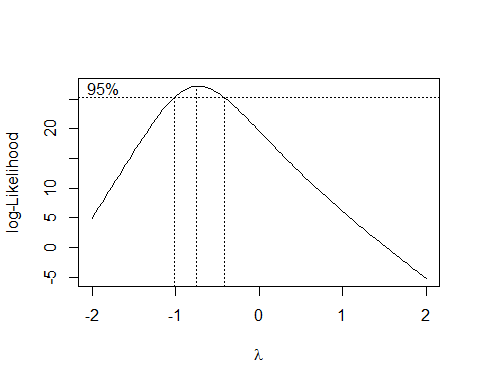
I am doing this two ways. One in which I use the boxcox method to determine a transformation for the dependent variable, and one in which I transform the independent variable. I will compare their results at the end in the discussion section.

#### 3.1. Transformation of Independent Variable

library(MASS)

## Warning: package 'MASS' was built under R version 3.1.3

bc.ind <-boxcox(vel ~ dc)



(maxll <- max(bc.ind$y))

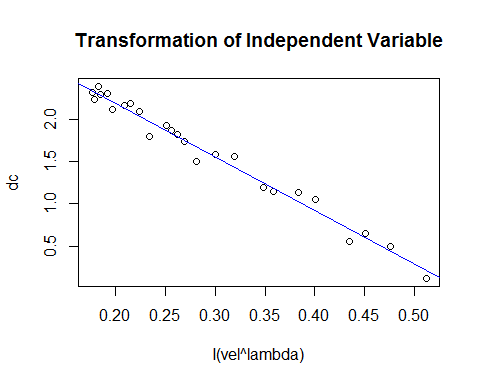
## [1] 27.17658

(lambda <- bc.ind$x[which(bc.ind$y == maxll)])

## [1] -0.7474747

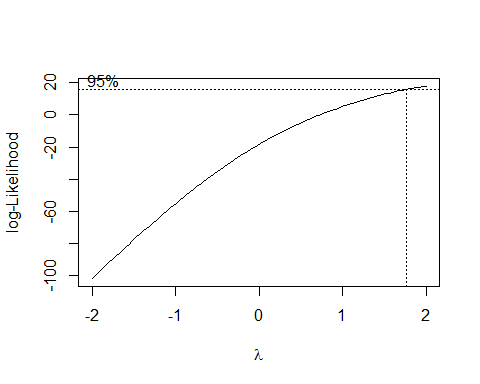
Maximum log likelihood of 27.17 is achieved when lambda is equal to -0.75.

reg.ind<-lm(dc ~ I(vel^lambda))  
plot(dc ~ I(vel^lambda), main="Transformation of Independent Variable")  
abline(reg.ind, col="blue")



#### 3.2. Transformation of Dependent Variable

bc.dep <- boxcox(dc ~ vel)



(maxll.dep <- max(bc.dep$y))

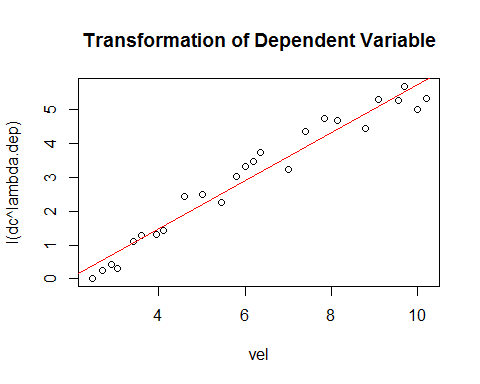
## [1] 17.6537

(lambda.dep <- bc.dep$x[which(bc.dep$y == maxll.dep)])

## [1] 2

Maximum log likelihood of 17.65 is achieved when lambda is equal to 2.

reg.dep<-lm(I(dc^lambda.dep) ~ vel)  
plot(I(dc^lambda.dep) ~ vel, main="Transformation of Dependent Variable")  
abline(reg.dep, col="red")

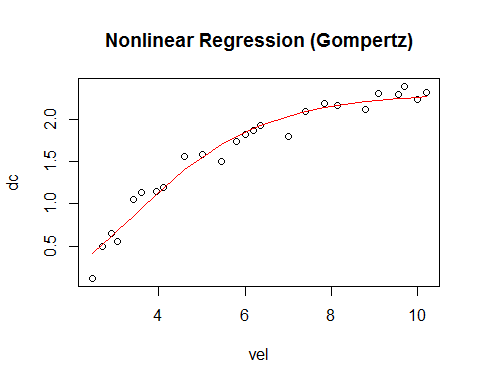


### 4. Nonlinear Regression

Following advice from Simran, I will investigate the Gompertz Growth Model.

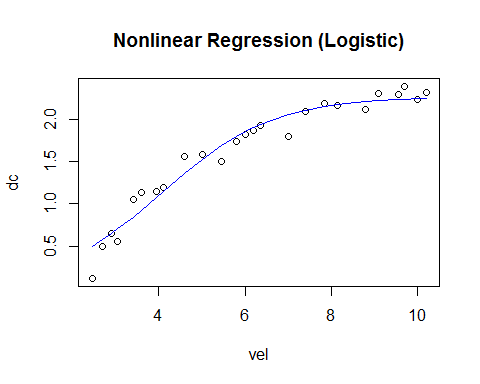
The nls model will allow me to find good values for the parameters. One important thing to note that I was unaware of is that I do not need to supply values for the parameters in the SS... functions, just variable names.

reg.gomp <- nls(dc ~ SSgompertz(vel, Asym, b2, b3))  
pred.gomp <- predict(reg.gomp)  
  
plot(dc ~ vel, main="Nonlinear Regression (Gompertz)")  
lines(vel,pred.gomp,col="red")



I will also try the Logistic Model.

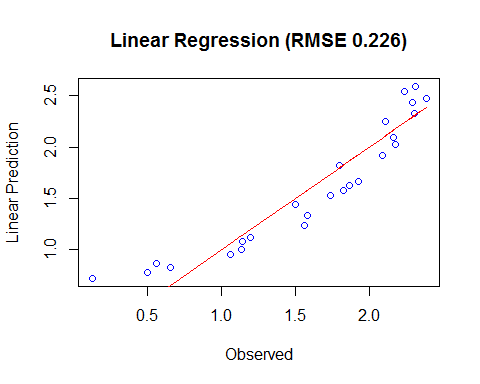
reg.log <- nls(dc ~ SSlogis(vel, Asym, xmid, scal))  
pred.log <- predict(reg.log)  
  
plot(dc ~ vel, main="Nonlinear Regression (Logistic)")  
lines(vel,pred.log,col="blue")



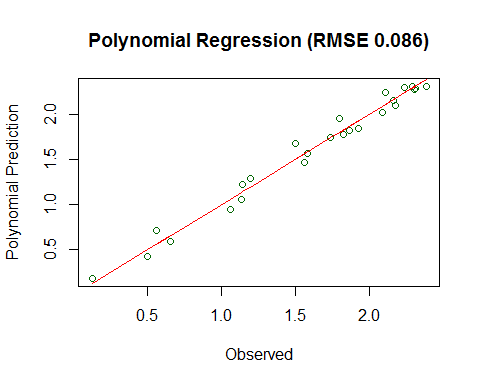
### 5. Discussion

To determine which model is best, I will calculate the RMSE of observed vs predicted values.

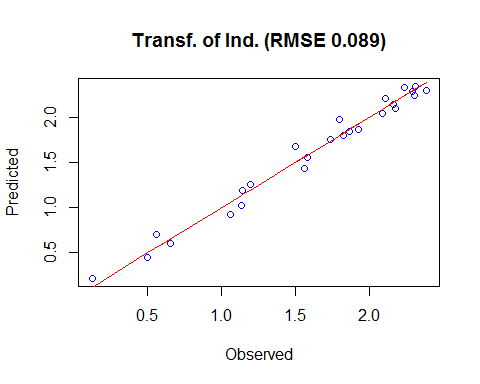
pred.linear <- predict(reg.linear)  
rmse.linear <- sqrt(mean((dc-pred.linear)^2))  
plot(pred.linear~dc, col="blue",main=sprintf("Linear Regression (RMSE %.03f)",rmse.linear),xlab="Observed",ylab="Linear Prediction")  
lines(dc, dc, col="red")



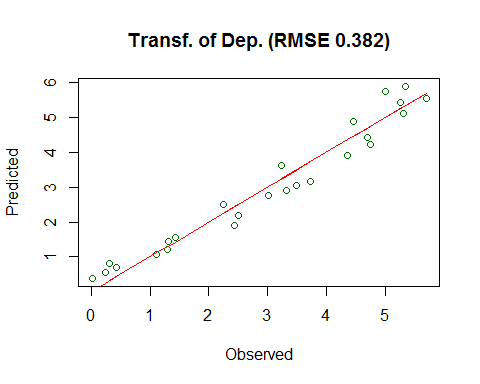
rmse.poly <- sqrt(mean((dc-pred.poly)^2))  
plot(pred.poly~dc, col="darkgreen",main=sprintf("Polynomial Regression (RMSE %.03f)",rmse.poly),xlab="Observed",ylab="Polynomial Prediction")  
lines(dc, dc, col="red")



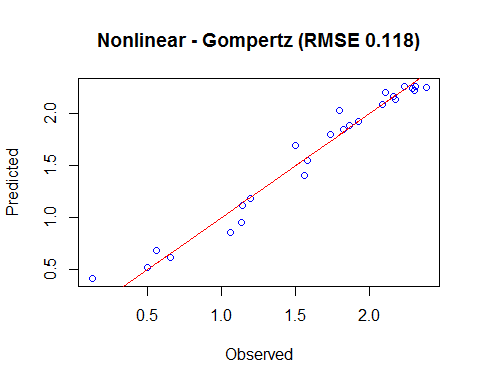
pred.ind <- predict(reg.ind)  
rmse.ind <- sqrt(mean((dc-pred.ind)^2))  
plot(pred.ind ~ dc, col="blue",main=sprintf("Transf. of Ind. (RMSE %.03f)",rmse.ind),xlab="Observed",ylab="Predicted")  
lines(dc,dc,col="red")



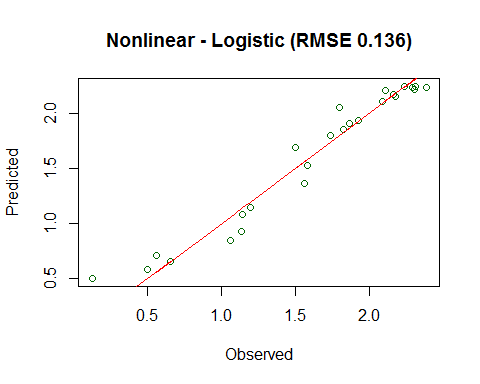
pred.dep <- predict(reg.dep)  
rmse.dep <- sqrt(mean((I(dc^lambda.dep)-pred.dep)^2))  
plot(pred.dep ~ I(dc^lambda.dep), col="darkgreen",main=sprintf("Transf. of Dep. (RMSE %.03f)",rmse.dep),xlab="Observed",ylab="Predicted")  
lines(I(dc^lambda.dep),I(dc^lambda.dep),col="red")



rmse.gomp <- sqrt(mean((dc-pred.gomp)^2))  
plot(pred.gomp ~ dc, col="blue",main=sprintf("Nonlinear - Gompertz (RMSE %.03f)",rmse.gomp),xlab="Observed",ylab="Predicted")  
lines(dc,dc,col="red")



rmse.log <- sqrt(mean((dc-pred.log)^2))  
plot(pred.log ~ dc, col="darkgreen",main=sprintf("Nonlinear - Logistic (RMSE %.03f)",rmse.log),xlab="Observed",ylab="Predicted")  
lines(dc,dc,col="red")



(sort(c("Linear"=rmse.linear, "Polynomial"=rmse.poly,"Transf. DC"=rmse.dep, "Transf. Vel"=rmse.ind, "Gompertz"= rmse.gomp, "Logistic"=rmse.log)))

## Polynomial Transf. Vel Gompertz Logistic Linear Transf. DC   
## 0.08629720 0.08881772 0.11796146 0.13600088 0.22641318 0.38168578

Using RMSE as our analysis point, we could determine that the polynomial regression is our best model. However, since polynomial regressions may have a problem of overfitting, we should be wary of using this model to infer anything about unknown data, especially since we know the 4th degree polynomial will dip back down instead of continuing to increase, which is what we would expect given our knowledge about our dataset.

Since the transformation of the Independent Variable is pretty close, I would choose this model.