Introduction to R (see R-start.doc)

Be careful -- R is case sensitive.

Setting and getting the working directory

- Use File > Change dir...
- setwd("P:/Data/MATH/Hartlaub/Regression")
- getwd()

Reading data (Creating a dataframe)

- mydata=read.csv(file=file.choose())
- mydata=read.table(file=file.choose()) #use to read in the txt files
- mydata=read.table("marks.txt",header=T)#eg of data read into R with column heading

Commands for dataframes

- mydata #shows the entire data set
- head(mydata) #shows the first 6 rows
- tail(mydata) #shows the last 6 rows
- str(mydata) #shows the variable names and types
- names(mydata) #shows the variable names
- rename(V1,Variable1, dataFrame=mydata) #renames V1 to Variable1; Note that epicalc package must be installed!
- ls() #shows a list of objects that are available
- attach(mydata) #attaches the dataframe to the R search path, which makes it easy to access variable names

Descriptive Statistics

- mean(x) #computes the mean of the variable x
- median(x) #computes the median of the variable x
- sd(x) #computes the standard deviation of the variable x
- IQR(x) #computes the IQR of the variable x
- summary(x) #computes the 5-number summary and the mean of the variable x
- cor(x,y) #computes the correlation coefficient
- cor(mydata) #computes a correlation matrix
- cor.test(x,y) #test plus CI for rho

Graphical Displays

- hist(x) #creates a histogram for the variable x
- boxplot(x) # creates a boxplot for the variable x
- boxplot(y~x) # creates side-by-side boxplots
- DOTplot(x) #creates a dotplot (UsingR package must be installed)
- stem(x) #creates a stem plot for the variable x
- plot(y~x) #creates a scatterplot of y versus x
- plot(mydata) #provides a scatterplot matrix
- abline(lm(y~x)) #adds regression line to plot
- lines(lowess(y~x)) # adds locally weighted scatterplot smoother line to plot
- qplot(x, y) #creates a quick plot (ggplot2 package must be installed)
- ci.plot(regmodel) #creates a scatterplot with fitted line, confidence bands, and prediction bands (HH package must be installed)

Liner Regression Models

- regmodel=lm(y~x) #fit a regression model
- summary(regmodel) #get results from fitting the regression model
- anova(regmodel) #get the ANOVA table fro the regression fit
- plot(regmodel) #get four plots, including normal probability plot, of residuals
- fits=regmodel$fitted #store the fitted values in variable named "fits"
- resids=regmodel$residuals #store the residual values in a varaible named "resids"
- beta1hat=regmodel$coeff[2] #assign the slope coefficient to the name "beta1hat"
- confint(regmodel) #CIs for all parameters
- predict.lm(regmodel, interval="confidence") #make prediction and give confidence interval for the mean response
- predict.lm(regmodel, interval="prediction") #make prediction and give prediction interval for the mean response
- newx=data.frame(X=4) #create a new data frame with one new x* value of 4
- predict.lm(regmodel, newx, interval="confidence") #get a CI for the mean at the value x*
- Tests for homogeneity of variance
    - bptest(regmodel) #get the Breusch-Pagan test (lmtest package must be installed)
    - levene.test(Y, groupvariable) #get the Levene test (lawstat package must be installed)
- Tests for normality
    - ad.test(resids) #get Anderson-Darling test for normality (nortest package must be installed)
    - cvm.test(resids) #get Cramer-von Mises test for normaility (nortest package must be installed)
    - lillie.test(resids) #get Lilliefors (Kolmogorov-Smirnov) test for normality (nortest package must be installed)
    - pearson.test(resids) #get Pearson chi-square test for normaility (nortest package must be installed)
    - sf.test(resids) #get Shapiro-Francia test for normaility (nortest package must be installed)

- Lack-of-fit test
    - Reduced=lm(y~x)#fit reduced model
    - Full=lm(y~0+as.factor(x)) #fit full model
    - anova(Reduced, Full) #get lack-of-fit test
- boxcox(regmodel) #evaluate possible Box-Cox transformations (MASS package must be installed)
- Model Selection
    - library(leaps) #load the leaps package
    - allmods = regsubsets(y~x1+x2+x3+x4, nbest=2, data=mydata) #(leaps package must be loaded), identify best two models for 1, 2, 3 predictors
    - summary(allmods) # get summary of best subsets
    - summary(allmods)$adjr2 #adjusted R^2 for some models
    - summary(allmods)$cp #Cp for some models
    - plot(allmods, scale="adjr2") # plot that identifies models
    - plot(allmods, scale="Cp") # plot that identifies models
    - fullmodel=lm(y~., data=mydata) # regress y on everything in mydata
    - MSE=(summary(fullmodel)$sigma)^2 # store MSE for the full model
    - extractAIC(lm(y~x1+x2+x3), scale=MSE) #get Cp (equivalent to AIC)
    - step(fullmodel, scale=MSE, direction="backward") #backward elimination
    - none(lm(y~1) #regress y on the constant only
    - step(none, scope=list(upper=fullmodel), scale=MSE) #use Cp in stepwise regression
- Diagnostics
    - sresids=rstandard(regmodel) #store the standardized residuals in a variable named "sresids"
    - standresid=stdres(regmodel) #store the standardized residuals in a variable named "standresids"
    - stud.del.resids=rstudent(regmodel) #store the studentized deleted residuals in a variable named "stud.del.resids"
    - hatvalues(regmodel) #get the leverage values (hi)
    - cooks.distance(regmodel) #get Cook's distance
    - dfbetas(regmodel) #print all dfbetas
    - dfbetas(regmodel)[4,1] #dfbeta for case 4, first coefficient (i.e., b_0)
    - dffits(regmodel) [4] #dffits for case 4
    - influence(regmodel) #various influence statistics, including hat values and dfbeta (not dfbetas) values
    - library(car) #load the package car
        - vif(regmodel) #variance inflation factors
        - avPlots(regmodel) #added variable plots