Module 9. Regression Analysis.

Overview:

This module introduces statistical methods of studying relationship among variables. The regression procedure is introduced where information about independent variable(s) X is used to predict a response variable Y. We introduce the concepts of linear correlation coefficient, and the linear regression model. We teach how to fit the linear regression model and how to check its model assumptions.

By the end of this module, you should be able to conduct linear regression analysis in R, and able use the R outputs to make appropriate statistical inferences. You should also know how to use diagnostic tools to check the regression model.

Learning Objectives

1. Calculate the **correlation coefficient**.
2. Fitting and interpreting the **linear regression model**.
3. Checking the linear regression model assumptions.

Readings:

Seefeld & Linder's book pages 275-286.
Krijnen's book pages 130-133.

Lesson 1: Correlation measures

Objectives

By the end of this lesson you will have had opportunity to:

- Calculate the **linear correlation coefficient**
- Carry out the **t-test** and **permutation test** for zero linear correlation coefficient
- Calculate the **bootstrap confidence interval** for the linear correlation coefficient

Overview

We study the relationship between two continuous random variables in this module. We first consider a numerical summation of the relationship between two random variables.

Linear correlation coefficient

Pearson's linear correlation coefficient is the most commonly used numerical summary for the strength of associate between the two random variables. The population linear correlation coefficient between two random variables X and Y is defined as $\rho(X,Y)=\frac{Cov(X,Y)}{\sqrt{V\text{ar}(X)Var(Y)}}$ where $Cov(X,Y)=E[(X-EX)(Y-EY)]$. Recall from earlier modules on probability, E(X) denotes the expectation of the random variable X, and the variance is defined as $Var(X)=E[(X-EX)^2]$.

When we have a random sample of paired observations on the two random variables $(X_1,Y_1),...,(X_n,Y_n)$, then the sample linear correlation coefficient is

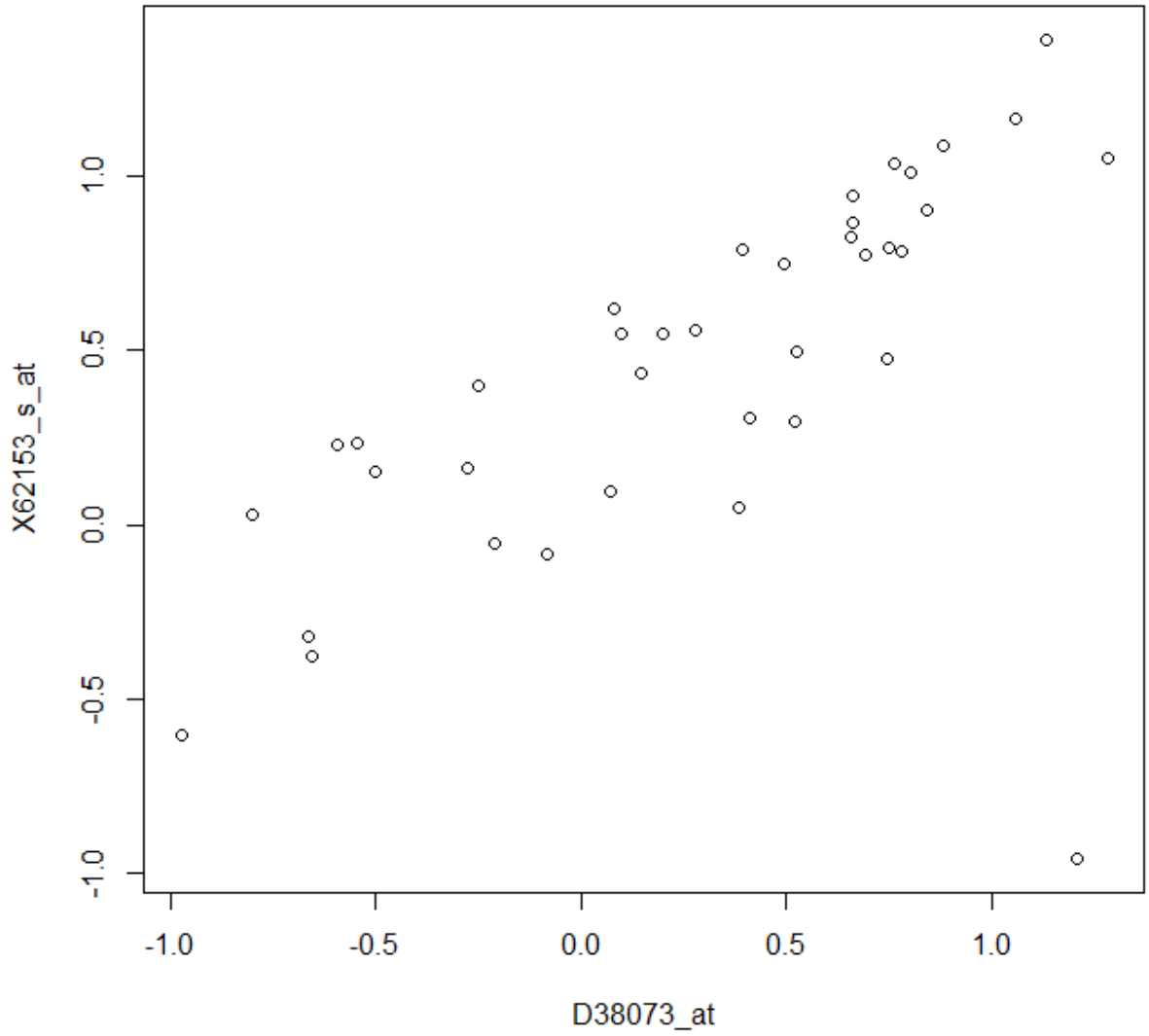
$$\hat{\rho}=\frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}.$$

Given a sample, $\hat{\rho}$ reflects the strength of linear relationship between the two random variables. Often we are interested to test if there is a linear relationship between X and Y. That is to test $H_0:\rho=0$ versus $H_A:\rho\neq 0$. Under the null hypothesis (ρ=0) with normally distributed X and Y, we can show that $t_{obs}=\hat{\rho}\sqrt{\frac{n-2}{1-\hat{\rho}^2}}$ asymptotically follows a t-distribution with n-2 degrees of freedom. Hence we reject the null hypothesis of no relationship when $|t_{obs}|>t_{1-\alpha/2,n-2}$.

The calculation of the linear correlation coefficient and the corresponding hypothesis testing are done in R by cor() and cor.test(). We illustrate them in the next example.

Demonstration: correlation of gene expression values from two probes.
Example 1. There are two sets of expression values of the MCM3 gene in the Golub et al. (1999) data. This gene encodes for highly conserved mini-chromosome maintenance proteins (MCM) which are involved in the initiation of eukaryotic genome replication. We found its row numbers at 2289 and 2430 with grep(). We will display the values from these two rows in a scatterplot and calculate the linear correlation coefficient.

```
> data(golub,package="multtest") #Load golub data from multtest package
> x <- golub[2289,] #MCM3 gene probeset1
> y <- golub[2430,] #MCM3 gene probeset2
> plot(x,y,xlab=golub.gnames[2289,3],ylab=golub.gnames[2430,3]) #scatterplot
> cor(x,y) #calculate correlation
[1] 0.6376217
```



Example 1. (Cont'd)

From the plot of data, we see a clear linear trend between the measurements from these two probes. We can further test for the existence of linear relationship by

```
> cor.test(x,y) #test if true correlation=0
        Pearson's product-moment correlation
data:  x and y
t = 4.9662, df = 36, p-value = 1.666e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3993383 0.7952115
sample estimates:
      cor
0.6376217
```

Since the p-value = 1.666e-05 is very small, we reject the null hypothesis $H_0 : \rho = 0$. That is, there is strong evidence that the two sets of expression values are correlated. This agrees with the pattern we see from the plot.

Here the sample size n=38 is moderately large enough, so the asymptotic tests may be used. However, we can also try nonparametric methods here. We will illustrate the permutation test and bootstrap confidence intervals on this example next.

Example 1. (Cont'd) Permutation test for correlation.

The permutation is done under the null hypothesis H_0 that X and Y are independent. Therefore, under the null hypothesis, X can be permuted without regard to Y. We can modify our permutation test code to calculate the linear correlation coefficients as below.

```
n<-length(x)  #sample size n = number of pairs
T.obs<- cor(x,y) #Observed statistic is the correlation rho
n.perm=2000  # We will permute 2000 times
T.perm = rep(NA, n.perm) #A vector to save the permuted statistics
for(i in 1:n.perm) {
    x.perm = sample(x, n, replace=F)   #permute data (x only)
    T.perm[i] = cor(x.perm, y) #Permuted statistic is the correlation
}
mean(abs(T.perm)>=abs(T.obs))  #p-value for 2-sided test
```

Notice for two-sided test, p-value= $P(|\hat{\rho}| \geq |\hat{\rho}_{obs}|)$ so that absolute values are needed.

Run the R code and we get

```
[1] 0
```

The p-value is zero. This agrees with the small p-value = 1.666e-05 from the cor.test().

For the nonparametric permutation test, we only used n.perm=2000 since we are not trying to be accurate after the third decimal space.

Example 1. (Cont'd) Bootstrap confidence interval for correlation.

We may also try to get nonparametric bootstrap confidence interval for the linear correlation coefficient. Notice that for this resampling, we need to keep the dependency structure. That is, we need to resample the whole pair (X_i, Y_i) together.

```
nboot <- 2000 # We will resample 2000 times
boot.cor <- matrix(0,nrow=nboot, ncol = 1) #A vector to save the resampled statistics
data <- cbind(x,y) #Data set with x and y in two columns.
for (i in 1:nboot){
  dat.star <- data[sample(1:nrow(data),replace=TRUE), ] #Resample the pairs
  boot.cor[i,] <- cor(dat.star[,1], dat.star[,2]) #Correlation on resampled data
}
quantile(boot.cor[,1],c(0.025,0.975)) #Find quantiles for resampled statistics
```

Run the R code and we get

```
    2.5%     97.5%
0.2384436 0.9170563
```

Hence the 95% CI is (0.238,0.917), much wider than the (0.399,0.795) from the cor.test(). This indicates that the normality assumption for the cor.test() may not hold and n=38 is not sufficiently large for the asymptotic approximation to become valid. We will stick with the nonparametric methods here.

Comparing the permutation test and bootstrap confidence interval

We have two nonparametric procedures based on resampling. Notice that they differs in two aspects of resampling: (1) The bootstrap method resamples the pairs together while permutation test resamples X and Y separately; (2) bootstrap method resamples with replacement, and the permutation test resamples without replacement.

The reason for (1) is that bootstrap CI tries to bound the linear correlation coefficient. To do this, the resampled data set should have similar dependence structure as the original data set. In contrast, the permutation test require the resampled data set to have no dependence structure (assumption of the null hypothesis). That way, we can compare how extreme the observed statistic is under the null distribution.

The reason for (2) is that bootstrap methods tries to find the randomness in the statistic through resampled data sets. If we resample without replacement, bootstrapped data sets would always be the same as the original data set. On contrary, the permutation test is aiming at capturing the null distribution of the statistic. We need the marginal distributions of X and Y remain exactly the same as the original data set. We just want to see resampled data sets without any dependence structure in it.

Summary

In this lesson, we introduced the linear correlation coefficient ρ between two variables X and Y.

You should know how to calculate the sample linear correlation coefficient from the data set using R, and how to conduct parameter and nonparametric tests for $H_0 : \rho = 0$. We also discussed the bootstrap confidence interval.

Next, we introduce the simple linear regression model fitting.

Lesson 2: Simple Linear Regression

Objectives

By the end of this lesson you will have had the opportunity to:

- Fit the simple linear regression model
- Make statistical inference on the parameter estimates
- Get prediction intervals and confidence intervals of the response variable
- Distinguish the regression model and ANOVA model

Overview

We use one (independent) variable X to predict the response variable Y. We first set up the linear regression model, and derive the point estimates from probability theory. We then consider the confidence intervals and hypothesis tests on the parameters.

We introduce the concept of prediction intervals versus confidence intervals for predicting the response variable Y given the independent variable X's value. We then study the ANOVA analysis of the regression model. We teach the usage of R-squared versus F-test. Finally, we distinguish the fitting of regression model and ANOVA model in R.

Linear regression model

When we have a random sample of $(X_1, Y_1), ..., (X_n, Y_n)$, the simple linear regression model assumes

$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$, for $i = 1, ..., n,$

where β_0 and β_1 are unknown parameters, and $\varepsilon_1, ..., \varepsilon_n$ are random noise $\sim N(mean = 0, sd = \sigma)$.

Notice that this is the **linear model** form we introduced in the last module. For statistical inferences, we assume that $X_1, ..., X_n$ are known and only consider the randomness from the noise $\varepsilon_1, ..., \varepsilon_n$. Another way statisticians treat this model is that, assume $X_1, ..., X_n$ are random variables, we make inferences conditional on their observed values. Either way, the essential point is that we do not make distributional assumptions on $X_1, ..., X_n$, we only assume the normal distribution on noise $\varepsilon_1, ..., \varepsilon_n$.

The main object in the linear regression analysis is to estimate the parameters β_0 and β_1. These parameters allows us to describe the relationship between X and Y: $E(Y) = \beta_0 + \beta_1 X$, thus allowing prediction of Y given an observed X value.

We derive the point estimators for the parameters next, using probability theory.

Following convention, we use the lower case letters to denote the observed values $(x_1, y_1), ..., (x_n, y_n)$. So we first derive the formula in terms of these values.

Parameter estimates for linear regression model

We derive, for linear regression model, the maximum likelihood estimate (MLE). (Review module 5 if you forgot about MLE).

From the normal density definition, we see that the likelihood (joint normal density in probability theory) is

$$L = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y_1-\beta_0-\beta_1 x_1)^2}{2\sigma^2}} \ldots \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y_n-\beta_0-\beta_1 x_n)^2}{2\sigma^2}} .$$

To maximize L, we equivalently maximize the log-likelihood

$$l = \log L = \sum_{i=1}^{n} [\frac{-(y_i-\beta_0-\beta_1 x_i)^2}{2\sigma^2} - \log(\sqrt{2\pi}\sigma)].$$

To maximize the log-likelihood, clearly the estimates for β_0 and β_1 have to minimize $\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2$. Therefore, the MLE for β_0 and β_1 is also called ***least squares estimators*** since they minimize $\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2$.

Setting the derivatives of against β_0 and β_1 to zeroes, we can solve for

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2} \quad \text{and} \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} .$$

We then make statistical inferences based on these estimates. For any new x value, we estimate the corresponding y value by $\hat{\beta}_0 + \hat{\beta}_1 x$.

There is one more parameter in the model, σ, which is not of main interest. But we will need it to judge how big the noise is, and it is used in testing the β parameters. Setting derivative of the log-likelihood against σ to zero, we found the MLE $\hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_i)^2$. Recall that MLE is the best estimator asymptotically (sample size n increases to ∞). For finite sample, we generally use an unbiased estimator

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_i)^2 .$$

The fitting of the linear regression model is done by the lm() function in R, since the linear regression model has the linear model representation.

Demonstration: Regression on gene expression values from two probes.

Example 1. For the expression values of the MCM3 gene at two probe positions in the Golub et al. (1999) data, we use 'D38073_at' measurements to predict the 'X62153_s_at' measurements. That is, regression X62153_s_at on D38073_at.

```
> D38073_at <- golub[2289,]
> X62153_s_at <- golub[2430,]
> reg.fit<-lm(X62153_s_at ~ D38073_at) #Regression X62153_s_at = b0+
b1*D38073_at
> reg.fit   #Results of the regression fit
Call:
lm(formula = X62153_s_at ~ D38073_at)
Coefficients:
(Intercept)    D38073_at
     0.2996       0.5433
```

This means $\hat{\beta}_0 = 0.2996$ and $\hat{\beta}_1 = 0.5433$ so that the estimated regression equation is

X62153_s_at = 0.2996 +0.5433*D38073_at

For example, if the D38073_at expression value is 0.5, we would expect the X62153_s_at expression value on the same patient is
0.2996+0.5433*(0.5)=0.57125

These are the point estimations. We will need to also do statistical inferences including confidence intervals and hypothesis test for the regression model. Next, we consider how to do inferences, again based on the probability theory.

Statistical inferences in linear regression model

The regression model aims to inference Y based on knowledge of X. Hence we assumed $X_1,...,X_n$ as known constants while $Y_1,...,Y_n$ follow the normal distribution due to the normal random noises $\varepsilon_1,...,\varepsilon_n$. Recall from the modules on probability theory, linear combinations of normal random variables are still normal random variables. We notice that the points estimators

$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})y_i}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$ and $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}$ are both linear combinations of $y_1,...,y_n$.

Hence, $\hat{\beta}_0$ and $\hat{\beta}_1$ are also normally distributed. Therefore, we can apply the t-tests and t-intervals similar to previous modules with their standard error proportional to $\hat{\sigma} = s$. That is, the (1-α) confidence interval for $\hat{\beta}_0$ and $\hat{\beta}_1$ are respectively $\hat{\beta}_0 \pm t_{1-\alpha/2,n-2}\cdot se(\hat{\beta}_0)$ and $\hat{\beta}_1 \pm t_{1-\alpha/2,n-2}\cdot se(\hat{\beta}_1)$. The hypothesis tests are done with corresponding t-tests. For example, to test $H_0:\beta_1 = b$ versus $H_A:\beta_1 \neq b$, we will reject the null hypothesis if $|\frac{\hat{\beta}_1 - b}{se(\hat{\beta}_1)}| > t_{1-\alpha/2,n-2}$.

The quantities needed in these inferences are already computed by the lm() fit. We can use summary() to see them.

Regression on gene expression values from two probes (continued)

```
reg.fit<-lm(X62153_s_at ~ D38073_at) #Regression fit
> summary(reg.fit)        #summary of regression results
Call:
lm(formula = X62153_s_at ~ D38073_at)
Residuals:
    Min      1Q  Median      3Q     Max
-1.9137 -0.2239  0.1139  0.2350  0.4749
Coefficients:
             Estimate     Std. Error t value Pr(>|t|)
(Intercept)     0.29958    0.07258   4.127  0.000208 ***
D38073_at       0.54326    0.10939   4.966  1.67e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.4029 on 36 degrees of freedom
Multiple R-squared:  0.4066,    Adjusted R-squared:  0.3901
F-statistic: 24.66 on 1 and 36 DF,  p-value: 1.666e-05
```

The standard errors $se(\hat{\beta}_0)$ and $se(\hat{\beta}_1)$ are provided right after the point estimates. Also, the summary provides the two-sided t-statistics and two-sided p-values for testing $H_0 : \beta_0 = 0$ and $H_0 : \beta_1 = 0$. The p-values of **0.000208** and **1.67e-05** are very small, so that we conclude both the intercept β_0 and the slope β_1 are nonzero.

To get the 90% confidence intervals for β_0 and the β_1, we can use the confint() function.

```
> confint(reg.fit, level=0.9) #Show 90% 2-sided CIs from regression fit
                  5 %       95 %
(Intercept)  0.1770347 0.4221227
D38073_at    0.3585766 0.7279465
```

Hence the 90% confidence intervals for the intercept β_0 is (0.177, 0.422), and the 90% confidence intervals for the slope β_1 is (0.359, 0.728).

Inference of Y given X value

The main goal of the regression analysis is to infer about the response variable Y using knowledge of the independent variable X. We already derived the point prediction of y given x is $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$. Since $\hat{\beta}_0$ and $\hat{\beta}_1$ are both linear combinations of $y_1, ..., y_n$, so does $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$. Therefore, $\hat{\beta}_0 + \hat{\beta}_1 x$ also follows a normal distribution, and we can make inferences based on the corresponding t-statistic.

When we make interval inferences for $\hat{\beta}_0 + \hat{\beta}_1 x$, we have to distinguish between the **confidence interval (CI)** and the **prediction interval (PI).**
$\hat{\beta}_0 + \hat{\beta}_1 x \pm t_{1-\alpha/2, df=n-2} \cdot se(\hat{\beta}_0 + \hat{\beta}_1 x)$ is a 1-α confidence interval for $\beta_0 + \beta_1 x$. That is, in repeated trials, in 1-α proportion of data sets $\beta_0 + \beta_1 x$ falls into the intervals given by this formula.

However, notice $\beta_0 + \beta_1 x = E(Y \mid x)$ is the mean of Y values when $X = x$. In other words, this interval inference is for the mean of Y given x, $E(Y \mid x)$. Often, our objective is to predict Y itself. The above interval does not contain 1-α proportion of Y values when $X = x$. The interval that contains Y (given x value) 1-α proportion of times in repeated trials is called the prediction interval. A prediction interval for Y given x is given by
$\hat{\beta}_0 + \hat{\beta}_1 x \pm t_{1-\alpha/2, df=n-2} \cdot se(pred)$
where $se(pred) = \sqrt{[se(\hat{\beta}_0 + \hat{\beta}_1 x)]^2 + \hat{\sigma}^2}$. Here $\hat{\sigma}^2 = s^2 = \frac{1}{n-2} \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$ is the point estimate given earlier.

These intervals can be calculated directly by R using predict() function on the linear model fitting object obtained by lm() earlier. We illustrate them next.

Regression on gene expression values from two probes (continued)

We can use 'D38073_at' measurement to predict the 'X62153_s_at' measurements on the same patient. From the lm() fit earlier, we get the regression equation:

X62153_s_at = 0.2996 +0.5433*D38073_at

Therefore, for a patient with the D38073_at expression value 0.5, we would expect the X62153_s_at expression value on the same patient as

0.2996+0.5433*(0.5)=0.5712

This can be calculated automatically in R by

```
> reg.fit<-lm(X62153_s_at ~ D38073_at)
> predict(reg.fit, newdata=data.frame(D38073_at=0.5)) #point prediction for Y when X=0.5
        1
0.5712095
```

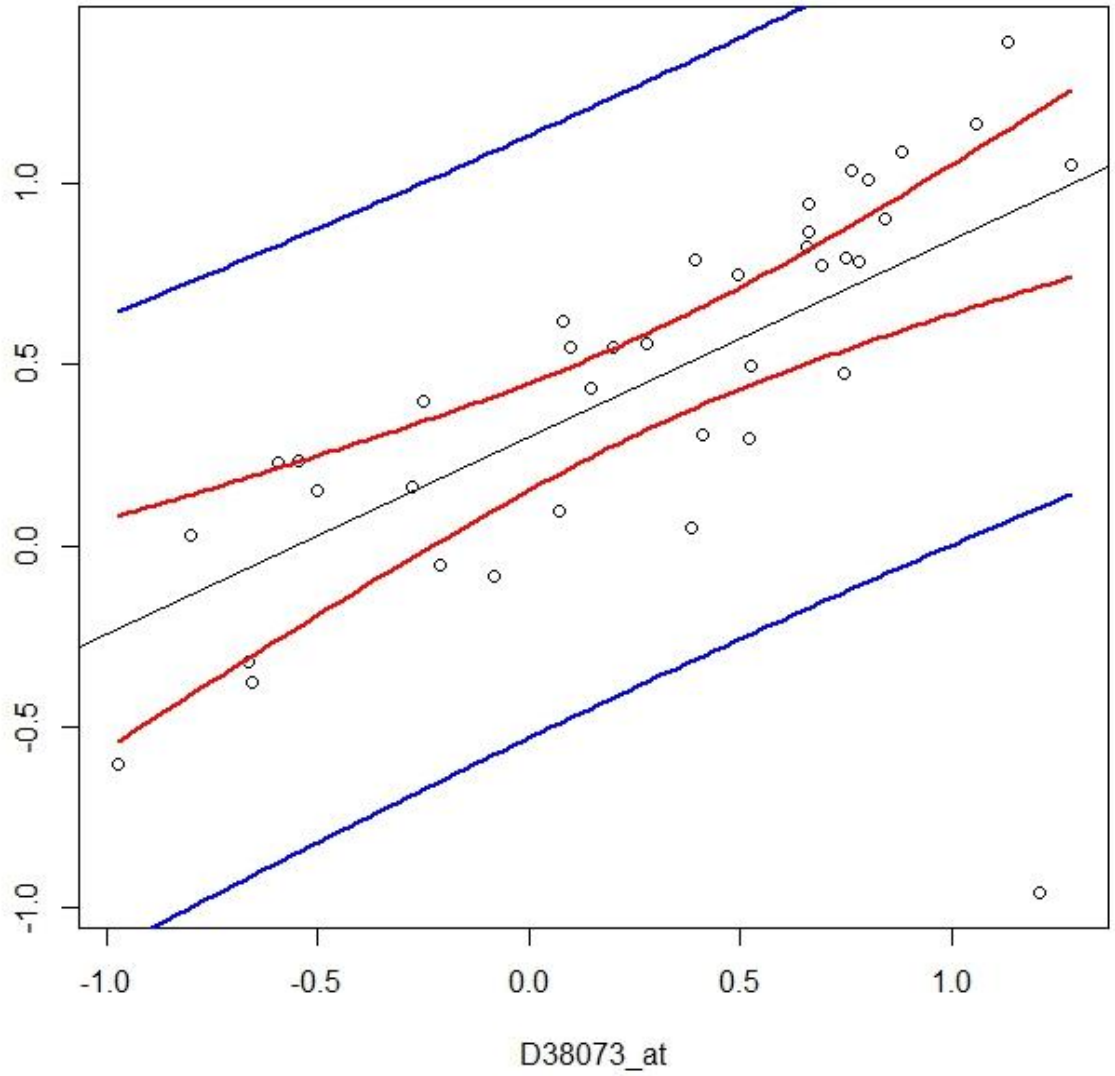
Furthermore, we can calculate the 95% confidence interval (CI) and 95% prediction interval (PI) as

```
> predict(reg.fit, newdata=data.frame(D38073_at=0.5), interval="confidence") #confidence interval for E(Y|X=0.5)
        fit       lwr       upr
1 0.5712095 0.4306099 0.7118091
> predict(reg.fit, newdata=data.frame(D38073_at=0.5), interval="prediction") #prediction interval for Y when X=0.5
        fit       lwr      upr
1 0.5712095 -0.2578912 1.40031
```

Therefore, the 95% CI for E(X62153_s_at | D38073_at=0.5) is (0.43,0.71) while the 95% PI for X62153_s_at given D38073_at=0.5 is (-0.26,1.40).

Notice that the 95% PI is much wider than the 95% CI, because the PI is expected to contain the X62153_s_at values of 95% patients with D38073_at value 0.5, and the CI does not. The CI is only aiming to contain the overall mean of X62153_s_at values among all patients with D38073_at value 0.5, not aiming to contain the individual X62153_s_at values that PI aims to contain.

Regression on gene expression values from two probes (continued)
We plot the data, fitted regression line, the 95% CIs (in red) and the 95% PIs (in blue) together. We can see that most data falls between the PIs but many are outside of the CIs.



We can also see that there seems to be an outlier at the right-lower corner of the plot. We will deal with the diagnostics of such outliers and other violations model assumption in another lesson later in this module.

The R commands for producing the plot

```
data(golub,package="multtest")
D38073_at <- golub[2289,]
X62153_s_at <- golub[2430,]
reg.fit<-lm(X62153_s_at ~ D38073_at) #Regression
predict(reg.fit, newdata=data.frame(D38073_at=0.5)) #point prediction for Y when
X=0.5
predict(reg.fit, newdata=data.frame(D38073_at=0.5), interval="confidence")
#confidence interval for E(Y|X=0.5)
predict(reg.fit, newdata=data.frame(D38073_at=0.5), interval="prediction")
#prediction interval for Y when X=0.5

plot(X62153_s_at ~ D38073_at) #Scatter plot
abline(reg.fit)                          #Regression line
xnew<-seq(min(D38073_at), max(D38073_at), length.out=100) #100 equally
spaced X values within range of X
conf<-predict(reg.fit,  newdata=data.frame(D38073_at=xnew),
interval="confidence")  #CIs at the 100 X values
conf.lower = conf[, "lwr"]
conf.upper = conf[,"upr"]
lines(xnew,conf.lower, lwd=2,col="red") #CI lower bounds plot
lines(xnew,conf.upper, lwd=2,col="red") #CI upper bounds plot
pred<-predict(reg.fit,  newdata=data.frame(D38073_at=xnew),
interval="prediction") #PIs at the 100 X values
pred.lower = pred[, "lwr"]
pred.upper = pred[,"upr"]
lines(xnew,pred.lower, lwd=2,col="blue") #PI lower bounds plot
lines(xnew,pred.upper, lwd=2,col="blue") #PI upper bounds plot
```

ANOVA analysis for regression

An important question in the linear regression is if the X really affects Y. To do this, we can also separate the variation in data into two components: regression variation and error variation. Then we can compares these two variance components through ANOVA analysis.

For the i-th observation in the data set, $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ is the predicted value of y_i. The regression variation is the variation in the regression prediction $SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$.

The error variation is the variation in the remaining errors $e_i = y_i - \hat{y}_i$:

$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$. They add up to the total variation in the data set:

$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2 = SSR + SSE$.

To compare these two components, we can put them into an ANOVA table.

Source of variation	Degrees of freedom	Sum of Squares	Mean Squares	F-statistic	p-value
Regression	1	SSR	MSR	$F_{obs} = \frac{MSR}{MSE}$	$P(F_{1,n-2} \geq F_{obs})$
Error	n-2	SSE	MSE		
Total	n-1	SST			

The null hypothesis that X does not affect Y ($H_0 : \beta_1 = 0$) is rejected if the p-value of the F-test is small.

This ANOVA table is calculated in R by anova(lm()).

Regression on gene expression values from two probes (continued)

This ANOVA can be gotten from the data set in R by anova(lm()).

```
> data(golub,package="multtest"); D38073_at <- golub[2289,]; X62153_s_at <- golub[2430,]
> reg.fit<-lm(X62153_s_at ~ D38073_at) #Fit a regression equation
> anova(reg.fit) #Show ANOVA table for regression fit
Analysis of Variance Table
Response: X62153_s_at
             Df Sum Sq Mean Sq F value    Pr(>F)
D38073_at     1 4.0033  4.0033  24.663 1.666e-05 ***
Residuals    36 5.8434  0.1623
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

From this output, the p-value=**1.666e-05** is small so that we reject the null hypothesis of zero slope parameter β_1.

Equivalence to other tests: Notice that we already have the t-test for $H_0 : \beta_1 = 0$. The t-test is equivalent to the F-test here: $(t_{df})^2 \sim F_{1,df}$. Compare with earlier R outputs for summary(reg.fit): t-value **4.966** which is the square-root of F-value **24.663** in this table, with the same p-value **1.67e-05**. Also, notice that $H_0 : \beta_1 = 0$ is the same as null hypothesis that the linear correlation coefficient is zero. The t-test for linear correlation coefficient is also the same test here.

```
> cor.test(X62153_s_at, D38073_at) #Test if correlation=0
        Pearson's product-moment correlation
data:  X62153_s_at and D38073_at
t = 4.9662, df = 36, p-value = 1.666e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3993383 0.7952115
sample estimates:
      cor
0.6376217
```

R-squared and F-test in the simple linear regression output

Let us study the R outputs for linear regression again in more detail.

```
> summary(reg.fit) #summary of regression results
Call:
lm(formula = X62153_s_at ~ D38073_at)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9137 -0.2239  0.1139  0.2350  0.4749

Coefficients:
             Estimate  Std. Error t value Pr(>|t|)
(Intercept)  0.29958    0.07258   4.127   0.000208 ***
D38073_at    0.54326    0.10939   4.966   1.67e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4029 on 36 degrees of freedom
Multiple R-squared:  0.4066,    Adjusted R-squared:  0.3901
F-statistic: 24.66 on 1 and 36 DF,  p-value: 1.666e-05
```

There are a few other quantities besides the parameter estimates we studied before. Particularly, the F-test results are also presented here. So if we do not care about other quantities in the ANOVA table, we can avoid using anova(reg.fit) and just use summary(reg.fit) for all regression inferences.

Also, we shall pay attention to the quantity R-squared $R^2 = SSR / SST$, the proportion of total variation that is explained by the regression model. Particularly we distinguish this from the F-statistic $F = MSR / MSE$. The F-test compares the mean squares which are adjusted for the degrees of freedom, to check if the regression effect is "statistically significant". R^2 compares the sum of squares, and reflects the "practical significance" of the regression model. A regression model explains only 0.1% of total variation can still be statistically significant (when the sample size n is big), but not useful in practice.

The R-squared $R^2 = \rho^2$ is also the square of the linear correlation coefficient. Hence the name. (You can check the R-squared **0.4066** in this example is the square of 0.6376217 from last page.) The Pearson's linear correlation coefficient definition is hard to extend to multiple random variables. Often $R^2 = SSR / SST$ is used to define a multiple linear correlation coefficient.

ANOVA versus regression (factor variable vs numerical variable)
Notice that the R command anova(lm(y~x)) is also used to do the ANOVA analysis in the previous module. The distinction between the regression and the previous ANOVA is in the type of variable x: continuous x for regression, and categorical (factor) x for ANOVA.

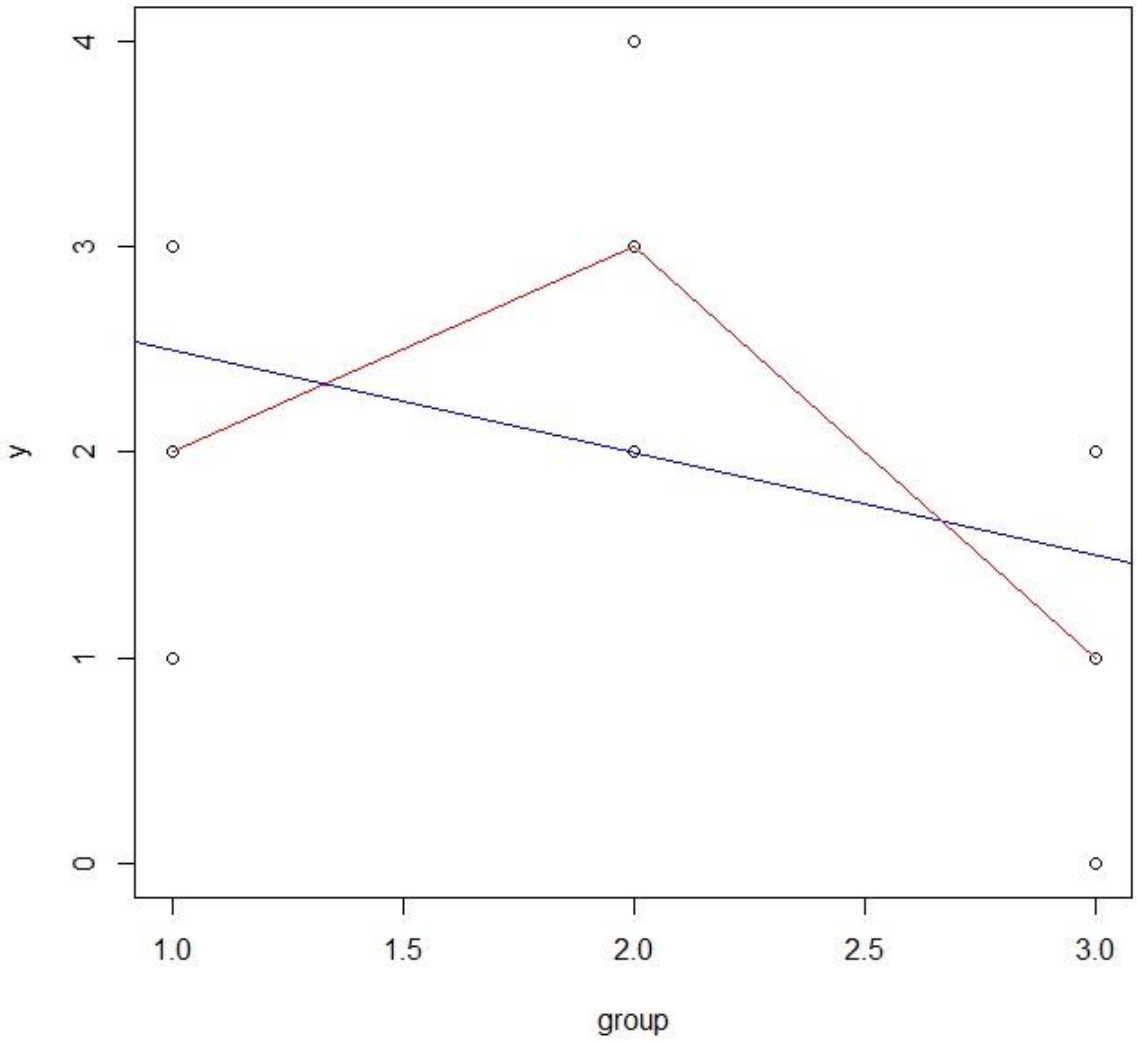
When the x values are just labels without real intrinsic numerical values, it is inappropriate to force a linear relationship on the data set. Then the one-way ANOVA model should be applied, by making x a factor variable using as.factor(x). Recall that actually creates dummy group indicator variables for the linear model fit (one variable for each level of x).

We illustrate this on the small artificial data set used in the module on ANOVA.

ANOVA versus regression (factor variable vs numerical variable)

We plot the data together with the regression fit and the ANOVA fit, using the following R commands

```
y<-c(2,3,1,3,2,4,2,0,1); group<-c(1,1,1,2,2,2,3,3,3) #create data in 3 groups
plot(y~group);  #scatterplot by group
abline(lm(y~as.numeric(group)), col='blue')  #regression fit line
lines(predict(lm(y~as.factor(group)))~group,pch="*", col=2) #anova fit line
```



We can see that the linear regression model tries to fit a linear trend (blue line) to the data, while ANOVA just fits to the group means (red line). Since the group label does not has real intrinsic numerical meanings, the linear fit is unreasonable.

Notice the ***numerical*** "group" variable results in a linear regression fit, and the ***factor*** "group" variable results in a one-way ANOVA fit.

Lesson Summary

We taught the statistical inferences with the linear regression model. We taught the t-intervals and t-test the parameters β_0 and β_1. You should also know the confidence interval and prediction interval for the response variable.

We also discussed the ANOVA analysis for the regression model. You should know the meaning of the various quantities in the R output for regression fit.

We will teach the model checking for linear regression in the next lesion.

Lesson 3: Diagnostic tests, and transformations.

By the end of this lesson you will have had opportunity to:

- Check the regression model assumptions by residuals plots
- Conduct normality assumption with Shapiro-Wilk test
- Use logarithm transformations to improve regression fitting

Overview

The data analyst should always check the model assumptions to make sure that the statistical analysis is appropriate. For the regression model, we can use the Shapiro-Wilk test on residuals to test the normality. We can also use the residuals plots to detect violations of the model assumptions.

Finally, we show one method of remedying the regression model by applying logarithm transformations on the variables.

Check for the regression model assumptions

The linear regression model assumes that

$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$, for $i = 1, ..., n$,

where $\varepsilon_1, ..., \varepsilon_n$ are random noises $\sim N(mean = 0, sd = \sigma)$.

We need to check if $\varepsilon_1, ..., \varepsilon_n$ are really normal random noises. We introduced before how to check the normality assumption by qq-plot and Shapiro-Wilk test. They can be applied here to the residuals $e_i = y_i - \hat{y}_i$.

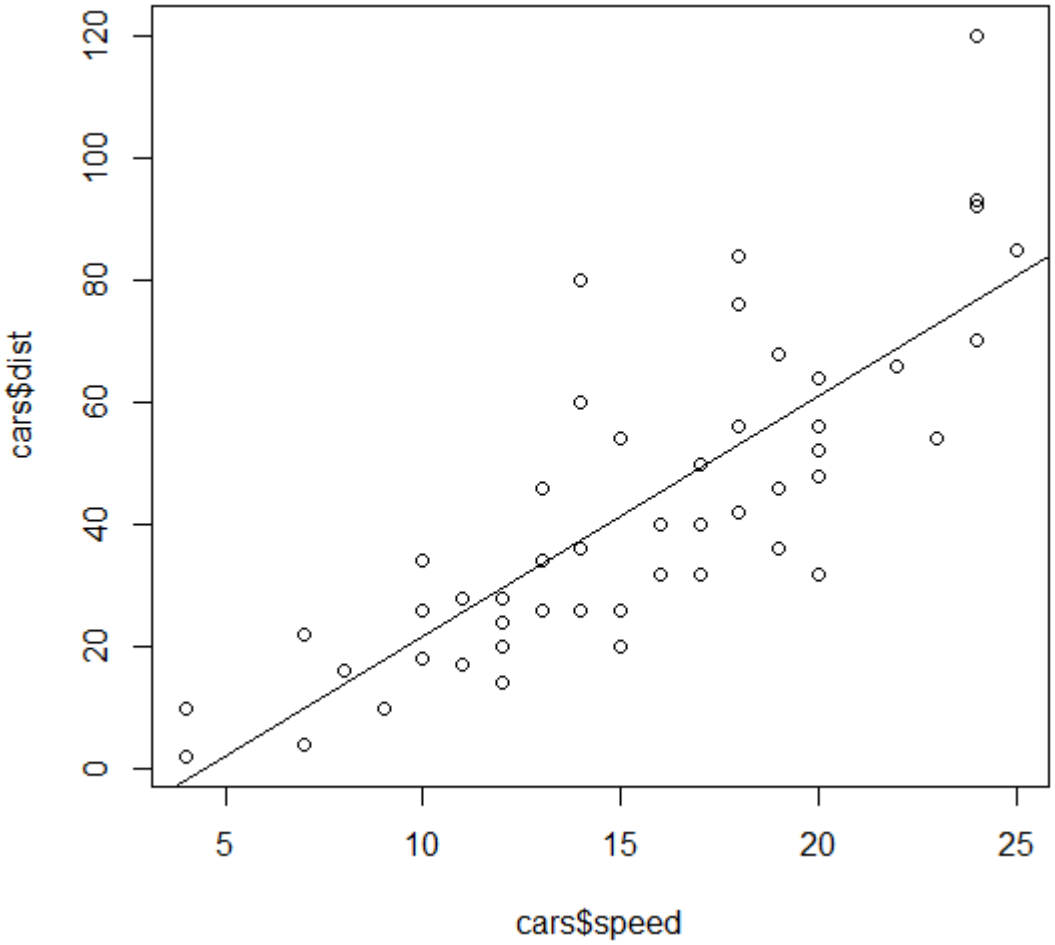
We also wants to check if the errors are independent of Xs as assumed. This generally is checked by plotting the residuals $e_i = y_i - \hat{y}_i$ versus the fitted values $\hat{y}_i$ to check for any patterns. (There should be no pattern if the errors are really random noises.)

We illustrate these checking techniques on the "cars" data set that comes with R.

Linear regression on 'cars' data set

Example 1: We look at the "cars" dataset that comes with R. The data includes measurements of speed (mph) and stopping distance (ft) of cars in the 1920s. We first plot the data with the regression line.

```
lin.reg<-lm(dist ~ speed, data=cars) #regression distance on speed for 'cars' data
plot(cars$speed,cars$dist)       #scatterplot
abline(lin.reg)   #add regression line to scatterplot
```

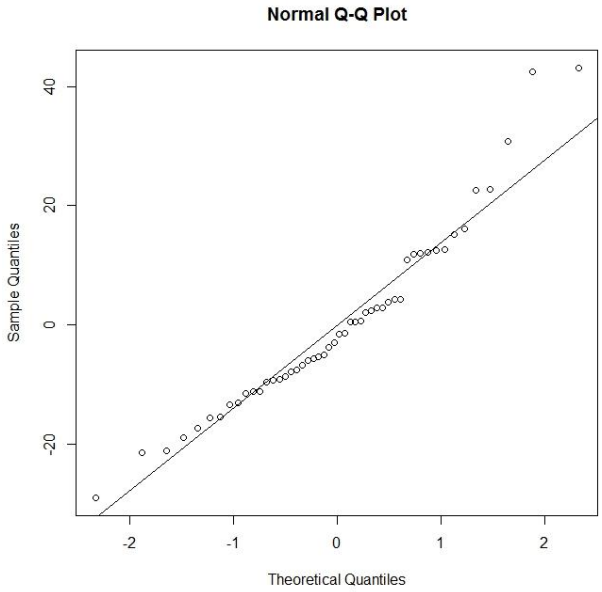


From the plot, there is a clear linear trend which seems to be captured by the regression. But does this really fit the linear regression model? The model checking would be easier when we plot the residuals instead of the raw data.

Linear regression on 'cars' data set (continued)

We can get the residuals from the regression by residuals() function. Then we can check the normality assumption from the qq-plot of the residuals.

```
qqnorm(resid(lin.reg)) #q-q plot of residuals
qqline(resid(lin.reg))   #add the straight line to q-q plot
```



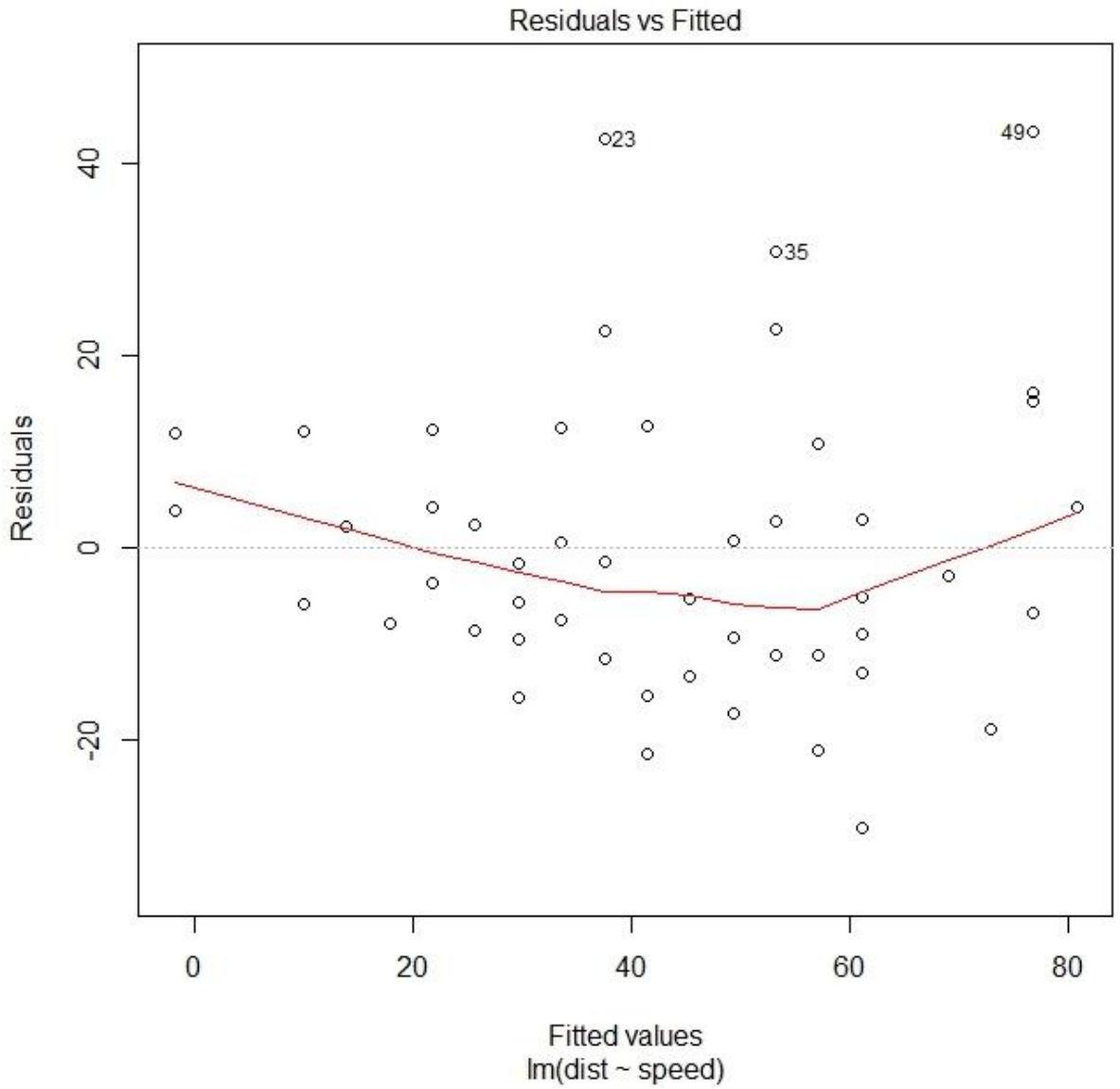
From the qq-plot, the residuals deviate from the normal distribution in the tails (mostly in the large values). We can further confirm it with the Shapiro-Wilk test introduced in earlier modules.

```
> shapiro.test(resid(lin.reg))  #normality test on residuals
        Shapiro-Wilk normality test
data:  resid(lin.reg)
W = 0.9451, p-value = 0.02152
```

Hence the noise are not normally distributed here.

Linear regression on 'cars' data set (continued)

We also often plot the residuals against the fitted values to find patterns of violation of the regression model assumption: plot(predict(lin.reg), resid(lin.reg)). This plot is one of the standard diagnostic plots in R. So we can also get it directly from plot() of the "lm" object: plot(lin.reg,which=1).



The R plot also add a smoothing (red) line to help us judge patterns in the mean. We can see that there seems to be some non-linear pattern in the means. Also, the variance looks to be increasing with fitted values, violating the homoscedasticity assumption.

R commands for diagnostic plots

R has programmed some standard diagnostic plots for the linear model. We can simply use plot() to get them. The first plot, plot(lm(),which=1), gives the residuals versus fitted values. The second plot, plot(lm(),which=2), gives the qq-plot of the residuals.

We do not teach the other diagnostic plots here. Interested students can read further the R help file for plot.lm and the references there.

We next teach a common remedy when the regression model assumptions are violated.

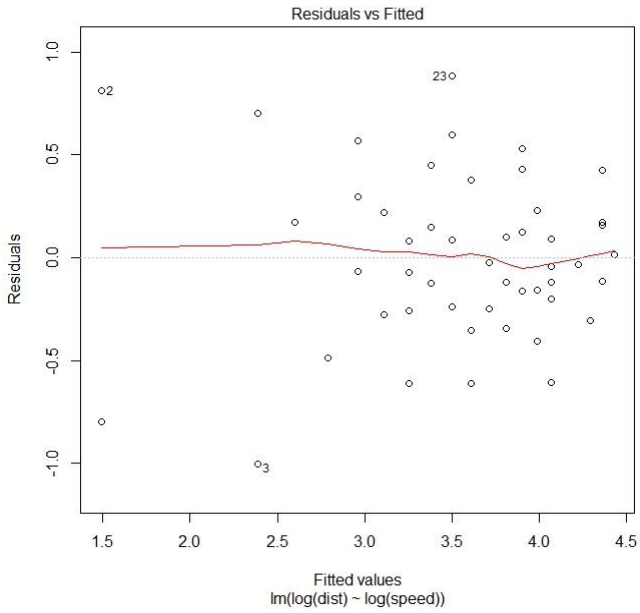
Log-transformation

When the variances increases with the fitted values that generally indicate a logarithm transformation should be applied to the variables before the linear regression. We illustrate this on the cars data set above. We apply the log-transformation on both the distance and the speed, then redo the linear regression.

```
> log.reg<-lm(log(dist) ~ log(speed), data=cars) #regression on log scales
> shapiro.test(resid(log.reg)) # normality test on residuals
        Shapiro-Wilk normality test
data:  resid(log.reg)
W = 0.9911, p-value = 0.9684
```

The residuals now looks like normally distributed. (qq-plot will also confirm this). Also the residuals now look random in the plot versus fitted value, with no obvious patterns in the means or the variances.

```
> plot(log.reg,which=1) #1st diagnostic plot (residuals vs. predicted)
```



Log-transformed linear regression on the cars data

Let us look at the regression fit on cars data after log-transformation on both variables.

```
> summary(log.reg) #summary of regression results
Call:
lm(formula = log(dist) ~ log(speed), data = cars)
Residuals:
     Min       1Q   Median       3Q      Max
-1.00215 -0.24578 -0.02898  0.20717  0.88289
Coefficients:
             Estimate  Std. Error   t value  Pr(>|t|)
(Intercept)   -0.7297     0.3758    -1.941   0.0581 .
log(speed)    1.6024     0.1395     11.484   2.26e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4053 on 48 degrees of freedom
Multiple R-squared:  0.7331,    Adjusted R-squared:  0.7276
F-statistic: 131.9 on 1 and 48 DF,  p-value: 2.259e-15
```

We can see that the linear regression now explains 73% of variation in the data, a very good fit. The regression equation is

log(dist)= -0.73 + 1.6 log(speed)

In other words,

$dist = e^{-0.73} speed^{1.6} = 0.48 speed^{1.6}$.

So a car driving at 30 mph will need 112 ($= (0.48)30^{1.6}$) feet distance to stop.

Lesson Summary

This lesson covered some basic diagnostic tools for linear regression model. You should know how to test the normality assumption. And also knows how to use the residuals plots to find possible violation of model assumption.

We also taught a common trick to improve linear regression fitting when the variance increases with the response variable: apply logarithm transformations on both variables. The linear regression model on the logarithm scales assume that the two variables are related through a power function.

We discuss multiple regression in the next lesson.

Lesson 4: Multiple Linear Regression.

By the end of this lesson you will have had opportunity to:

- Fit a multiple regression model
- Check the model assumptions with diagnostic tools

Overview

There are often more than one variables that can affect the response variable Y. We can include them jointly in a regression model. That is called multiple linear regression. We describe the model notation, and use one example to illustrate the fitting of multiple linear regression.

After this lesson, you should know how to fit the multiple linear regression model in R, and how to read the R outputs.

Multiple linear regression model

We there are multiple (p≥2) independent variables $X_1,...,X_p$, we can regress the response variable Y on them together. That is called the multiple linear regression model, which assumes

$Y_i = \beta_0 + \beta_1 X_{1,i} + ... + \beta_p X_{p,i} + \varepsilon_i$, for $i = 1,...,n$,

where $\varepsilon_1,...,\varepsilon_n$ are random noises $\sim N(mean = 0, sd = \sigma)$.

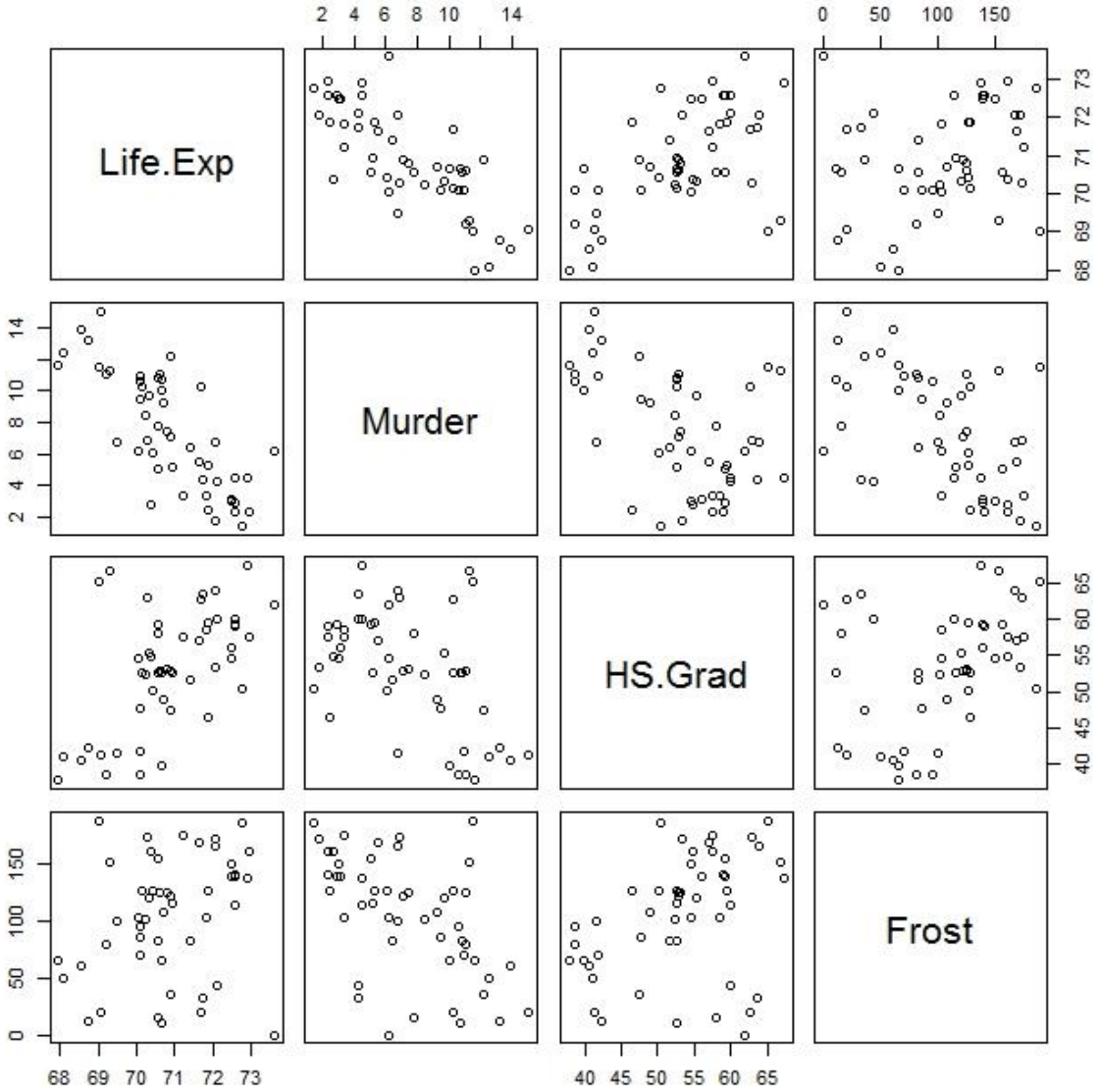
The fitting is still done in R by lm(). We illustrate this on the 'state.x77' data set that comes with R. The data set contains statistics about the 50 states in US, published by the Bureau of Census in the year 1977. You can inspect the data set with str(state.x77) and find explanation and source references by ?state.x77

Demonstration: multiple regression on state.x77 data

We consider the prediction of the life expectancy by the murder rates, percentage of high-school graduates and mean number of frost days (days with minimum temperature below freezing). We first take a subset of data containing these variables and make it a data frame. Also, change the names to avoid space which makes it hard for us to apply functions on it.

```
the.data<-as.data.frame(state.x77[,c('Life Exp', 'Murder', 'HS Grad', 'Frost')]) #take out the four variables from state.x77 data set
names(the.data)<-c('Life.Exp', 'Murder', 'HS.Grad', 'Frost') #variable names
```

We can first draw the scatterplots for a visual inspection using `pairs(the.data)`



There are clear linear trends of Life.Exp with regard to Murder and HS.Grad. The Frost also seems to have some impact.

Demonstration: multiple regression on state.x77 data

We conduct the multiple regression in R.

```
> lin.reg<-lm(Life.Exp~Murder+HS.Grad+Frost, data=the.data) #multiple regression of Life.Exp on 3 variables
> summary(lin.reg) #summary of regression results
Call:
lm(formula = Life.Exp ~ Murder + HS.Grad + Frost, data = the.data)
Residuals:
    Min      1Q  Median      3Q     Max
-1.5015 -0.5391  0.1014  0.5921  1.2268
Coefficients:
             Estimate   Std. Error  t value  Pr(>|t|)
(Intercept) 71.036379   0.983262  72.246   < 2e-16 ***
Murder      -0.283065   0.036731  -7.706    8.04e-10 ***
HS.Grad      0.049949   0.015201   3.286    0.00195 **
Frost       -0.006912   0.002447  -2.824    0.00699 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7427 on 46 degrees of freedom
Multiple R-squared:  0.7127,    Adjusted R-squared:  0.6939
F-statistic: 38.03 on 3 and 46 DF,  p-value: 1.634e-12
```

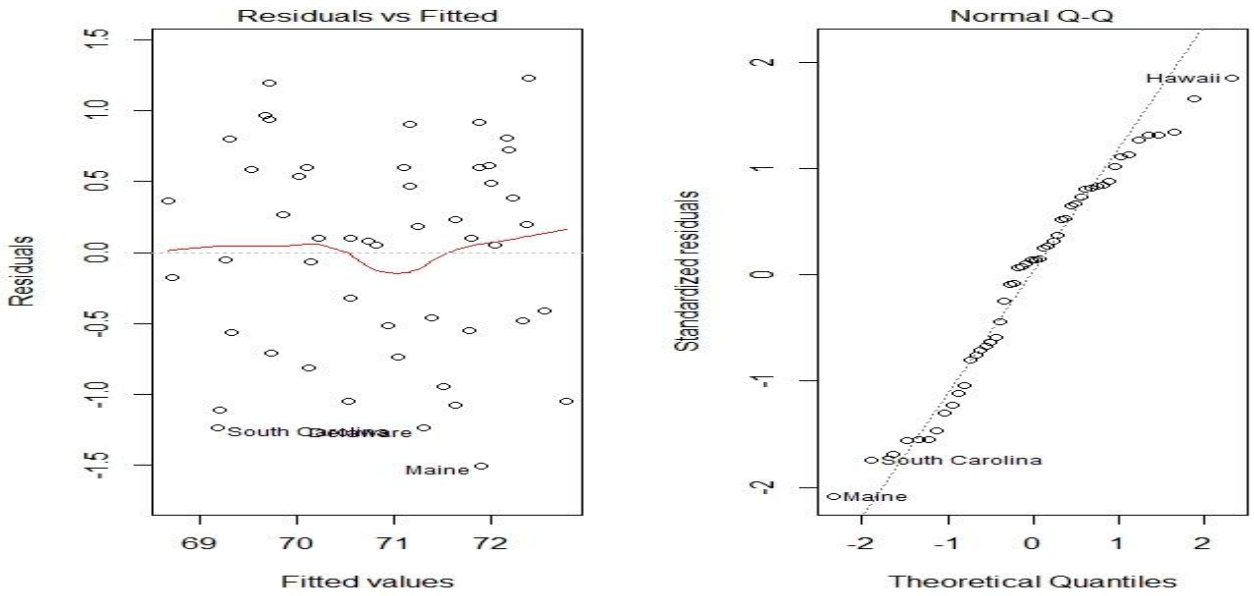
We can see that fitted regression equation is

Life.Exp = 71 -0.28Murder + 0.050 HS.Grad -0.0069 Frost.

All three variables are statistically significantly related to the life expectance. Together they explain (R-squared) 71.3% of total variation in the life expectance across states.

Demonstration: multiple regression on state.x77 data

We check the model assumptions using the residuals plots.



The residuals looks random with no obvious patterns in means or in variances. The qq-plot seems fine with slight derivation from normal distribution in the right tail. The formal normality test has a p-value=0.10.

```
> shapiro.test(resid(lin.reg)) #normality test on residuals
        Shapiro-Wilk normality test
data:  resid(lin.reg)
W = 0.9615, p-value = 0.1024
```

Lesson Summary

This lesson taught how to conduct multiple linear regression model. You should know how to fit the multiple linear regression model in R, and how to use diagnostic tools to check the regression model assumptions.

Module Summary

This module covered linear regression model. You should know how to calculate the linear correlation coefficient and how to do hypothesis testing about the linear correlation coefficient. You should know how to fit the linear regression model, and make appropriate statistical inferences based on the R outputs. Finally, you should be able to check the regression model assumptions on the data set.

In the next module, we will consider the processing of microarray data.