R Session 1

HOW TO INPUT DATA

1) Manual input:

```
year <- c(2005, 2006, 2007, 2008, 2009, 2010)
lfp <- c(66, 66.2, 66, 66, 65.4, 64.7)
unemp <- c(5.1, 4.6, 4.6, 5.8, 9.3, 9.6)
data1 <- data.frame(year, lfp, unemp)
```

```
data1
```

[To view the data in *data1*]

2) Import from Excel:

[First, install the "xlsx" package. Packages - Install - Choose mirror - Choose xlsx. Once the package is installed you can always use it.]

```
library (xlsx)
data2 <- read.xlsx("C:/ Price and Mortgage Interest Rate.xls", 1)
data2
```

[The first row should contain variable names. Otherwise, the data will not be imported correctly. To see what happens, try opening *Price and Mortgage Interest Rate2.xls* file.]
[Note: in the directory, use \ on mswindows systems, / on mac.]

3) Import from a Comma Delimited Text File:
[Using this method, you can import files from different data formats. Stata, SPSS, SAS, Eviews, Excel and other format files can be saved in a *.csv* format. We can save *Price and Mortgage Interest Rate.xls* as *.csv* file.]

```
data3 <- read.csv("C:/Price and Mortgage Interest Rate.csv")
data3
```

HOW TO SAVE DATA

[File - Save Workspace]

HOW TO OPEN R DATA

[File - Load Workspace]

```
ls()
```

[List objects in the workspace (data frames, vectors, matrices, functions, etc.]

R Session 2

```
library (xlsx)
mydata <- read.xlsx("C:/Labor.xlsx", 1)
mydata
```
[Import *Labor.xlsx* dataset]

```
?subset
```
[Getting help on any command]

VIEWING VARIABLES

```
ls()
```
[Shows a list of objects that are available]

```
mydata
```
[Shows the entire data set]

```
head(mydata)
```
[Shows the first 6 rows]

```
tail(mydata)
```
[Shows the last 6 rows]

```
mydata$year
```
[Shows the variable year of mydata]

```
mydata$year [3]
```
[Shows the third element of the vector year of mydata]

```
str(mydata)
```
[Shows the variable names and types]

```
names(mydata)
```
[Shows the variable names]

```
attach(mydata)
```
[Attaches the dataframe to the R search path, which makes it easy to access variable names]
```
year
year [3]
```

```
detach(mydata)
```
[Undo attach]

RENAMING VARIABLES

```
names(mydata)
```
[View current names]

```
names(mydata) [3] <- "unemployment"
```
[Change the third element of the vector names. Repeat the previous command to see the changes]

DESCRIPTIVE STATISTICS

```
summary(mydata)
```
[Summarizes the variables, returns mean, min, max, median, 1st and 3rd quartiles]

```
summary(mydata$lfp)
```

```
mean(mydata)
```
[Computes the mean of all the variables]

```
median(mydata$year)
```
[Computes the median of the variable year of mydata]

```
sd(mydata$unemployment)
```
[Computes the standard deviation of the variable unemployment of mydata]

```
cor(mydata$year, mydata$lfp)
```
[Computes the correlation coefficient between variables]

```
cor(mydata)
```
[Computes a correlation matrix]

```
summary(subset(mydata, mydata$year >= 2007))
```
[Conditional summary]
[Relational operators:>,<,<=,>=, == (equal), != (not equal). Relational operators are used only for conditions.]

```
table(mydata$lfp)
```
[Computes frequencies of the variable]

```
table(mydata$lfp, mydata$year)
```
[Computes two-way frequencies]

```
prop.table(mydata$lfp)
```
[Computes relative frequencies (proportions) of the variable]

[Note: any output provided by commands in R can be saved as object in the workspace. For example,

```
k <- mean(mydata$year)
myfrequencies <- table(mydata$lfp)
```
Now number k and frequencies table myfrequencies are in the workspace and you can view or use them anytime.]

GENERATING NEW VARIABLES

```
mydata$x <- mydata$year + mydata$lfp
```

or

```
attach(mydata)
mydata$x <- year + lfp
```

[Create a new variable named x in mydata and found is a sum of year and lfp. Another examples below.]

```
mydata$y <- 5
mydata$z <- mean(year)
mydata$t [year<=2005] <- 7*lfp^2
mydata$t [year>2005] <- 0
```

```
mydata$m [year>2007] <-1
```
[Not specified values of the variables are missing (showed as NA). You can specify their values using the identical command.]

```
mydata$m [3] <- 8
```
[Replace the third value of the variable m of mydata with 8].

SELECTING OBSERVATIONS

```
newdata1 <- mydata[1:5,]
```
[First 5 observations]

```
newdata2 <- mydata[ which(mydata$year>2008), ]
```
[Conditional on other variables values]

REMOVING THE VARIABLES (and other objects)

```
mydata$x <- NULL
```
[Delete variable x of mydata]

```
remove(k)
remove(myfrequencies)
```
[Delete the stored objects]

R Session 3.1

```
d <- read.csv("C:/Income and Consumption.csv")
d
```
[Import *Income and Consumption.csv* dataset]

CREATING A SCATTER PLOT

```
plot(d$Consumption ~ d$Income)
```
[Creates a scatterplot of Consumption versus Income]

```
abline(lm(d$Consumption ~ d$Income))
```
[Adds regression line to the plot. Can be used only after the previous command.]

[Note:
```
plot(mydata)
```
Creates a scatterplot matrix of all pairs of variables]

ESTIMATION OF REGRESSION

```
myreg <- lm(d$Consumption ~ d$Income)
```
[Estimates a regression model *Consumption = B1 + B2*Income + u* using OLS method.]

```
summary(myreg)
```
[Get results from fitting the regression model]

```
anova(myreg)
```
[Get the ANOVA table for the regression fit]

```
confint(myreg)
```
[Get confidence intervals for all parameters]

R Session 3.2

```
> d <- read.csv("C:/Child Mortality.csv")
d
```
[Use *Child Mortality.csv* dataset.]

REGRESSIONS

```
> reg1 <- lm(d$CM ~ d$FLR)
> summary(reg1)
```
[Estimates a regression model CM = B1 + B2*FLR + u.]

```
> reg2 <- lm(d$CM ~ d$FLR + d$PGNP)
> summary(reg2)
```
[Estimates a regression model CM = B1 + B2*FLR + B3*PGNP + u.]

```
> reg3 <- lm(d$CM ~ d$FLR + d$PGNP + d$TFR)
> summary(reg3)
```
[Estimates a regression model CM = B1 + B2*FLR + B3*PGNP + B4*TFR + u.]

```
> anova(reg3)
> confint(reg3)
```

```
> c <- read.csv("C:/Education Expenditure.csv")
c
```
[Use *Education Expenditure.csv* dataset.]

```
> reg <- lm(c$EDUC ~ c$GDP + c$POP)
> summary(reg)
```
[Estimates a regression model EDUC = B1 + B2*GDP + B3*POP + u.]
[Note: This is an example of a regression where all the variables are jointly significant (high value of F-test), but not all the variables are individually significant.]

R Session 4

```
ls()
```
[Shows all the objects in the workplace]

[Use *"Math SAT Score"* dataset. It is saved in the workplace as "m" data.]

```
m
```

```
m$lny <- log(m$y)
m$lnx <- log(m$x)
```
[Create two new variables in the m dataset.]

```
m
```
[Now this dataset contains four variables: y, x, lny, lnx.]

```
regm1 <- lm(m$y ~ m$x)
summary(regm1)
```
[Regress y on x (linear in x model)]

```
regm2 <- lm(m$lny ~ m$lnx)
summary(regm2)
```
[Regress lny on lnx (example of a double log model)]

```
regm3 <- lm(m$y ~ m$lnx)
summary(regm3)
```
[Regress y on lnx (example of a lin-log model)]

```
regm4 <- lm(m$lny ~ m$x)
summary(regm4)
```
[Regress lny on x (example of a log-lin model)]

[Use *"Production Function for Mexican Economy"* dataset. It is saved in the workplace as "f" data.]

```
f
```

```
f$lngdp <- log(f$gdp)
f$lnlabor <- log(f$labor)
f$lncapital <- log(f$capital)
```
[Create three new variables in the f dataset.]

```
f
```
[Now this dataset contains new variables.]

```
regf <- lm(f$lngdp ~ f$lnlabor + f$lncapital)
summary(regf)
```
[This is another example of the model where both dependent variable and independent variables are in log form (double-log or "log-linear" model)]

[Use *"Population Growth"* dataset. It is saved in the workplace as "p" data.]

```
head(p)
p$lnuspopulation <- log(p$uspopulation)
head(p)
```

```
regp <- lm(p$lnuspopulation ~ p$time)
summary(regp)
```
[This is another example of log-lin model]
[NOTE: Log-lin and log-linear models are not the same! Log-linear is the same as double-log.]

[Use *"Inflation and Unemployment"* dataset. It is saved in the workplace as "u" data.]

```
head(u)
u$unrateinv <- 1/u$unrate
head(u)
```

```
regu <- lm(u$inflrate ~ u$unrateinv)
summary(regu)
```
[This is an example of a reciprocal model; compare it with the linear model]

[Use *"Hourly Wage and Age"* dataset. It is saved in the workplace as "w" data.]

```
head(w)
w$agesq <- w$age^2
head(w)
```

```
regw <- lm(w$wage ~ w$age + w$agesq)
summary(regw)
```
[This is an example of a polynomial model]

[Use *"Political Instability and Size Interaction*" dataset. It is saved in the workplace as "i" data.]

```
head(i)
i$sizepins <- i$size*i$pins
head(i)
```

```
regi <- lm(i$lnprod ~ i$size + i$pins + i$sizepins)
summary(regi)
```
[This is an example of a model with an interaction term]

[Use again "*Math SAT Score"* dataset. It is save in the workplace as "m" data.]

```
regm5 <- lm(m$y ~ 0 + m$x)
summary(regm5)
```
[This is an example of regression through the origin (no constant regression)]

[Use again "*Math SAT Score"* dataset. It is save in the workplace as "m" data.]

```
> m$stdy <- scale(m$y)
> m$stdx <- scale(m$x)
```
[Standardize variables x and y, and save as new variables]

```
m
```

```
regm6 <- lm(m$stdy ~ 0 + m$stdx)
summary(regm6)
```
[Run a regression of standardized variables. Must be a regression through the origin.]

R Session 5.1

[Import *PUMS_NYC.csv* dataset and denote it as "d".]

TYPES OF VARIABLES
[There are two main types of variables in datasets: numeric and categorical.]

[**Numeric** (or quantitative) variables (such as *age*, *travel_time_to_work*, *personal_income*) have meaningful numeric values and are subject to all arithmetic operations. They have meaningful mean, standard deviation, median, and etc.]

```
summary(d$age)
summary(d$travel_time_to_work)
summary(d$personal_income)
```

[We may also find the frequencies for different values, although this information is not usually practically useful.]

```
table(d$age)
table(d$travel_time_to_work)
table(d$personal_income)
```

[**Categorical** (or qualitative) variables (such as *sex*, *boroughs*, *marital_status*) assign individual observations to some particular groups, or categories.

Categorical variables may be presented as **text code**:
sex (male, female)
boroughs (the Bronx, Manhattan, Staten Island, Brooklyn, Queens)
marital_status (married, widowed, divorced, separated, never married)

For these variables, descriptive statistics makes no sense. There is no way to calculate the mean marital status, or the minimum sex. On the other hand, frequencies are very useful.]

```
table(d$sex)
table(d$boroughs)
table(d$marital_status)
```

[R automatically treats variables with the text code as "factor" variables, and the command summary will show frequencies as well.]

```
summary(d$sex)
summary(d$boroughs)
summary(d$marital_status)
```

[Categorical variables may be presented as **numeric code**:
sex2 (1, 2)
boroughs2 (1, 2, 3, 4, 5)
marital_status2 (1, 2, 3, 4, 5)

The numbers are assigned randomly and are not meaningful. They can be changed among each other, or even replaced by any other numbers. Nothing prevents us from having a code like this: {*the Bronx = 4, Manhattan = 100, SI = -3, Brooklyn = 7500, Queens = 0*}.
R doesn't know that these variables are, in fact, categorical. So you may technically find descriptive statistics, but it will be making no sense.]

```
summary(d$sex2)
summary(d$boroughs2)
summary(d$marital_status2)
```

[At the same time, frequencies are very useful.]

```
table(d$sex2)
table(d$boroughs2)
table(d$marital_status2)
```

[There is one special type of variables – **binary** variables (or dummy variables). They take a value of 1 for presence of some property and a value of 0 for absence of that property. In other words, 1 means "true" and 0 means "false". The examples of binary variables are *africanamerican, kids_under6, foreign_born*.
For binary variables, both the descriptive statistics and the frequencies are useful.]

```
summary(d$africanamerican)
summary(d$kids_under6)
summary(d$foreign_born)
table(d$africanamerican)
table(d$kids_under6)
table(d$foreign_born)
```

[So binary variables have the features of both quantitative and qualitative variables.]

	Numeric	Binary	Categorical
Descriptive Statistics	+	+	-
Frequencies	-	+	+

HOW TO CREATE DUMMY VARIABLES
[We can create dummy variables using any variables.
Consider first how to create dummies **out of categorical variables**.

```
d$male [d$sex==male] <-1
d$male [d$sex==female] <- 0
```

or

```
d$male [d$sex2==1] <-1
d$male [d$sex2==2] <-0
```

[Now we will create dummies out of *boroughs* (*1=the Bronx, 2=Manhattan, 3=Staten Island, 4=Brooklyn, 5=Queens*)]

```
d$bronx [d$boroughs2==1] <- 1
d$bronx [d$boroughs2!=1] <- 0
d$manhattan [d$boroughs2==2] <- 1
d$manhattan [d$boroughs2!=2] <- 0
d$si [d$boroughs2==3] <- 1
d$si [d$boroughs2!=3] <- 0
d$brooklyn [d$boroughs2==4] <- 1
d$brooklyn  [d$boroughs2!=4] <- 0
d$queens [d$boroughs2==5] <- 1
d$queens [d$boroughs2!=5] <- 0
```

[There is a special command that automatically creates dummies for each category of categorical variable]

```
d$bor <- as.factor(d$boroughs2)
d$dummies  <- model.matrix(~d$bor)
```

[Now consider how to create dummies **out of numeric variables**. We may be interested in using such variables in some cases. We will use the variable *age*.]

```
d$young [d$age<=25] <- 1
d$young [d$age>25] <- 0
```

[We can have more groups]

```
d$old [d$age>65] <- 1
d$old [d$age<=65] <- 0
```

```
d$midage [d$young==0&d$old==0] <- 1
d$midage [is.na(d$midage)] <- 0
```

[In the last command we replaced all the missing values with 0. You can always use this trick when creating dummies.]

[Recall that "&" means "and" (intersection), "|" means "or" (union)]

[We may create a dummy for **more than one category of a qualitative variable.**]

```
d$brq [d$boroughs2==4|d$boroughs2==5] <- 1
d$brq [ is.na(d$brq)] <- 0
```

R Session 5.2

[Use *PUMS_NYC* dataset.]

```
reg1 <- lm(d$personal_income ~ d$male)
summary(reg1)
```
[Regress dependent variable on one dummy]

```
reg2 <- lm(d$personal_income ~ d$male + d$age)
summary(reg2)
```
[Regress dependent variable on one dummy and one quantitative variable]

```
reg3 <- lm(d$personal_income ~ d$male + d$africanamerican)
summary(reg3)
```
[Regress dependent variable on two dummies]

```
reg4 <- lm(d$personal_income ~ d$bronx + d$manhattan + d$si +
d$brooklyn)
summary(reg4)
```
[Regress dependent variable on several dummies that represent one categorical variable. The variable *queens* is excluded to avoid multicollinearity.]

```
reg5 <- lm(d$personal_income ~ d$bronx + d$si + d$brooklyn + d$queens)
summary(reg5)
```
[The same case as the previous one; now *manhattan* is excluded]

```
reg6 <- lm(d$personal_income ~ factor(d$boroughs))
summary(reg6)
```
[Another way to do the regression 4. We ask R to use dummies created from categories of *boroughs*.]

```
reg7 <- lm(d$personal_income ~ d$age + d$work_experience + d$male +
d$africanamerican + d$nativeamerican + d$asianamerican + d$Hispanic +
d$kids_under6 + d$foreign_born + factor(d$marital_status))
```
summary(reg7)
[Regress dependent variable on many quantitative and qualitative variables]

```
d$formanh <- d$foreign_born*d$manhattan
```
[This generates the interaction term for foreign_born and Manhattan]

```
reg8<- lm(d$personal_income ~ d$foreign_born + d$manhattan +
d$formanh)
summary(reg8)
```
[Regress dependent variable on two dummies and their interaction]

```
d$agemale <- d$age*d$male
```
[This generates the interaction term for age and male]

```
reg9<- lm(d$personal_income ~ d$age + d$male + d$agemale)
summary(reg9)
```

[Regress dependent variable on one quantitative variable, one dummy, and on their interaction]

[Use *Refrigerator Sales* dataset. Denote it as "r".]

```
regr <- lm(r$refrigerator_sales ~ r$quarter2 + r$quarter3 +
r$quarter4)
summary(regr)
```

[This is an example of a seasonal analysis]

[Use again PUMS_NYC dataset]

```
reglinprob <- lm(d$manhattan <- d$age + d$male + factor(d$educ_indx))
summary(reglinprob)
```

[This is an example of a linear probability model, where the dependent variable is binary. We are predicting the probability that a person lives in Manhattan given his age, gender and level of education.]

R Session 6

[Use *Los Angeles Restaurants* dataset]

```
head(la)
```

```
attach(la)
```
[Now we are working with the "la" dataset, and we don't need to use "la$" when referring to existing variables. We, however, need to use "la$" when we generate a new variable.]

```
reg <- lm(price ~ food + service)
summary(reg)
```
[This is our main regression model. We will be testing it for heteroskedasticity using different indicators.]

```
la$pricehat <- fitted(reg)
```
[Using the results of the last regression, compute predicted values of dependent variable, or yhats; save as a new variable named pricehat]

```
la$res <- resid(reg)
```
[Using the results of the last regression, compute residuals; save as a new variable named res]

```
la$ressq <- res^2
```
[Generate squares of residuals; save as a new variable named ressq]

```
hist(res)
hist(ressq)
```
[Create histogram of residuals or squared residuals distribution]

```
plot(ressq ~ food)
plot(ressq ~ service)
```
[Create residual plot: squared residuals versus explanatory variable, to detect a possible heteroscedasticity]

```
bpf <- lm(ressq ~ food)
summary(bpf)
bps <- lm(ressq ~ service)
summary(bps)
bpp <- lm(ressq ~ pricehat)
summary(bpp)
```

[Breusch-Pagan test: regress squared residuals on the X-variable; if more than one X are correlated with residuals, then regress squared residuals on Yhat (since Yhat is a linear combination of X's.]
[If F-test is significant, then there may be heteroscedasticity.]

```
la$lnressq <- log(ressq)
la$lnpricehat <- log(pricehat)
```
[Generate two new variables.]

```
park <- lm(lnressq ~ lnpricehat)
summary(park)
```
[Park test: regress log of squared residuals on X-variable (or Yhat)]
[If the coefficient on the explanatory variable used is significant, then there may be heteroscedasticity.]

```
la$absres <- abs(res)
la$rootpricehat <- sqrt(pricehat)
la$pricehatinv <- 1/pricehat
```
[Generate three new variables.]

```
glejser1 <- lm(absres ~ pricehat)
summary(glejser1)
glejser2 <- lm(absres ~ rootpricehat)
summary(glejser2)
glejser3 <- lm(absres ~ pricehatinv)
summary(glejser3)
```
[Glejser test in three versions: regress absolute value of residuals on X, or square root of X, or X inverse]
[If the coefficient on the explanatory variable used is significant, then there may be heteroscedasticity.]

```
la$foodsq <- food^2
la$servicesq <- service^2
la$foodservice <- food*service
[Generate three new variables]
```

```
white <- lm(ressq ~ food + service + foodsq + servicesq + foodservice)
summary(white)
```
[White's test: regress squared residuals on X's, there squares and cross-products.]
[If n*R-squared is higher than Chi-Sq(k-1), then there may be heteroscedasticity.]

[Load the packages "lmtest" and "car". Use the dropdown menu Packages - Install Package - Choose any mirror from US - find the package "lmtest" - Ok. Do the same for "car" package.]

```
library(lmtest)
library(car)
```

```
coeftest(reg, vcov= hccm)
```
[Run the original regression (price on food and service), but estimate standard errors using White's

heteroscedasticity correction.]

```
plot(ressq ~ pricehat)
```

[Create a residual plot to check if there is linear or a quadratic relationship between squared residuals and X's]

[If the relationship is linear]

```
la$y <- price/rootpricehat
la$x1 <- 1/rootpricehat
la$x2 <- food/rootpricehat
la$x3 <- service/rootpricehat
```

[Generate four new variables.]

```
transformlin <- lm(y ~ 0 + x1 + x2 + x3)
summary(transformlin)
```

[Perform a square root transformation of the original regression to stabilize the variance of disturbances.]

[At this point we can try to check if we got rid of heteroscedasticity, using any of the methods. We predict residuals and test them for relationship with X's.]

```
la$yhat <- fitted(transfromlin)
la$r <- resid(transformlin)
la$rsq <- r^2
transformlincheck <- lm(rsq ~ yhat)
summary(transformlincheck)
```

[If the F-test is not significant, then we actually got rid of heteroscedasticity.]

[If the relationship is not linear]

```
la$y <- price/pricehat
la$x1 <- 1/pricehat
la$x2 <- food/pricehat
la$x3 <- service/pricehat
```

[Generate four new variables.]

```
transformsq <- lm(y ~ 0 + x1 + x2 + x3)
summary(transformsq)
```

[Perform a square root transformation of the original regression to stabilize the variance of disturbances.]

[At this point we can try to check if we got rid of heteroscedasticity, using any of the methods. We predict residuals and test them for relationship with X's.]

```
la$yhat <- fitted(transfromlin)
la$r <- resid(transformlin)
```

```
la$rsq <- r^2
transformsqcheck <- lm(rsq ~ yhat)
summary(transformsqcheck)
```

[If the F-test is not significant, then we actually got rid of heteroscedasticity.]

```
la$lnprice <- log(price)
la$lnfood <- log(food)
la$lnservice <- log(service)
newreg <- lm(lnprice ~ lnfood + lnservice)
summary(newreg)
```

[Respecification of the original regression. May be used if the log-linear functional form is also acceptable, which is not always the case.]

R Session 7

[Use *Dividends and Corporate Profits* dataset.]

```
head(t)
```

```
reg <- lm(t$div ~ t$prof)
summary(reg)
```

[This is our main regression model. We will be testing it for autocorrelation using different indicators.]

```
t$res <- resid(reg)
t$time <- 1:244
```

[Create a sequence variable (1, 2, 3, 4, etc.) named *time*.]

```
plot(t$res ~ t$time)
```

[Plot residuals against time to assess the possible autocorrelation]

[Install package DataCombine]

```
library(DataCombine)
```

```
t$reslag <- shift(t$res, -1)
```

[Generate a lagged variable for *res* named *reslag* (The first value of *reslag* is the second value of *res*; the second value of *reslag* is the third value of *res*, and etc.).]

```
plot(t$res ~ t$reslag)
```

[Plot residuals against one period lagged residuals to assess the possible autocorrelation]

[Install package lmtest]

```
library(lmtest)
```

```
dwtest(reg)
```

[Perform Durbin-Watson test for autocorrelation.]

[Now let's do the same test (DW) manually.]

```
t$x <- (t$res-t$reslag)^2
sum(t$x, na.rm=TRUE)
```

[The second argument is for ignoring the missing values.]

```
anova(reg)
```

[Out of the anova table we take the RSS which is the of d-statistic.]

```
70377.79/118661
```

[The results are the same.]

[Install package randtests]

```
library(randtests)
```

```
runs.test(t$res, "two-sided", 0)
```
[Perform runs test, use the variable res and set threshold level at 0. (If this option is not specified, the default threshold is the median value).]

[Model transformations: using different values of rho.]

```
t$divlag <- shift(t$div, -1)
t$proflag <- shift(t$prof, -1)
```

[Generate one period lagged variables for dependent and independent variables]

```
rho1 <- 1
```
[Set the first possible estimate of rho as 1]

```
rho2 <- 1-(0 .5931/2)
rho2
```
[Find the second possible estimate of rho out of DW test]

```
resmodel <- lm(t$res ~ t$reslag)
summary(resmodel)
rho3 <- 0.72883
```
[Find the third possible estimate of rho from the residuals regression]

[Run the first transformed regression]

```
t$divstar1 <- t$div – rho1*t$divlag
t$divstar1 [1] <- t$div*sqrt(1-rho1^2)
t$profstar1 <- t$prof – rho1*t$proflag
t$profstar1 [1] <- t$prof*sqrt(1-rho1^2)
```

```
transformed1 <- lm(t$divstar1 ~ 0 + t$profstar1)
summary(transformed1)
```

[Test it for autocorrelation]

```
dwtest(transformed1)
```

```
t$resstar1 <- resid(transformed1)
runs.test(t$resstar1, "two.sided", 0)
```

[Run the second transformed regression]

```
t$divstar2 <- t$div – rho2*t$divlag
t$divstar2 [1] <- t$div*sqrt(1-rho2^2)
t$profstar2 <- t$prof – rho2*t$proflag
t$profstar2 [1] <- t$prof*sqrt(1-rho2^2)
```

```
transformed2 <- lm(t$divstar2 ~ t$profstar2)
summary(transformed2)
```

[Test it for autocorrelation]

```
dwtest(transformed2)
```

```
t$resstar2 <- resid(transformed2)
runs.test(t$resstar2, "two.sided", 0)
```

[Run the third transformed regression]

```
t$divstar3 <- t$div – rho3*t$divlag
t$divstar3 [1] <- t$div*sqrt(1-rho3^2)
t$profstar3 <- t$prof – rho3*t$proflag
t$profstar3 [1] <- t$prof*sqrt(1-rho3^2)
```

```
transformed3 <- lm(t$divstar3 ~ t$profstar3)
summary(transformed3)
```

[Test it for autocorrelation]

```
dwtest(transformed3)
```

```
t$resstar3 <- resid(transformed3)
runs.test(t$resstar3, "two.sided", 0)
```

[Install package sandwich]

```
library(sandwich)
```

```
coeftest(reg, vcov = NeweyWest)
```

[Run a regression with Newey-West HAC (heteroscedasticity and autocorrelation) corrected standard errors. Compare to the original OLS regression: standard errors were underestimated.]