```
# Intro to R lecture

# Anything after '#' will not be submitted to the console for
processing,
# but will be printed in the console

#
#===============@ Clearing the Workspace
@================================#
#
# This clears everything above it from the workspace. Thus, functions
and
# seeds should be put after this.
#

rm(list=ls())

#===============@ Direct Submission to the Console
@======================#

1 + 2 # This returns '3'

try(a) # Won't work. R is looking for an object called 'a', but we
haven't defined 'a' yet

#===============@ Sequences
@=============================================#

1:10

#
#===============@ Creating objects(vectors)
@=============================#
#

x <- 2 # 'x' is a vector/object that contains the element '2'
x

y <- c(1, 2, 3) # 'y' is a vector/object that contains three elements:
1, 2, 3
y

z <- 1:3 # Same thing as 'y'
z
class(z) # Tells us what the object 'z' is

z[2] # This retuns the second element of the 'z' vector
z[-2] # This returns all of the elements of 'z' EXCEPT the second
element
z[c(1,2)] # This returns the first and second elements of 'z'
```

```
#
#===============@ Mathematical Operations
@===============================#
#

x <- 2 + 2 # creates an object, 'x', that contains the results of '2 +
2'
x # Note that this overwrites the previous object called 'x'

y <- 2
x/y
x*y

xx <- c(1, 2, 3)
yy <- c(4, 5, 6)

xx + yy # adding two objects of the same length

#
#===============@ Statistical Operations
@================================#
#

y <- 1:10
mean(y) # returns the mean of the sum of 1 through 10
help(mean)

z <- mean(y) # stores the mean of the sum of 1 through 10 as an object
z

sd(y) # returns the standard deviation of 'y'
max(y) # returns the maximum of 'y'
min(y) # returns the minimum of 'y'
median(y) # returns the median of 'y'
sum(y) # returns the sum of 'y'

#
#================@ Generating Data
@=======================================#
#

x <- rnorm(100, mean = 5, sd=1) # Normal distribution: 100 obs, mean =
5 and sd = 1
help(rnorm)
z <- runif(100, min = -1, max = 1) # Uniform distribution: 100 obs,
min = -1 and max = 1

#
#=================@ Generating a Data Frame
```

```
@===================================#
#

x <- 1:10
y <- runif(length(x), min = 5, max = 6)

df1 <- data.frame(x, y)

xx <- 1:9
yy <- runif(10, min = 5, max = 6)

try(df2 <- data.frame(xx, yy)) # Won't work since 'xx' and 'yy' are of
different lengths

#
#=================@ Getting data into R
@===================================#
#

nba <- read.csv("/home/greg/Dropbox/My Stuff/Economic Forecasting
2013/nbasalary.csv", header=TRUE)

#
# Adding a variable
#

nba$points.sq <- nba$points^2

#
#==================@ Running a regression
@==================================#
#

reg <- lm(wage ~ points, data=nba) # Creates an object 'reg' which
contains the regression of wage on points and includes an intercept
summary(reg) # Detailed summary of the above regression

res <- resid(reg) # Creates an object 'res' which is the residuals of
the object 'res'

#
#==================@ Plotting
@===============================================#
#

plot(res)

plot(nba$points, nba$wage) # bad
plot(nba$points, nba$wage,
    main="Scatter plot of wages and points", xlab="points",
ylab="wage", col="blue")

hist(nba$wage, main="Histogram of Wage", xlab="wage")
plot(density(res))

#
#=================@ Loops
@==================================================#
#

x <- rep(NA,10) # creating an empty x vector with 10 empty slots
x[1] <- 1 # putting a '1' in the first element

for (t in 2:10) x[t] <- 2*x[t -1]

View(x)

#
#=================@ Functions
@==============================================#

fun.sq <- function(x){x^2}

# Creates a function 'fun.sq' that takes one argument, 'x', and
returns the
# square of the argument

fun.sq(2)

fun.add <- function(x, y){x + y}

# Creates a function 'fun.add' that takes two arguments, 'x' and 'y',
and
# returns the sum of the two arguments

fun.add(1:2, 3:4)

#
#=================@ The Shift Function
=====================================#
#

# This function is very useful for creating lag, or lead, variables
for a
# data.frame object. Use negative numbers for the argument 'shift_by'
to
# create lag variables and positive numbers for the argument
'shift_by' to
# create lead variables.
#
```

```
# Shift function from r-bloggers.com
# http://www.r-bloggers.com/generating-a-laglead-variables/

shift<-function(x,shift_by){
  stopifnot(is.numeric(shift_by))
  stopifnot(is.numeric(x))

  if (length(shift_by)>1)
    return(sapply(shift_by,shift, x=x))

  out<-NULL
  abs_shift_by=abs(shift_by)
  if (shift_by > 0 )
    out<-c(tail(x,-abs_shift_by),rep(NA,abs_shift_by))
  else if (shift_by < 0 )
    out<-c(rep(NA,abs_shift_by), head(x,-abs_shift_by))
  else
    out<-x
  out
}

x <- rnorm(10)
y <- runif(10)

df.xy <- data.frame(x, y)

df.xy$x.lag <- shift(df.xy$x, -1)

View(df.xy)

#=================@ Turning a series into a time series
@====================#

n <- 100

x <- rnorm(n)

plot(x)

x <- ts(x) # transforms 'x' into a time series object

plot(x) # Much better default plot for a time series

x.lag <- lag(x)
class(x.lag)

df3 <- data.frame(x, x.lag)
View(df3) # Bad, use the shift function instead
```