R Companion Notes for APS Workshop on Regression

Paul Esker
University of Wisconsin-Madison

Larry Madden
The Ohio State University

These notes are provided as a companion guide to conduct regression analyses using the R statistical package (http://www.r-project.org). As written on the R website, R is a language and environment for statistical computing and graphics. R is free software available under the Free Software Foundation's GNU General Public License.

In these notes, different methods of data entry will be illustrated, but not discussed. We recommend that you consult the series of excellent R references regarding how to get started with R. Below, we highlight some documents that are helpful for regression and also topics in plant disease epidemiology. These goal of these notes is to focus on the different regression examples discussed during the workshop. The primary examples will be highlighted and discussed.

Some useful references for a general introduction to R and for regression analyses:

1) From http//www.r-project.org
    a. An Introduction to R: Select "Manuals" link
2) Practical Regression and Anova using R, Author: Julian Faraway, Select "Contributed" link on the R website
3) R Functions for Regression Analysis, Author: Vito Ricci, Select "Contributed" link on the R website
4) http://www.apsnet.org/education/AdvancedPlantPath/Topics/Top.html, a series of R-related documents for Ecology and Epidemiology in R (including regression)
    a. Also includes an introduction to R that can be useful for learning how to work with data, enter data manually or from file, etc.

Note: Code will be illustrated using a PC; slight differences for Macs and Linux operating systems. Also, there may be some output in the text, but not always and code may not cover all examples exactly as shown during the workshop, rather, this code highlights the main examples and functions.

Note2: R **IS** case-sensitive…

Note3: If, at any time, you are not sure how a function performs, you can "ask" R for help, using the following different forms:

> ?"argument"
> help.search("argument")

Note4: The notes are presented as copied from R – do not directly copy and paste into R as you will get an error due to the ">" symbol. However, these follow common nomenclature for R reference materials.

Regression1.

Relationship between wheat yield-loss and Fusarium head blight intensity
at Feekes' growth stage 11.2. Data from tables in Mesterhazy et al. Plant Dis. 87: 1107-1115.
Data reported as percentage losses. For this exercise, we converted to yield-loss in MT/ha,
assuming a fixed yield for the disease-free check. (This conversion is an arbitrary decision for
teaching purposes).

loss: response variable (estimated yield loss in MT/ha)
FHB: Fusarium head blight intensity
FDK: percentage of harvested wheat kernels with Fusarium symptoms

Libraries/Functions/Arguments Illustrated:
"plot", "lm", "names", "summary", "anova", "influence.measures", "MASS", "studres", "abline", "as.data.frame", "predict", "options", "se.fit", "data.frame", "interval"

Example Code (Libraries, Functions or Arguments are in bold the first time they are used):

```
# Data entry via "Manual" entry in R
> FHB=c(23.8, 23.9, 24.8, 28.5, 30.8, 32.4, 36.5, 38.2, 38.2, 51.8)
> Loss=c(1.440, 1.525, 1.535, 1.675, 1.775, 1.830, 2.215, 2.250, 2.165, 2.935)
> FDK=c(14.6, 15.9, 19.6, 18.4, 19.3, 27.1, 29.7, 33.1, 32.0, 43.7)

#Plot of the raw data
> plot(FHB,Loss,xlab='FHB field severity ("index")', ylab='Wheat yield loss (MT/ha)', main='Regression
Workshop 2009, Example 1',xlim=c(20,60), ylim=c(1,3.5), pch=19, col='blue')

#To run a simple linear regression, we use the "lm" function (also #used for ANOVA
> reg1=lm(Loss~FHB)
#The names argument shows all of the possible output components
> names(reg1)
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values" "assign"
"qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model"
#For example, to obtain the estimates of the coefficients…
> reg1$coefficients
(Intercept)         FHB
 0.20416916  0.05260963

#To look at a summary of the output, use "summary" and/or "anova"
> summary(reg1) #Provides measures of residuals, coefficients, R2
> anova(reg1) #Provides an ANOVA table only

#To examine the fit of the model, we can use the "plot" function and the "which" function #to look at
selective graphs
> plot(reg1, which=1) #Examines the residuals against the fitted values
> plot(reg1, which=2) #Normal QQ plot
> plot(reg1, which=4) #Cook's distance plot
> plot(reg1, which=5) #Standardized residuals versus leverage plot

#To obtain measures of influence, we can use "influence.measures" – these include the DFBETAS for #each
model variable, DFFITS, covariance ratios, Cook's distances and the diagonal elements of #the hat
matrix. Cases which are influential with respect to any of these measures are marked #with an asterisk

> influence.measures(reg1)

#To look at the studentized residuals, we need to call the "MASS" library as:
> library(MASS)
> studreg1=studres(reg1)
> studreg1
 1          2          3          4          5          6          7          8          9         10
-0.3025577  1.3043032  0.4823653 -0.5071842 -0.9022365 -1.5732077  1.9552156  0.6546990 -0.9083145  0.1508218

#We can then plot the studentized residuals against the fitted values as:
> plot(reg1$fitted.values,studreg1, ylim=c(-3,3), xlim=c(1,3))
> abline(h=0)
```

```
#To look at the fitted values and their corresponding standard errors, we will create a data #frame of
our original data, and use the “predict” function
> reg1B=as.data.frame(cbind(FHB, Loss, FDK))
> reg1B
    FHB  Loss  FDK
1  23.8 1.440 14.6
2  23.9 1.525 15.9
3  24.8 1.535 19.6
4  28.5 1.675 18.4
5  30.8 1.775 19.3
6  32.4 1.830 27.1
7  36.5 2.215 29.7
8  38.2 2.250 33.1
9  38.2 2.165 32.0
10 51.8 2.935 43.7
> options(digits=4) #Rounding to four digits
> fit_se_reg1=predict(reg1,se.fit=TRUE)
> data.frame(reg1B,residual=resid(reg1),fit_se_reg1$fit,fit_se_reg1$se.fit)
    FHB  Loss  FDK  residual fit_se_reg1.fit fit_se_reg1.se.fit
1  23.8 1.440 14.6 -0.016278           1.456            0.02702
2  23.9 1.525 15.9  0.063461           1.462            0.02686
3  24.8 1.535 19.6  0.026112           1.509            0.02544
4  28.5 1.675 18.4 -0.028544           1.704            0.02057
5  30.8 1.775 19.3 -0.049546           1.825            0.01873
6  32.4 1.830 27.1 -0.078721           1.909            0.01819
7  36.5 2.215 29.7  0.090579           2.124            0.01982
8  38.2 2.250 33.1  0.036143           2.214            0.02160
9  38.2 2.165 32.0 -0.048857           2.214            0.02160
10 51.8 2.935 43.7  0.005652           2.929            0.04543

#Finally, to see the corresponding prediction intervals
> predict(reg1,interval='prediction')
     fit   lwr   upr
1  1.456 1.310 1.603
2  1.462 1.315 1.608
3  1.509 1.364 1.654
4  1.704 1.563 1.844
5  1.825 1.685 1.964
6  1.909 1.770 2.048
7  2.124 1.984 2.264
8  2.214 2.072 2.355
9  2.214 2.072 2.355
10 2.929 2.761 3.098
Warning message:
In predict.lm(reg1, interval = "prediction") :
  Predictions on current data refer to _future_ responses
```

Regression2.

Generated data used to demonstrate 'problems' with the simple linear model. All can be fixed with transformations.

New Libraries/Functions/Arguments Illustrated:
"log", "sqrt", "lines", "mtext"

Example Code (Libraries, Functions or Arguments are in bold the first time they are used):

```
#Enter the data
> X=c(1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10)
>Y1=c(2.3,4.2,7.6,14.7,16.7,20.3,23.0,42.3,34.9,65.6,0.8,1.1,4.4,13.4,9.1,26.4,23.7,29.8,38.1,50.9)
> Y2=c(2.4,3.6,4.5,5.8,5.7,5.7,5.5,7.4,5.7,7.5,1.4,2.0,3.5,5.6,4.7,6.7,5.6,5.8,6.2,6.9)
> Y3=c(1.9,2.5,3.2,4.5,4.2,4.2,3.9,6.7,4.1,8.7,1.4,1.6,2.3,4.1,2.6,5.6,4.0,4.2,4.6,5.6)

#Transformations following SAS Code examples
> lnY1=log(Y1)
> lnY2=log(Y2)
> lnY3=log(Y3)
> sqrtY1=sqrt(Y1)
> sqrtY2=sqrt(Y2)
> sqrtY3=sqrt(Y3)
> lnX=log(X)
> sqrtX=sqrt(X)

#Plot of Y1 and X
> plot(X, Y1, pch=19, col='blue')

#Simple linear regression of Y1 versus X – we will invoke many of the same statements as in
#Regression1
> regY1X=lm(Y1~X)
> summary(regY1X)
> plot(regY1X,which=1)
> plot(regY1X,which=2)
> plot(regY1X,which=4)
> plot(regY1X,which=5)
> studY1X=studres(regY1X)
> plot(regY1X$fitted,studY1X,pch=19, ylim=c(-4,6))
> abline(h=0)

#Examine the fitted values, predictions, and respective errors
#We are using the Confidence interval and Prediction interval option
> predict(regY1X,interval='confidence')
> predict(regY1X,interval='prediction')

#Examine the fit of regression visually using:
> plot(X,Y1,type='p',pch=19,col='blue', ylim=c(-5,50))
> lines(X, regY1X$fitted, type='b', pch=20, col='red',lty=1, lwd=1.5)
> mtext(side=3, line=2,"Raw Data and Fitted Regression Line for Y1 Versus X")

#Because of the problems with the fit and residuals, try the square-root transformation and #follow the
code above for diagnostics
> plot(X,sqrtY1,pch=19,col='blue', ylim=c(0,10))
> sqrtY1X=lm(sqrtY1~X)

#AS EXERCISES, EXAMINE THE RESPONSE VARIABLES Y2 and Y3 AS WELL AS THE DIFFERENT TRANSFORMATIONS
```

Regression3.

Data from D. Johnson et al. (2009). Plant Disease 93: 272-280.

Relationship between late blight incidence and cumulative solar radiation (sr) from 1 April to 31 July.

Example Code – Run different models for the incidence versus solar radiation; try different transformations – Code follows from Regression1 and Regression2.

```
>Year=c(1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007)
> SR=c(2658,2635,2676,2560,2744,2644,2750,2628,2659,2775,2720,2885,2869,2749,2705,2684,2819,2951)
> Incidence=c(0.4,24,42,45,1.1,80,44,70,64,5,13,0.6,1.8,3,35,22,5.7,0.6)
> logInc=log(Incidence/100)
> logitInc=log(Incidence/(100-Incidence))
> cll=log(-log(1-(Incidence/100)))
> logSR=log(SR)
```

Regression4.

Data from Paul, El-Allaf, Lipps, and Madden (2004). Phytopathology 94: 1342-1349.

Relationship between spore flux density (Fusarium graminearum) in spore samplers and rain-splash intensity.

Assume the best relationship is a linear model between log10(sporeflux) as the response variable and log10(splash) as the predictor variable. In reality, the data are from different elevations in a wheat canopy, but we will pool together.

New Libraries/Functions/Arguments Illustrated:
"read.csv", "range", "log10", "rlm", "order", "quantreg", "rq", "points", loops

In the example code, we illustrate reading a .csv file from a directory in Windows. For illustration, a generic "temp" folder on the C-drive was created. Reading in files of .txt, .csv, etc., from different computers (Mac, PC, etc.) are slightly different and we recommend you consulting the reference material for further information.

Example Code (Libraries, Functions or Arguments are in bold the first time they are used):

```
> pauletal=read.csv('c:/temp/regression4_09.csv',header=T)
> pauletal[1:5,] #Just a check to see that the data appear the way we intended
  splash sporeflux
1 1.1979     0.719
2 1.1780     0.704
3 0.9965     2.491
4 0.3616     0.267
5 0.8589     1.288

#We use "range" to examine the data range for plotting purposes
> range(pauletal$splash)
[1]  0.1582 11.5371
> range(pauletal$sporeflux)
[1]   0.198 136.263

#Attach allows us to call our different column headings directly (corresponding function to #detach
this option is "detach"
> attach(pauletal)
> plot(splash,sporeflux,xlim=c(0, 12), ylim=c(0, 140), pch=19,col='blue')

#Transformations – log, base 10
> lgsplash=log10(splash)
> lgsporeflux=log10(sporeflux)

#Plots of the transformed variables
> plot(lgsplash, lgsporeflux, xlim=c(-1,1.5), ylim=c(-1, 2.5), pch=19, col='green')

#Following the SAS code, examine the following for simple linear regression, robust regression, #and
quantile regression
> reg1_trans=lm(lgsporeflux~lgsplash)
> library(MASS) #Needed to call the function for robust regression 'rlm'
> influence.measures(reg1_trans)

#To extract some for the similar output as SAS, we call "summary" for each variable in the model:
> summary(lgsplash)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.80080 -0.05697  0.12990  0.11410  0.38070  1.06200
> summary(lgsporeflux)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.7033  0.1447  0.4684  0.4564  0.7976  2.1340

#Robust regression using M-estimation
> robust=rlm(lgsporeflux~lgsplash, method="M")
> summary(robust) #OUTPUT NOT SHOWN…compare this with the estimates from the OLS

#To examine of the fit of the robust regression model in comparison with the OLS
> stdres=stdres(reg1_trans) #Extract the studentized residuals from the OLS
```

```
> cooksd=cooks.distance(reg1_trans) #Extract Cook's distances
> new.data=cbind(pauletal, stdres, cooksd, robust$w)
#Combine original data with measures from OLS and Robust regression (robust$w = weights)
> new.data[1:5,] #Data not shown
> assorted=new.data[order(robust$w),]
#Ordering the data based on the weights – typically, larger residuals have lower weights in the #Robust
regression

#Quantile regression
#We will need to use the rq function in the package quantreg
#quantreg needs to be downloaded from CRAN
> library(quantreg)
> quant1=rq(lgsporeflux~lgsplash, tau=c(0.1,0.25,0.5,0.75,0.90))
> summary(quant1)

#To create a plot of the raw data and quantile regression lines, the following code can be used #based
on Koenker, 'Quantile Regression in R : A Vignette'

> plot(lgsplash,lgsporeflux, type='n') #Type = 'n' calls for nothing to be plotted
> points(lgsplash,lgsporeflux,cex=0.5, col='blue') #Plotting the raw data
> abline(rq(lgsporeflux~lgsplash,tau=0.5),col='blue') #Adding the median quantile regression line
> taus=c(0.1,0.25,0.75,0.9)

#The following code runs a loop and plots different quantile regression lines
> for (i in 1:length(taus)) {
abline(rq(lgsporeflux~lgsplash, tau=taus[i]),
col='gray')
}
```

Regression5.

Example 5: Multiple Regression -- biological response to temperature.

Growth of Colletotrichum coccodes on PDA at different temperatures.

t: temperature (C)
isolE: linear growth rate on PDA for isolate E
isolS: linear growth rate on PDA for isolate S
growth: re-names response variable (growth rate of selected isolate [S])

New Libraries/Functions/Arguments Illustrated:
"qqnorm", "qqline"

Example Code (Libraries, Functions or Arguments are in bold the first time they are used):

```
#Manual data entry and recode isolS to growth (see SAS notes)
> temp=c(5,7.5,10,12.5,15,17.5,20,22.5,25.0,26.5,27.5,28.5,29,30,32.5,33.5,35)
> isolE=c(0,1,6.3,10,13.1,16.7,23.3,28.3,31.6,34.1,31.7,30.3,29.0,25.1,23.0,12.8,2.0)
> isolS=c(0.5,1,6.3,11.8,15.1,19.5,25.9,31.1,37.3,40.3,38.6,38.6,35.4,31.6,19.5,9.2,0)
> growth=isolS

#Plot of growth versus temperature
> range(temp)
[1]  5 35
> range(growth)
[1]  0.0 40.3
> plot(temp,growth,xlim=c(2,38), ylim=c(0,42),type='p',pch=19,col='blue')

#Set up different transformations
> srgrowth=sqrt(growth)
> lgrowth=log(growth+1)
> temp12=sqrt(temp)
> temp2=temp^2 #Note, in R, can also be written as temp**2
> temp3=temp^3 #Also as temp**3
> temp4=temp^4 #Also as temp**4
> ltemp=log(temp+0.1)
> ltemp2=ltemp^2
> ltempn1=log(temp)
> ltempnh=log(36-temp)
> wt=growth+1
> wt2=growth^2

#Running a quadratic model for temperature – still use the lm function in R
> quadt=lm(growth~temp + temp2)
> summary(quadt) #Output not shown

#Diagnostics – partial list compared with SAS ouput
> library(MASS)
> stdquadt=studres(quadt)
> range(stdquadt) #Output not shown
> plot(quadt$fitted.values, stdquadt, ylim=c(-3,3), pch=19, col='blue')
> abline(h=0)
> mtext(side=3, line=1.5, "N=17, Rsq=0.7083, AdjRsq=0.6666")
> qqnorm(studres(quadt)) #QQ Normal Plot of Studentized Residuals
> qqline(studres(quadt))
#QQ Line added to QQ Plot that passes through the first and third quartiles
> influence.measures(quadt) #Different measures of influence – see regression1 for further info
> plot(temp,quadt$residual,pch=19,col='blue') #Plot of residuals against each predictor variable
> abline(h=0)
> plot(temp2,quadt$residual,pch=19,col='green')
> abline(h=0)

#Plot of the raw data along with the fitted regression line
> plot(temp,growth,xlim=c(2,38), ylim=c(0,42),type='p',pch=19,col='blue')
> lines(temp,quadt$fitted,lty=1,col='dark red', lwd=2)

#TRY FURTHER TWO-VARIABLE REGRESSION MODELS WHERE EACH VARIABLE IS A POWER OF TEMPERATURE
```

Regression6.

Use of a smoothing spline ("nonparametric" regression model).

Daily weather data from Wooster, OH, in 2008, starting on May 1.
temper: daily average T (F)
RH: daily average RH
rain: daily total rain

In R, there are different functions that can be used for fitting smoothing splines or locally weight regression models. For comparison with SAS, we will use the "smooth.spline" function – note, there may be some slight differences in the fitting algorithms between SAS and R, however, both are based on methods of cross-validation (ordinary or generalized).

To fully understand the suite of nonparametric regression in R, consult the following to start:

Venables and Ripley. 2002. Modern Applied Statistics with S. Springer, New York.

Note: Upon running the example analyses, it appears that there may be a slight difference in the df fitting between SAS and R based on a visual assessment of the plots for different dfs.

New Libraries/Functions/Arguments Illustrated:
"smooth.spline"

Example Code (Libraries, Functions or Arguments are in bold the first time they are used):

```
#Similar to Regression4, data are brought into R using the read.csv function (data were compiled #in
column format and saved as a .csv file)
> weather=read.csv('c:/temp/weather.csv', header=T)
> weather[1:10,] #Check of the data
   day temper RH rain
1    0   59.4 56 0.00
2    1   66.3 62 0.31
3    2   60.2 81 0.38
4    3   52.0 66 0.00
5    4   55.9 62 0.00
6    5   62.0 60 0.01
7    6   64.1 66 0.08
8    7   53.9 88 0.30
9    8   55.1 79 0.24
10   9   53.9 69 0.07
> attach(weather)

#Examine plot of temperature (temper) and day
> plot(day,temper,pch=3,col='blue', cex=0.25)
#Different plotting character requested and cex = 0.25 reduced the size by ¼ for visual effects

#Run smooth.spline with degrees-of-freedom = 6
> spline1=smooth.spline(x=day,y=temper,df=6)
#For this, better to code "x" and "y" to minimize switching variables around

> spline1
Call:
smooth.spline(x = day, y = temper, df = 6)

Smoothing Parameter  spar= 0.8253714  lambda= 0.00314444 (12 iterations)
Equivalent Degrees of Freedom (Df): 6.000922
Penalized Criterion: 3191.715
GCV: 28.67903

#Note, the equivalent degrees of freedom slightly different from SAS output
#Plot the fitted line from the analysis (if you have not closed your original plot, can just use #the
line option and "add=T", otherwise, you will need to plot the raw data again
> predict.spline1=predict(spline1)
> names(predict.spline1)
[1] "x" "y"
> plot(day,temper,pch=3,col='blue', cex=0.25)
```

```
> lines(predict.spline1, lty=1, lwd=2.5, col='dark red')

#Example for RH
> plot(day, RH, pch=3, cex=0.5, col='blue')
> splineRH=smooth.spline(x=day,y=RH) #No specification for df…will overfit (dfs=23.79…)
> lines(splineRH, lty=1,lwd=2.5, col='dark red')
```