# Simple Linear Regression in R

Oct. 2019 STAT 206

Fitting a simple linear regression model may seem easy and intuitive. But when the data set is large, doing the calculation manually gets tedious and may become impractical. Here, we will learn how we can use R to make things a lot easier.

## 1 Reading Data into R

Lets consider the Copier maintenance problem in the textbook (problem 1.20). X denotes the number of copiers serviced. And Y denotes the number of minutes spent by the service person. We have data on 45 locations. And we want to see if our simple regression model is appropriate.

1. Save the data file (Copier.txt) to your local drive and remember the location of your file.
2. Open R. In R console, type in the following command.
   **copier=read.table("Copier.txt",header=FALSE)**

We have now defined a data frame called "copier" and imported the dataset saved in "Copier.txt" into this data frame. Notice here "header=FALSE" means that in the text file "Copier.txt", there is no variable name in the "header" (the first row is data instead of variable name). If the text file contained variable names, we would have changed to "header=TRUE". Here we could look at the summary of the data set using summary function in R.

```
> summary(copier)
       V1               V2        
 Min.   :  3.00   Min.   : 1.000  
 1st Qu.: 36.00   1st Qu.: 2.000  
 Median : 74.00   Median : 5.000  
 Mean   : 76.27   Mean   : 5.111  
 3rd Qu.:111.00   3rd Qu.: 7.000  
 Max.   :156.00   Max.   :10.000  
```

## 2 Running Simple Linear Regression Using "lm"

Now that data was read into R, we can see the two columns are automatically named "V1" and "V2" respectively. V1 is number of minutes spent by service person (our Y). And V2 is the number of copier machines (our X). Type in the following command:
**fit=lm(V1∼ V2, data=copier)**
We have fit the model $V1 = \beta_0 + \beta_1 V2 + \epsilon$ to the data and saved the results in "fit". In lm function, the response variable should be on the left side of the formula, and the predictor(s) should be on the right of the formula. To see the summary of the least square fit, type in the following
**summary(fit)**
Here is a screenshot of the summary.

```
> fit = lm(V1~V2,data=copier)
> summary(fit)

Call:
lm(formula = V1 ~ V2, data = copier)

Residuals:
     Min       1Q   Median       3Q      Max
-22.7723  -3.7371   0.3334   6.3334  15.4039

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.5802     2.8039  -0.207    0.837
V2           15.0352     0.4831  31.123   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 8.914 on 43 degrees of freedom
Multiple R-squared:  0.9575,    Adjusted R-squared:  0.9565
F-statistic: 968.7 on 1 and 43 DF,  p-value: < 2.2e-16
```

The output tells us that the estimate of the intercept $\beta_0$ is -0.5802. Its standard error is 2.8039. The estimate of the slope $\beta_1$ is 15.0352 with a standard error of 0.4831. Our fitted model is $Y = -0.5802 + 15.0352 * X$.

The "t value" provides us with observed test statistics for testing whether the effect of each predictor is significant or not and the " Pr(¿—t—)" is its corresponding p value. For example, for predictor V2, the p value is less than 2e-16 which means the effect of V2 is significant under level 0.01.

The "Multiple R-squared" in the output is our coefficient of determination, which is 0.9575 here. The "Residual standard error" in the output gives the square root of MSE. So the MSE=$8.9142^2$=79.459. With the information given in the output above, one should be able to construct confidence intervals for $\beta_0$ and $\beta_1$. For example, a 95% confidence interval for $\beta_1$ is

$$\hat{\beta}_1 \pm t(1-\alpha/2; n-2) * s\{\hat{\beta}_1\} = 15.0352 \pm t(0.975; 43) * 0.4831 = 15.0352 \pm 0.9743.$$

(Question: why the multiplier is t(0.975; 43)? do you know how to get t(0.975; 43) in R.

To see the estimates, residuals or fitted values we can use the following codes.

```
coef(fit)
fit$coefficients

residuals(fit)
fit$residuals

fitted(fit)
fit$fitted

fitted(fit)[5] # fifth fitted value
residuals(fit)[5] # fifth residual
```

To see the **anova** table we can use the **anova** function.

```
> anova(fit)
Analysis of Variance Table
```

```
Response: V1
          Df Sum Sq Mean Sq F value    Pr(>F)
V2         1  76960   76960  968.66 < 2.2e-16 ***
Residuals 43   3416      79
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

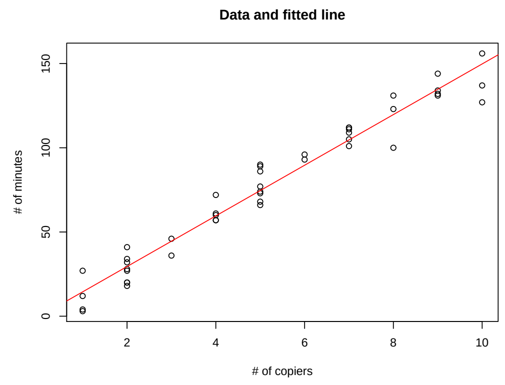
(Question: What is the relation between "F value" in the anova table and "t value" in summary fit of V2.)

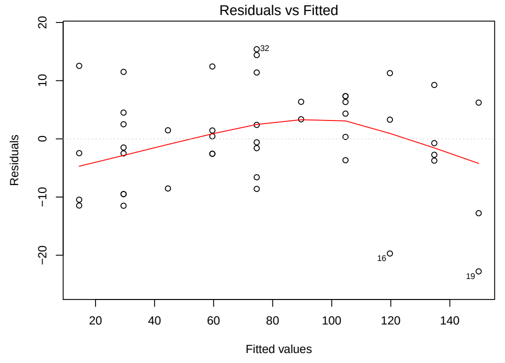
## 3 Plotting

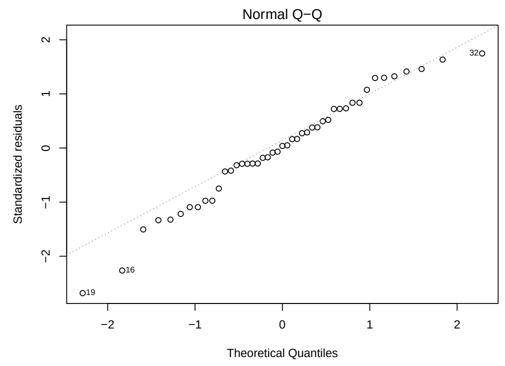
We can also make plots to have a visual understanding of our fits and do diagnostics. When using the plot function, the first entry is for the predictor variable X. And the second entry is for the response variable Y. "xlab" and "ylab" in plot function specify the labels for the horizontal and vertical axis, respectively. The "abline" function adds a straight line to an existing plot.

```
par(mfrow=c(2,2))  #plot several figures in one panel

plot(copier$V2,copier$V1,xlab='# of copiers',ylab='# of minutes',
                                      main='Data and fitted line')

abline(fit, col='red') #adding the fitted line (red)

plot(fit,which=1) # plot the residuals, which=1 means plot against the
                  # fitted value
plot(fit,which=2) #plot the normal-qq plot of the residuals
```

Then we have the following plots. If the model is adequate, the "residual vs fitted" plot should show no obvious nonlinear pattern. Moreover, the spread of the residuals should be roughly the same across the horizontal axis. If the errors are normally distributed, the points should be roughly along the diagonal line in the qq plot.







## 4 Finding Confidence and Prediction intervals

You can make prediction and compute confidence intervals for new data using the "predict" function. We can see a prediction interval for a new observation is wider than the corresponding confidence interval for the mean response as we expected.

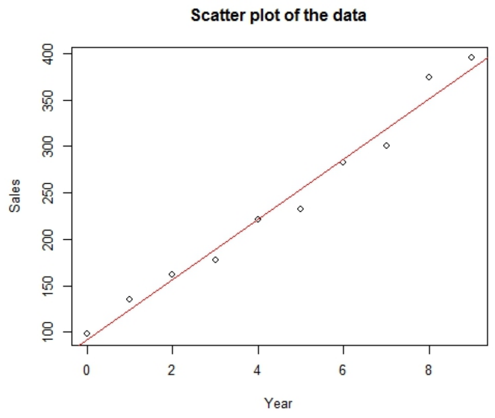
```
> newX=data.frame(V2=5)
> # interval specifies the type of intervals you want to obtain
> # level specifies the confidence level
> predict(fit, newX, interval='confidence', level=0.95)
       fit      lwr      upr
1 74.59608 71.91422 77.27794
> predict(fit, newX, interval='confidence', level=0.99)
```

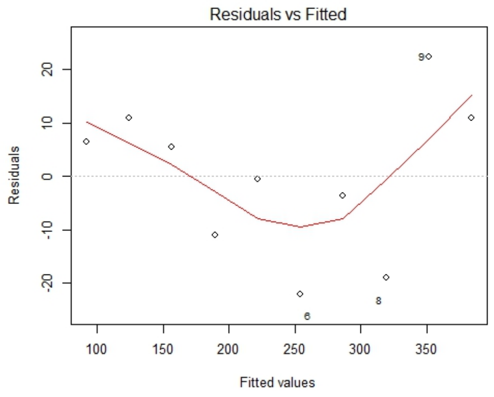
```
       fit      lwr      upr
1 74.59608 71.01205 78.18011
> predict(fit, newX, interval='prediction', level=0.95)
       fit      lwr      upr
1 74.59608 56.42133 92.77084
> predict(fit, newX, interval='prediction', level=0.99)
       fit      lwr      upr
1 74.59608 50.30738 98.88478
```

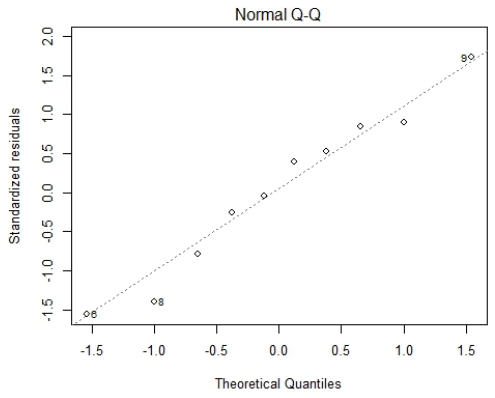
(Question: Why is the prediction interval wider than the corresponding confidence interval)

## 5 Box-Cox Transformation

Let's consider the Sales growth problem in the textbook (problem 3.17). A marketing researcher studied annual sales of a product that had been introduced 10 years ago. X is the year (coded) and Y is sales in thousands of units.
A scatterplot of the data suggest that a linear relation appear adequate here. However, if we fit a simple regression model to the data, the diagnostic plots will display a warning sign. There is apparently a nonlinear trend in the residual vs. fitted plot.



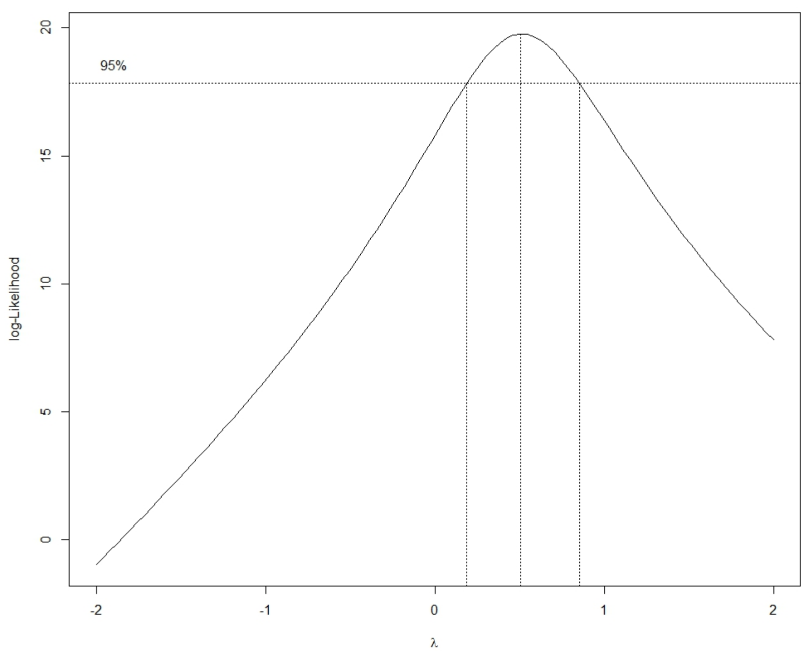




We can use Box-Cox procedure to find an appropriate transformation. Box-Cox transformation is in MASS (math) library of R. Type in the following:

```
library(MASS)
boxcox(V1~V2, data=sales)
```

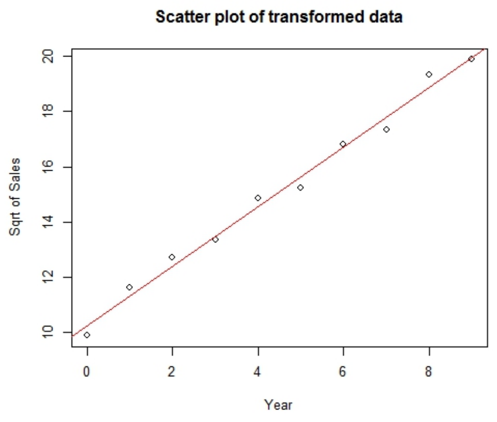
The following plot will pop up. The vertical axis of the plot is the loglikelihood based on different transformations on Y. The horizontal axis is the value of the power of Y. The transformation is $Y_{\text{new}} = Y^{\lambda}$.
We can see the log-likelihood is largest when $\lambda$ is around 0.5, which means the square root transformation is appropriate.

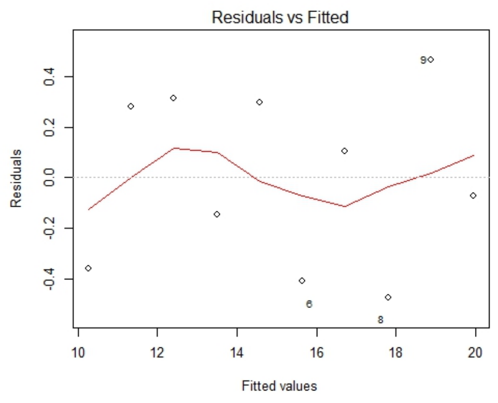


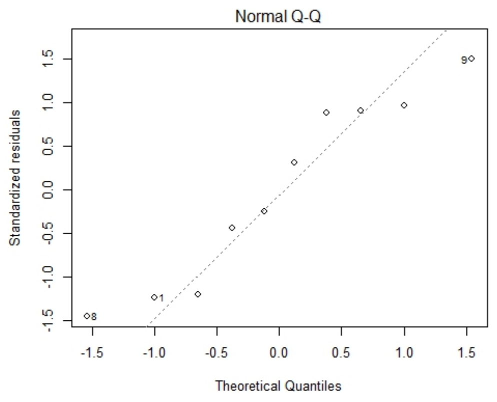
Now let's use square root transformation and see if it improves the fit.

```
newfit_sales=lm(V1^(1/2)~V2,data=sales)
```

The multiple R-squared increased from 0.9798 to 0.9891.
The following is a scatter plot of the transformed data with fitted regression line in red. The diagnostic plots suggest that conformity to assumptions has been improved.







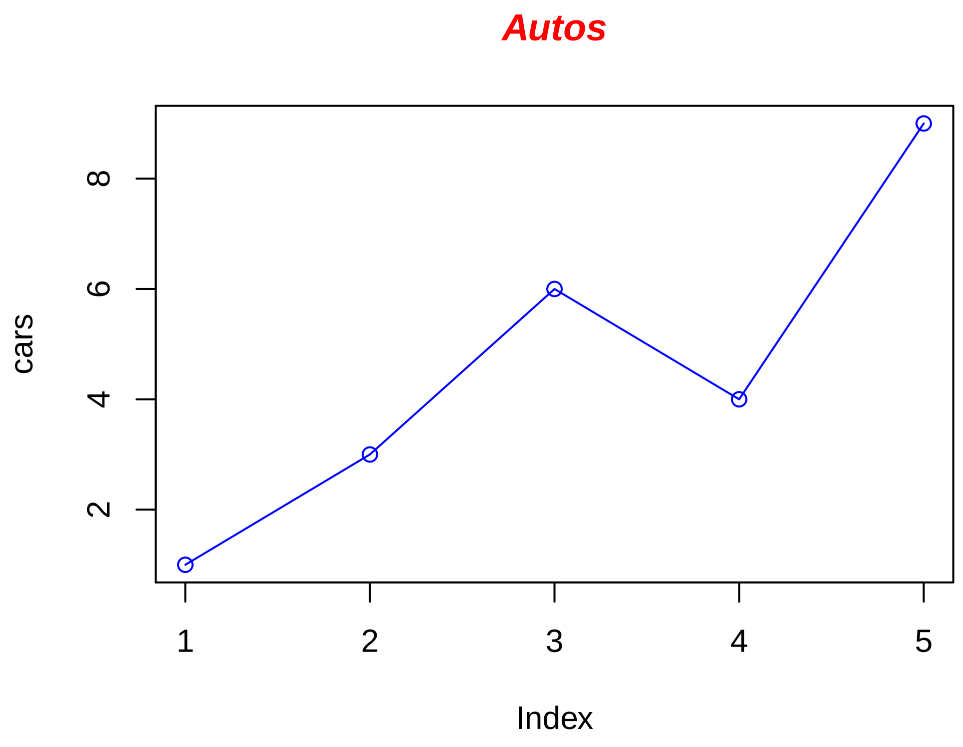
## 6 Saving the Results

To save the least squares model, simply type the following and replace "name" by a file name easy to remember
**save(fit, file="name.RData")**
To load this saved file in the future, you need to open R console first, then double click on this file. Or, you can also type
**load("name.RData")**
You can also save the entire R space by
**save.image(file="name.RData")**

## 7 More about R plotting

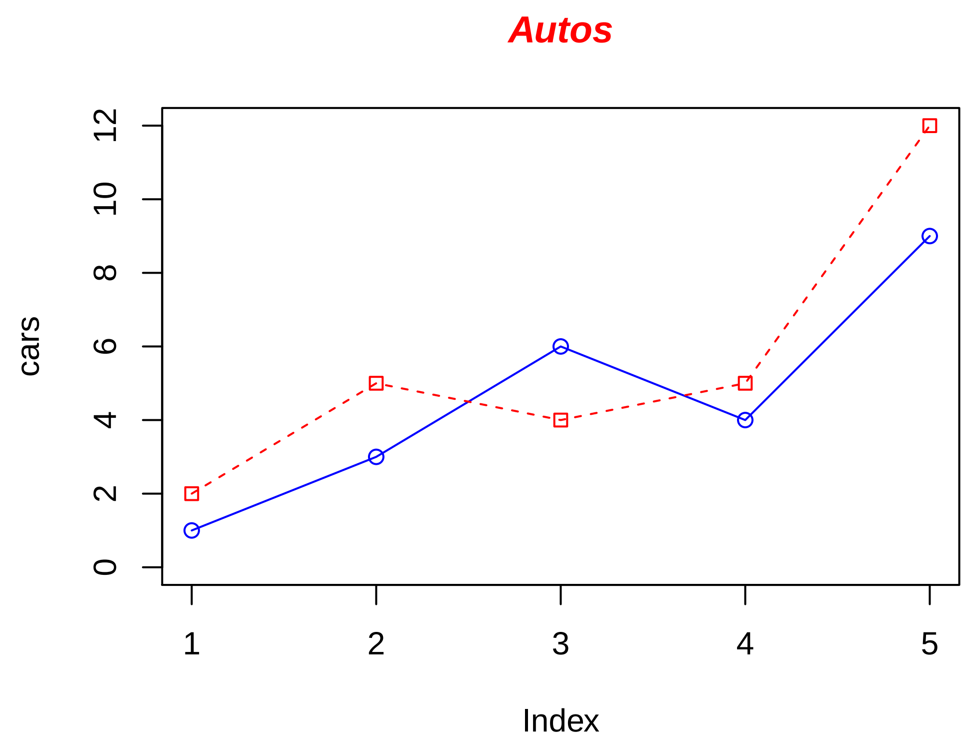
Here we go into more details about R plotting. More materials can be found in http://www.harding.edu/fmccown/r/.
First we'll produce a very simple graph using the values in the car vector. Let's add a title, a line to connect the points, and some color:

```
# Define the cars vector with 5 values
cars <- c(1, 3, 6, 4, 9)

# Graph cars using blue points overlayed by a line
plot(cars, type="o", col="blue")

# Create a title with a red, bold/italic font
title(main="Autos", col.main="red", font.main=4)
```



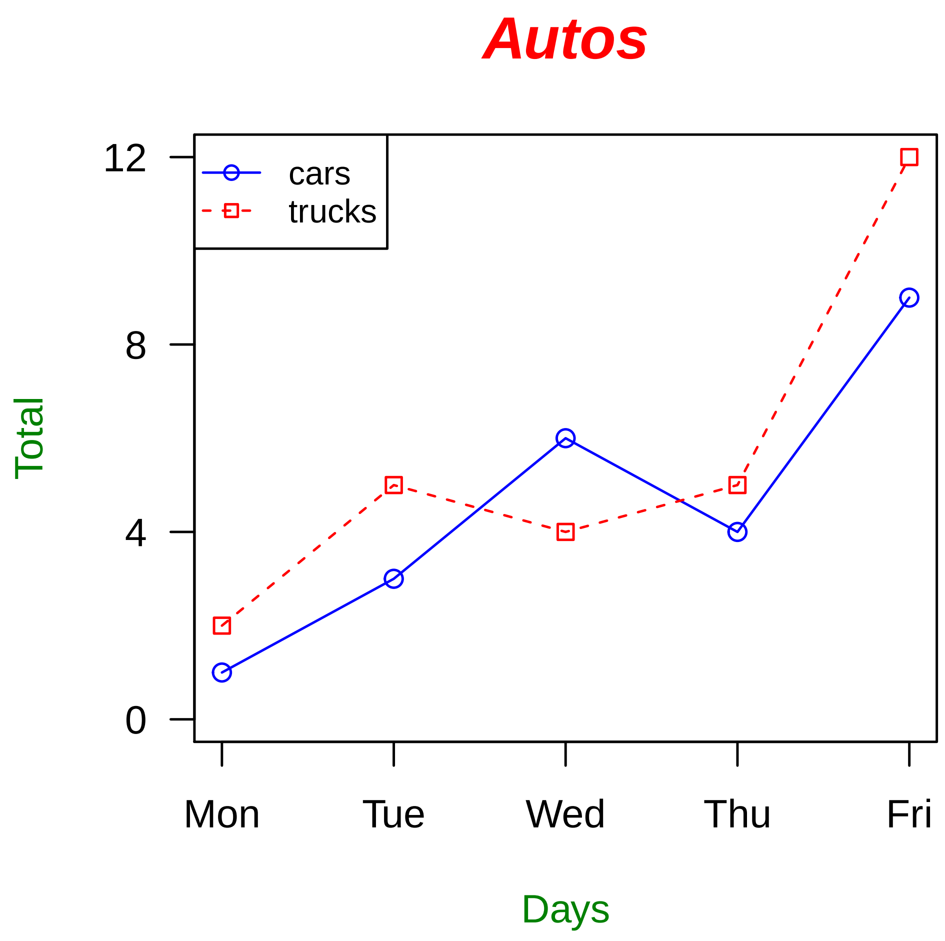
Now let's add a red line for trucks and specify the y-axis range directly so it will be large enough to fit the truck data:

```
# Define 2 vectors
cars <- c(1, 3, 6, 4, 9)
trucks <- c(2, 5, 4, 5, 12)

# Graph cars using a y axis that ranges from 0 to 12
plot(cars, type="o", col="blue", ylim=c(0, 12))

# Graph trucks with red dashed line and square points
lines(trucks, type="o", pch=22, lty=2, col="red")

# Create a title with a red, bold/italic font
title(main="Autos", col.main="red", font.main=4)
```



Next let's change the axes labels to match our data and add a legend. We'll also compute the y-axis values using the max function so any changes to our data will be automatically reflected in our graph.

```
# Define 2 vectors
cars <- c(1, 3, 6, 4, 9)
trucks <- c(2, 5, 4, 5, 12)

# Calculate range from 0 to max value of cars and trucks
g_range <- range(0, cars, trucks)

# Graph autos using y axis that ranges from 0 to max
# value in cars or trucks vector.  Turn off axes and
# annotations (axis labels) so we can specify them ourself
plot(cars, type="o", col="blue", ylim=g_range,
axes=FALSE, ann=FALSE)

# Make x axis using Mon-Fri labels
axis(1, at=1:5, lab=c("Mon","Tue","Wed","Thu","Fri"))

# Make y axis with horizontal labels that display ticks at
# every 4 marks. 4*0:g_range[2] is equivalent to c(0,4,8,12).
axis(2, las=1, at=4*0:g_range[2])

# Create box around plot
box()

# Graph trucks with red dashed line and square points
lines(trucks, type="o", pch=22, lty=2, col="red")

# Create a title with a red, bold/italic font, and slightly larger
title(main="Autos", col.main="red", font.main=4, cex.main=1.5)

# Label the x and y axes with dark green text
title(xlab="Days", col.lab=rgb(0,0.5,0))
title(ylab="Total", col.lab=rgb(0,0.5,0))

# Create a legend at topleft that is slightly smaller
# (cex) and uses the same line colors and points used by
# the actual plots
legend("topleft", c("cars","trucks"), cex=0.8,
col=c("blue","red"), pch=c(21,22), lty=c(1,2))
```



## 8 Density, cumulative probability, percentile and random generator of distributions in R

We have very convenient functions to use in R to obtain density, cumulative probability, percentile and even generate random numbers according to different distributions.

```
##### density, cumulative probability, quantile and random generator
##### of basic distributions

# d=density, p=cumulative probability, q=quantile, r=random generator
# norm=normal distribution, t=t distribution, f=f distribution,
# chisq=chi-square distribution
dnorm(0,mean=0,sd=1)
pnorm(0,mean=0,sd=1)
qnorm(0.975,mean=0,sd=1)
rnorm(5,mean=0,sd=1) #generate 5 standard normal random variable

test1 = rnorm(10,mean=0,sd=1)
## rnorm used to get a sample from a normal distribution.
## 10 is the sample size
mean(test1)
sd(test1)

test2 = rnorm(10000,mean=0,sd=1)
mean(test2)
sd(test2)

qt(0.995,43) #0.995 quantile of t distribution

pf(20,1,43) #cumulative probability of a F_1,43 distribution at 20

dchisq(1.2,43) #density of chi-square distribution of d.f. 43 at 1.2
qchisq(0.975,43) #quantile of chi-square distribution of d.f. 43 at probability 0.
```

## 9 R help

If you have questions about how to use R functions, you can always type "?function name" to get the documentation that explains the function. For example, you can try "?lm". In most cases you can find a lot of material using google too.