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/DataAnalysis")
- getwd()

Reading data (Creating a dataframe)

- mydata=read.csv(file=file.choose())

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
- 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) #computer the IQR of the variable x
- summary(x) #computes the 5-number summary and the mean of the variable x
- t.test(x) #get a one sample t test
- t.test(x,y) #get a two sample t test

- t.test(x, y, paired=TRUE) #get a paired t test
- cor(x,y) #computes the correlation coefficient
- cor(mydata) #computes a correlation matrix
- cor.test(x,y) #test plus CI for rho

Graphical Displays

- windows(record=TRUE) #records your work, including plots
- 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
- 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(x,y)) # adds lowess line (x,y) to plot

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"
- sresids=rstandard(regmodel) #store the standardized residuals in a variable named "sresids"
- studresids=rstudent(regmodel) #store the studentized residuals in a variable named "studresids"
- beta1hat=regmodel$coeff[2] #assign the slope coefficient to the name "beta1hat"
- qt(.975,15) # find the 97.5% percentile for a t distribution with 15 df
- confint(regmodel) #CIs for all parameters

- newx=data.frame(X=41) #create a new data frame with one new x* value of 41
- predict.lm(regmodel,newx,interval="confidence") #get a CI for the mean at the value x*
- predict.lm(model,newx,interval="prediction") #get a prediction interval for an individual Y value at the value x*
- hatvalues(regmodel) #get the leverage values (hi)
- 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
  - step(fullmodel, scale=MSE, direction="forward") #forward elimination
  - step(fullmodel, scale=MSE, direction="both") #stepwise regression
  - none(lm(y~1) #regress y on the constant only
  - step(none, scope=list(upper=fullmodel), scale=MSE) #use Cp in stepwise regression

Logistic Regression

- table(y) #get a table of the distribution of y

- mytable=table(y, x) #get a 2-way table of y by x
- chisq.test(mytable) #Chi-sq test with Yates continuity correction
- chisq.test(mytable, correction=FALSE) #Chi-sq test of independence without Yates continuity correction
- prop.table(table(y, x),1) #get a table of row proportions
- prop.table(table(y, x),2) #get a table of column proportions
- prop.test(c(39,22), c(100,100), correction=FALSE) #2-sample proportion test without Yates continuity correction
- plot(x,jitter(y,amount=0.05)) #jitter y in the plot
- anova(reducedmodel, fullmodel, test="Chisq") #nested G test
- drop1(mymodel, test="Chisq") #G tests to see what to drop next
- as.factor(X) #create dummy variables for the levels of the variable X
- model1=glm(y~as.factor(X), family=binomial) #fit model with the categories of X as predictors
- summary(model1) #gives Z tests, residual deviance, and null deviance
- anova(model1, test="Chisq") #test of H0: constant term is all that is needed. (i.e., nested G test against the model y~1.)
- confint(model1) #CIs for all parameters
- confint(model1, parm="x") #CI for the coefficient of x
- exp(confint(model1, parm="x")) #CI for odds ratio
- shortmodel=glm(cbind(y1,y2)~x, family=binomial) binomial inputs
- dresid=residuals(model1, type="deviance") #deviance residuals
- presid=residuals(model1, type="pearson") #Pearson residuals
- plot(residuals(model1, type="deviance")) #plot of deviance residuals
- newx=data.frame(X=20) #set (X=20) for an upcoming prediction
- predict(mymodel, newx, type="response") #get predicted probability at X=20

Analysis of Variance

- t.test(y~x, var.equal=TRUE) #pooled t-test where x is a factor
- x=as.factor(x) #coerce x to be a factor variable
- tapply(y, x, mean) #get mean of y at each level of x
- tapply(y, x, sd) #get stadard deviations of y at each level of x
- tapply(y, x, length) #get sample sizes of y at each level of x
- library(gplots) #load gplots package

    - plotmeans(y~x) #means and 95% confidence intervals
- AOVmodel=aov(y~x) #one-way ANOVA
- summary(AOVmodel) #get ANOVA output
- oneway.test(y~x, var.equal=TRUE) #one-way test output
- library(car) #load car package
    - levene.test(y,x) #Levene's test for equal variances
- blockmodel=aov(y~x+block) #Randomized block design model with "block" as a variable
- tapply(lm(y~x1:x2,mean) #get the mean of y for each cell of x1 by x2
- anova(lm(y~x1+x2)) #a way to get a two-way ANOVA table
- interaction.plot(FactorA, FactorB, y) #get an interaction plot
- pairwise.t.test(y,x,p.adj="none") #pairwise t tests
- pairwise.t.test(y,x,p.adj="bonferroni") #pairwise t tests
- TukeyHSD(AOVmodel) #get Tukey CIs and P-values
- plot(TukeyHSD(AOVmodel)) #get 95% family-wise CIs
- library(multcomp) #load multcomp package
    - contrast=rbind(c(.5,.5,-1/3,-1/3,-1/3)) #set up a contrast
    - summary(glht(AOVmodel, linfct=mcp(x=contrast))) #test a contrast
    - confint(glht(AOVmodel, linfct=mcp(x=contrast))) #CI for a contrast
- kruskal.test(y~x) #Kruskal-Wallis test
- friedman.test(y,x,block) #Friedman test for block design