Making Predictions Using Linux and R

timriley@appahost.com

Descriptive Statistics

Mean

```
$ less dice_samples.c
$ (echo dice; dice_samples 1 5 | piece ‘,’ 0 ) > five_dice.csv
> table = read.csv( file=”five_dice.csv”, header=T)
> table$dice
> sum = sum( table$dice )
> length = length( table$dice )
> mean = sum / length
> mean = mean( table$dice )
> average = mean
```

Median

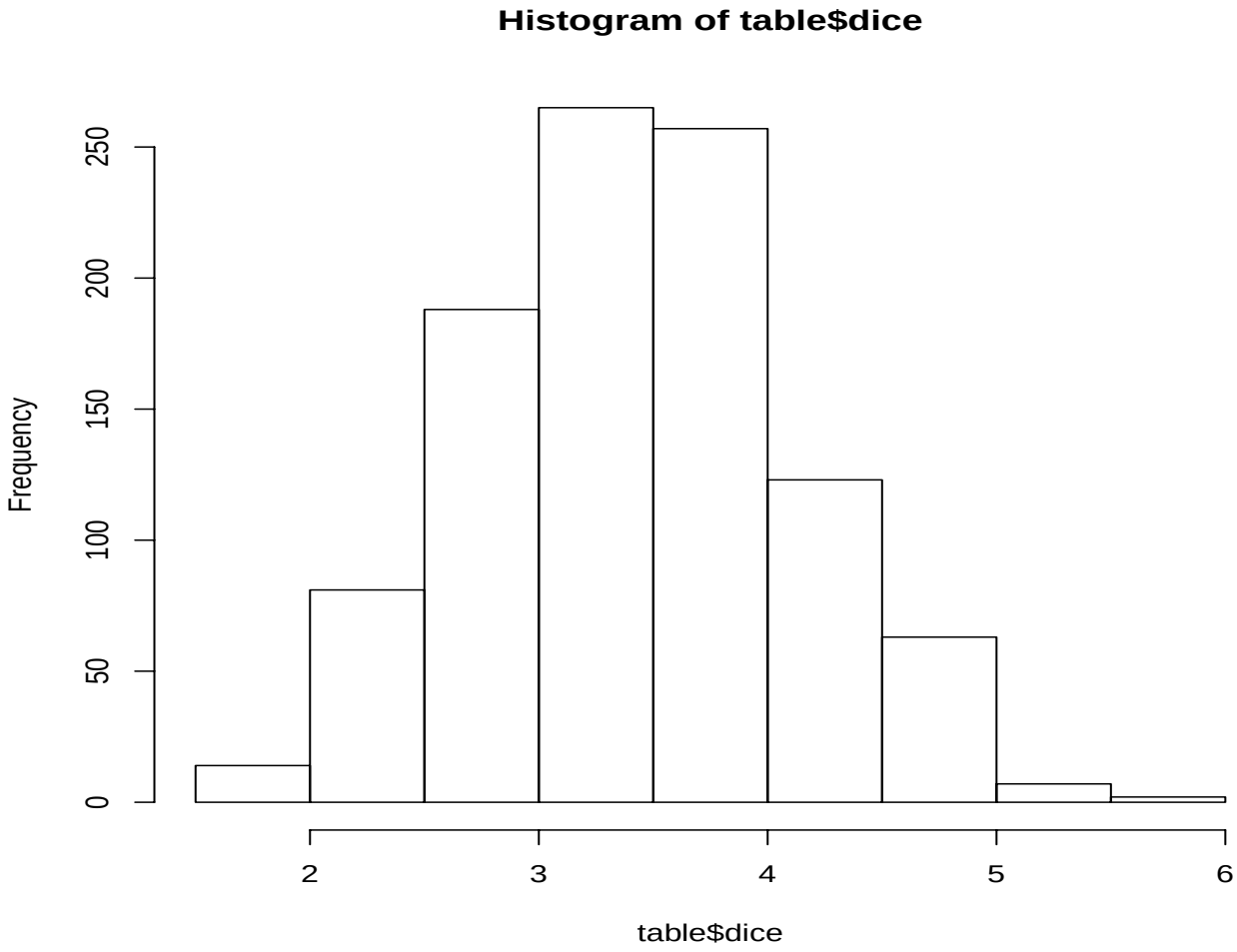
```
> sort = sort( table$dice )
>       if ( length %% 2 == 1 )
        {
                middle_index = trunc( length / 2 + 1 )
                median = sort[ middle_index ]
        }
        if ( length %% 2 == 0 )
        {
                middle_index_1 = length / 2
                middle_index_2 = length / 2 + 1
                median = ( sort[ middle_index_1 ] + sort[ middle_index_2 ] ) / 2
        }
> median = median( table$dice )
```

Standard Deviation

```
> squared_distance_from_mean_array =
        c( ( table$dice - mean ) ^ 2 )
> sum_squared_distance_from_mean =
        sum( squared_distance_from_mean_array )
> degrees_of_freedom = length - 1
> variance = sum_squared_distance_from_mean / degrees_of_freedom
> standard_deviation = sqrt( variance )
> standard_deviation = sd( table$dice )
```

Note: 75% of the measurements will fall between +- 2 standard deviations around the mean. 89% will fall between +- 3 standard deviations.

Normal Distribution



Characteristics

1. The normal distribution is a mound shaped histogram.
2. X axis are buckets.
3. Y axis are the number of measurements in each bucket.
4. 67% of measurements will fall between +- 1 standard deviation around the mean.
5. 95% of measurements will fall between +- 2 standard deviations around the mean.
6. 99.7% of measurements will fall between +- 3 standard deviations around the mean.

Central Limit Theorem

You will always get an approximately normal distribution if you place into the buckets the averages taken from many samples.

1. The larger the sample size making up the averages or the larger the number of samples, the more mound-shaped.
2. Repetions of 30 in each sample is sufficient.
3. The number of samples necessary is dependent upon the variability of the population.

```
$ less dice_samples_average.sh
$ less dice_samples_average_histogram.sh
$ dice_samples_average_histogram.sh 1 300
$ dice_samples_average_histogram.sh 2 300
$ dice_samples_average_histogram.sh 30 30
$ dice_samples_average_histogram.sh 6 150
```

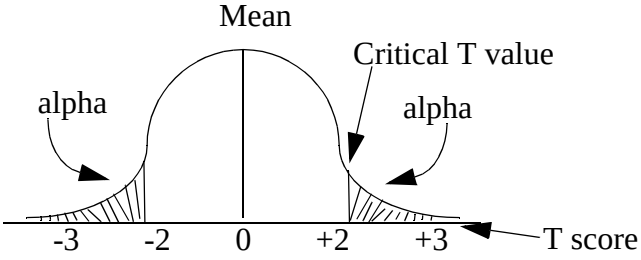
Hypothesis Test

```
> table = read.csv( file="lake_acidity.csv", header=T )
> table
> attach( table )
> hist( ph_level )
> minimum = min( ph_level )
> maximum = max( ph_level )
> mean = mean( ph_level )
> median = median( ph_level )
> standard_deviation = sd( ph_level )
> n = length( ph_level )
> degrees_of_freedom = n - 1
```

Assume there is too much acid in all the lakes -- the mean pH of all the lakes is 6.

```
> null_hypothesis = 6
> standard_deviation_of_the_mean = standard_deviation / sqrt( n )
> t_score = abs( mean - null_hypothesis ) / standard_deviation_of_the_mean
> confidence_level = 0.95
> alpha = ( 1 - confidence_level ) / 2
> critical_t_value = abs( qt( alpha, degrees_of_freedom ) )
```

The T Curve




```
>       if ( t_score > critical_t_value )
        {
                print( "reject, the population mean is different from the null hypothesis" )
        }
> decision_confidence = pt( t_score, degrees_of_freedom ) * 2 - 1
> p_value = 1 - decision_confidence
```

Population Confidence Interval

```
> margin_of_error = critical_t_value * standard_deviation_of_the_mean
```

```
> population_mean_minimum = mean - margin_of_error
> population_mean_maximum = mean + margin_of_error
```

Pooled t-test

1. Take a population and divide it into two subsets.
2. Are the means of the two subsets truly different? Or is the difference because of sampling variability?
3. Pooled t-test assumes the two subsets have the same standard deviation.

```
> private = read.csv( file=”private.csv”, header=T )
> public = read.csv( file=”public.csv”, header=T )
> length_private = length( private$private )
> length_public = length( public$public )
> mean_private = mean( private$private )
> mean_public = mean( public$public )
> hist( private$private )
> hist( public$public )
> min( private$private ); max( public$public )
> sd_private = sd( private$private )
> sd_public = sd( public$public )
> sd_ratio = sd_private / sd_public
>       if ( sd_ratio < 0.5 || sd_ratio > 2 )
        {
                print( “Subsets do not have the same standard deviations.” )
        }
```

Calculate Pooled Standard Deviation

```
> degrees_of_freedom_private = length( private$private ) - 1
> degrees_of_freedom_public = length( public$public ) - 1
> variance_private = var( private$private )
> variance_public = var( public$public )
> degrees_of_freedom =
        length( private$private ) + length( public$public ) - 2
> pooled_variance =               ( degrees_of_freedom_private * variance_private +
                                    degrees_of_freedom_public * variance_public ) /
                                  degrees_of_freedom
> pooled_standard_deviation = sqrt( pooled_variance )
```

Calculate Pooled t score

```
> pooled_t_score_numerator =
        abs( mean( private$private - mean( public$public ) )
> pooled_t_score_denominator =
                                  pooled_standard_deviation *
                                  sqrt(         1 / length( private$private ) +
                                                1 / length( public$public ) )
```

```
> pooled_t_score = pooled_t_score_numerator / pooled_t_score_denominator
```

Make Decision

```
> confidence_level = 0.95
> alpha = ( 1 - confidence_level ) / 2
> critical_t_value = abs( qt( alpha, degrees_of_freedom ) )
>       if ( pooled_t_score > critical_t_value )
        {
                print( “reject, population means are different” )
        }
> decision_confidence = pt( pooled_t_score, degrees_of_freedom ) * 2 - 1
> p_value = 1 - decision_confidence
```

Mean Difference Confidence Interval

```
> mean_difference = abs( mean_private - mean_public )
> margin_of_error = critical_t_value * pooled_t_score_denominator
> mean_difference_minimum = mean_difference - margin_of_error
> mean_difference_maximum = mean_difference + margin_of_error
```

Analysis of Variance (ANOVA)

1. Take a population and divide it into many subsets.
2. Are the means of many subsets truly different? Or is the difference because of sampling variability?
3. ANOVA assumes the many subsets have the same standard deviation.

```
> private = read.csv( file=”private.csv”, header=T )
> public = read.csv( file=”public.csv”, header=T )
> length_private = length( private$private )
> length_public = length( public$public )
> level_count = 2
> mean_private = mean( private$private )
> mean_public = mean( public$public )
> hist( private$private )
> hist( public$public )
> min( private$private ); max( public$public )
> sd_private = sd( private$private )
> sd_public = sd( public$public )
> sd_ratio = sd_private / sd_public
>       if ( sd_ratio < 0.5 || sd_ratio > 2 )
        {
                print( “Subsets do not have the same standard deviations.” )
        }
> global_array = c( private$private, public$public )
> global_mean = mean( global_array )
> global_length = length( global_array )
```

Variation Among

```
> numerator_degrees_of_freedom = level_count - 1
> sum_distance_level_mean_from_global_mean_squared_times_level_length =
        length_private * ( mean_private - global_mean )^2 +
        length_public * ( mean_public - global_mean )^2
> space_between_levels =
        sum_distance_level_mean_from_global_mean_squared_times_level_length /
        numerator_degrees_of_freedom
> distance_between = space_between_levels
> mean_square_treatment = space_between_levels
> variation_among = space_between_levels
```

Variation Within

```
> variance_private = var( private$private )
> variance_public = var( public$public )
> sum_level_variance_times_degrees_of_freedom =
        ( length_private - 1 ) * variance_private +
        ( length_public - 1 ) * variance_public
> denominator_degrees_of_freedom = global_length - level_count
> space_within_levels =
        sum_level_variance_times_degrees_of_freedom /
        denominator_degrees_of_freedom
> distance_within = space_within_levels
> mean_square_error = space_within_levels
```

F-statistic

```
> f_score = distance_between / distance_within
> confidence_level = 0.95
> alpha = ( 1 - confidence_level ) / 2
> critical_f_value =
        qf(     1 - alpha,
                numerator_degrees_of_freedom,
                denominator_degrees_of_freedom )
>       if ( f_score > critical_f_value )
        {
                print( “reject, population means are different” )
        }
> decision_confidence =
        pf(     f_score,
                numerator_degrees_of_freedom,
                denominator_degrees_of_freedom )
> p_value = 1 - decision_confidence
```

Aov()

```
$ (     echo “salary,level”                 ;
        cat private.csv                     |
        grep -v private                     |
        sed ‘s/$/,private/’                 ;
        cat public.csv                      |
        grep -v public                      |
        sed ‘s/$/,public/’ )                |
  cat > professor_salaries.csv
> table = read.csv( file=”professor_salaries.csv”, header=T )
> summary( table )
> boxplot( table$salary ~ table$level )
> aov = aov( table$salary ~ table$level )
> summary( aov )
```

Best Fit Line (Linear Regression)

```
> table = read.csv( file=”fire_damage.csv”, header=T )
> plot( table$distance, table$damage, pch=19 )
> x_array = table$distance
> y_array = table$damage
> fit = lm( y_array  ~ x_array )
> summary( fit )
> abline( fit, col=”red” )
```

Expected Values

```
> sum_distance_x_from_mean_times_distance_y_from_mean =
        sum( c( ( x_array - mean( x_array ) ) * ( y_array - mean( y_array ) ) ) )
> sum_distance_x_from_mean_squared =
        sum( c( x_array - mean( x_array )  ) ^ 2 )
> slope =
        sum_distance_x_from_mean_times_distance_y_from_mean /
        sum_distance_x_from_mean_squared
> slope = fit$coefficients[[ 2 ]]
> y_intercept = mean( y_array ) - slope * mean( x_array )
> y_intercept = fit$coefficients[[ 1 ]]
> f = function( x_point )
        {
                return ( slope * x_point + y_intercept )
        }
> expected_x_array = min( x_array ):max( x_array )
> expected_y_array = c( f( expected_x_array ) )
> points( expected_x_array, expected_y_array, col=”red” )
```

Residual Standard Deviation

```
> residual_array = c( ( y_array - f( x_array ) ) )
> residual_array = resid( fit )
> residual_array = fit$residuals
> sum_residual_array = sum( residual_array )
> print( sprintf( “%15.13f”, sum_residual_array ) )
> sum_residuals_squared = sum( c( residual_array ^ 2 ) )
> sum_residual_squared = deviance( fit )
> degrees_of_freedom = length( x_array ) - 2
> degrees_of_freedom = df.residual( fit )
> residual_variance =
      sum_residuals_squared / degrees_of_freedom
> residual_standard_deviation = sqrt( residual_variance )
> residual_standard_deviation = summary( fit )[[ 6 ]]
```

As an acid test, 95% of the dependent points should lie within
+- 2 * residual_standard_deviation of the best fit line.

Coefficient of Determination

```
> sum_distance_y_from_mean_squared =
      sum( c( y_array - mean( y_array ) ) ^ 2 )
> total_variability = sum_distance_y_from_mean_squared
> explained_variability =
      total_variability -
      sum_residual_squared
> coefficient_of_determination =
      explained_variability / total_variability
> coefficient_of_determination = cor( y_array, x_array ) ^ 2
> coefficient_of_determination = summary( fit )$r.squared
> coefficient_of_determination = summary( fit )[[ 8 ]]
> unexplained_variability_ratio = 1 - coefficient_of_determination
```

Is the slope significant?

Assume the slope is not significant.

```
> sum_distance_x_from_mean_squared =
      sum( c( x_array - mean( x_array ) ) ^ 2 )
> slope_standard_deviation =
      residual_standard_deviation /
      sqrt( sum_distance_x_from_mean_squared )
> slope_standard_deviation = summary( fit )[[ 4 ]][[ 4 ]]
> t_score = abs( slope / slope_standard_deviation )
> t_score = abs( summary( fit )[[ 4 ]][[ 6 ]] )
> confidence_level = 0.95
> alpha = ( 1 - confidence_level ) / 2
> critical_t_value = abs( qt( alpha, degrees_of_freedom ) )
```

```
>       if ( t_score > critical_t_value )
        {
                print( “reject, the slope is significant” )
        }
> decision_confidence =
        pt( t_score, degrees_of_freedom ) * 2 - 1
> print( sprintf( “%15.13f”, decision_confidence ) )
> p_value = 1 - decision_confidence
> p_value = summary( fit )[[ 4 ]][[ 8 ]]
> print( sprintf( “%15.13f”, p_value ) )
```

Slope Confidence Interval

```
>       slope_minimum =
                slope -
                critical_t_value *
                slope_standard_deviation
> slope_minimum = confint( fit, level=confidence_level )[ 2, 1 ]

>       slope_maximum =
                slope +
                critical_t_value *
                slope_standard_deviation
> slope_maximum = confint( fit, level=confidence_level )[ 2, 2 ]
```

Y-Intercept Confidence Interval

```
> sum_x_then_square = sum( c( x_array ) ) ^ 2
> sum_x_squared = sum( c( x_array ^ 2 ) )
> y_intercept_standard_deviation =
        residual_standard_deviation *
        sqrt( 1 / ( length( x_array ) -
        ( sum_x_then_square / sum_x_squared ) ) )
> y_intercept_standard_deviation = summary( fit )[[ 4 ]][[ 3 ]]

> y_intercept_minimum =
                y_intercept -
                critical_t_value *
                y_intercept_standard_deviation
> y_intercept_minimum = confint( fit, level=confidence_level )[ 1, 1 ]

> y_intercept_maximum =
                y_intercept +
                critical_t_value *
                y_intercept_standard_deviation
> y_intercept_maximum = confint( fit, level=confidence_level )[ 1, 2 ]
```

Confidence Interval at a Point

```
> predicted_mean_standard_deviation = function( x_point )
      {
            residual_standard_deviation *
            sqrt(             ( 1 / length( x_array ) ) +
                              ( x_point - mean( x_array ) ) ^ 2 /
                              sum_distance_x_from_mean_squared )
      }
>     f_min = function( x_point )
      {
            return (          f( x_point ) -
                              critical_t_value *
                              predicted_mean_standard_deviation( x_point ) )
      }
>     f_max = function( x_point )
      {
            return (          f( x_point ) +
                              critical_t_value *
                              predicted_mean_standard_deviation( x_point ) )
      }
> predicted_minimum_y_array = c( f_min( predicted_x_array ) )
> points( predicted_x_array, predicted_minimum_y_array, col="red" )
> predicted_maximum_y_array = c( f_max( predicted_x_array ) )
> points( predicted_x_array, predicted_maximum_y_array, col="red" )
```

Best Fit Curve (Curvilinear Regression)

```
> table = read.csv( file="home_electricity.csv", header=T )
> plot( table$size, table$usage, pch=19 )
> x_array = table$size
> x_squared = I(x_array ^ 2 )
> y_array = table$usage
> fit = lm( y_array ~ x_array + x_squared )
> summary( fit )
> y_intercept = fit$coefficients[[ 1 ]]
> beta = fit$coefficients[[ 2 ]]
> quadratic_beta = fit$coefficients[[ 3 ]]
> f = function( x_point )
      {
            return (          quadratic_beta * ( x_point ^ 2 ) +
                              beta * x_point +
                              y_intercept )
      }
> predicted_x_array = min( x_array ):max( x_array )
> predicted_y_array = c( f( predicted_x_array ) )
> points( predicted_x_array, predicted_y_array, col="red" )
```

Residual Standard Deviation

```
> residual_standard_deviation = summary( fit )[[ 6 ]]
```

As an acid test, 95% of the dependent points lie within
+- 2 * residual_standard_deviation of the best fit curve.

Coefficient of Determination

```
> coefficient_of_determination = summary( fit )$r.squared
> unexplained_variability_ratio = 1 - coefficient_of_determination
```

Is quadratic beta significant?

Assume quadratic beta is not significant.

```
> degrees_of_freedom = df.residual( fit )
> quadratic_beta_standard_deviation = summary( fit )[[ 4 ]][[ 6 ]]
> t_score = abs( quadratic_beta / quadratic_beta_standard_deviation )
> t_score = abs( summary( fit )[[ 4 ]][[ 9 ]] )
> confidence_level = 0.95
> alpha = ( 1 - confidence_level ) / 2
> critical_t_value = abs( qt( alpha, degrees_of_freedom ) )
>       if ( t_score > critical_t_value )
        {
                print( “reject, the quadratic beta is significant” )
        }
> decision_confidence =
        pt(     t_score, degrees_of_freedom ) * 2 - 1
> print( sprintf( “%15.13f”, decision_confidence ) )
> p_value = 1 - decision_confidence
> p_value = summary( fit )[[ 4 ]][[ 12 ]]
> print( sprintf( “%15.13f”, p_value ) )
```

Quadratic Beta Confidence Interval

```
>       quadratic_beta_minimum =
                quadratic_beta -
                critical_t_value *
                quadratic_beta_standard_deviation
> quadratic_beta_minimum = confint( fit, level=confidence_level )[ 3, 1 ]

>       quadratic_beta_maximum =
                quadratic_beta +
                critical_t_value *
                quadratic_beta_standard_deviation
> quadratic_beta_maximum = confint( fit, level=confidence_level )[ 3, 2 ]
```

Confidence Interval

```
> f_min = function( x_point )
	{
		return ( f( x_point ) - 2 * residual_standard_deviation )
	}
> f_max = function( x_point )
	{
		return ( f( x_point ) + 2 * residual_standard_deviation )
	}
> predicted_minimum_y_array = c( f_min( predicted_x_array ) )
> points( predicted_x_array, predicted_minimum_y_array, col=”red” )
> predicted_maximum_y_array = c( f_max( predicted_x_array ) )
> points( predicted_x_array, predicted_maximum_y_array, col=”red” )
```

Multiple Regression

```
> install.packages( “scatterplot3d”, repos=”http://R-Forge.R-project.org” )
> library( scatterplot3d )
> table = read.csv( file=”clock_auction.csv”, header=T )
> attach( table )
> pairs( price ~ age + bidders )
> s3d = scatterplot3d( age, bidders, price, pch=19, type=”h” )
> x_array = age
> y_array = price
> z_array = bidders
> fit = lm( y_array ~ x_array + z_array )
> summary( fit )
> s3d$plane3d( fit, col=”red” )
> y_intercept = fit$coefficients[[ 1 ]]
> beta_x = fit$coefficients[[ 2 ]]
> beta_z = fit$coefficients[[ 3 ]]
> f = function( x_point, z_point )
	{
		return (		beta_x * x_point +
					beta_z * z_point +
					y_intercept )
	}
```

Residual Standard Deviation

```
> residual_standard_deviation = summary( fit )[[ 6 ]]
```

As an acid test, 95% of the dependent points lie within
+- 2 * residual_standard_deviation of the best fit plane.

Coefficient of Determination

```
> coefficient_of_determination = summary( fit )$r.squared
> unexplained_variability_ratio = 1 - coefficient_of_determination
```

Is beta_x significant?

Assume beta_x is not significant.

```
> degrees_of_freedom = df.residual( fit )
> beta_x_standard_deviation = summary( fit )[[ 4 ]][[ 5 ]]
> t_score = abs( beta_x / beta_x_standard_deviation )
> t_score = abs( summary( fit )[[ 4 ]][[ 8 ]] )
> confidence_level = 0.95
> alpha = ( 1 - confidence_level ) / 2
> critical_t_value = abs( qt( alpha, degrees_of_freedom ) )
>       if ( t_score > critical_t_value )
        {
                print( “reject, beta_x is significant” )
        }
> decision_confidence =
        pt( t_score, degrees_of_freedom ) * 2 - 1
> print( sprintf( “%15.13f”, decision_confidence ) )
> p_value = 1 - decision_confidence
> p_value = summary( fit )[[ 4 ]][[ 11 ]]
> print( sprintf( “%15.13f”, p_value ) )
```

Are all the betas significant?

Assume all the betas are not significant.

```
> number_of_independent_variables = 2
> sample_size = length( x_array )
> numerator_degrees_of_freedom = number_of_independent_variables
> denominator_degrees_of_freedom =
        sample_size - ( number_of_independent_variables + 1 )
> adjusted_explained_variability_ratio =
        coefficient_of_determination / numerator_degrees_of_freedom
> adjusted_unexplained_variability_ratio =
        ( 1 - coefficient_of_determination ) /
        denominator_degrees_of_freedom
> f_score =
        adjusted_explained_variability_ratio /
        adjusted_unexplained_variability_ratio
> f_score = summary( fit )[[ 10 ]][[ 1 ]]
> confidence_level = 0.95
> alpha = ( 1 - confidence_level ) / 2
```

```
> critical_f_value =
      qf(   1 - alpha,
            numerator_degrees_of_freedom,
            denominator_degrees_of_freedom )
>     if ( f_score > critical_f_value )
      {
            print( “reject, all the betas are significant” )
      }
> decision_confidence =
      pf(   f_score,
            numerator_degrees_of_freedom,
            denominator_degrees_of_freedom )
> p_value = 1 - decision_confidence
```

Confidence Interval

```
> f_min = function( x_point, z_point )
      {
            return ( f( x_point, z_point ) - 2 * residual_standard_deviation )
      }
> f_max = function( x_point, z_point )
      {
            return ( f( x_point, z_point ) + 2 * residual_standard_deviation )
      }
```

Multiple Regression With Interaction

```
> table = read.csv( file=”clock_auction.csv”, header=T )
> head( table )
> attach( table )
> pairs( price ~ age + bidders )
> x_array = age
> y_array = price
> z_array = bidders
> xz_array = I( age * bidders )
> fit = lm( y_array  ~ x_array  + y_array + z_array + xz_array )
> summary( fit )
> y_intercept = fit$coefficients[[ 1 ]]
> beta_x = fit$coefficients[[ 2 ]]
> beta_z = fit$coefficients[[ 3 ]]
> beta_xz = fit$coefficients[[ 4 ]]
> f = function( x_point, z_point )
      {
            return (          beta_x * x_point +
                              beta_z * z_point +
                              beta_xz * ( x_point * z_point ) +
                              y_intercept )
      }
```

Residual Standard Deviation

```
> residual_standard_deviation = summary( fit )[[ 6 ]]
```

As an acid test, 95% of the dependent points lie within
+- 2 * residual_standard_deviation of the best fit plane.

Coefficient of Determination

```
> coefficient_of_determination = summary( fit )$r.squared
> unexplained_variability_ratio = 1 - coefficient_of_determination
```

Is beta_xz significant?

Assume beta_xz is not significant.

```
> degrees_of_freedom = df.residual( fit )
> beta_xz_standard_deviation = summary( fit )[[ 4 ]][[ 8 ]]
> t_score = abs( beta_xz / beta_xz_standard_deviation )
> t_score = abs( summary( fit )[[ 4 ]][[ 12 ]] )
> confidence_level = 0.95
> alpha = ( 1 - confidence_level ) / 2
> critical_t_value = abs( qt( alpha, degrees_of_freedom ) )
>       if ( t_score > critical_t_value )
        {
                print( “reject, beta_xz is significant” )
        }
> decision_confidence =
        pt( t_score, degrees_of_freedom ) * 2 - 1
> print( sprintf( “%15.13f”, decision_confidence ) )
> p_value = 1 - decision_confidence
> p_value = summary( fit )[[ 4 ]][[ 16 ]]
> print( sprintf( “%15.13f”, p_value ) )
```

Confidence Interval

```
> f_min = function( x_point, z_point )
        {
                return ( f( x_point, z_point ) - 2 * residual_standard_deviation )
        }
> f_max = function( x_point, z_point )
        {
                return ( f( x_point, z_point ) + 2 * residual_standard_deviation )
        }
```

Estimate Home Price

Browser: http://ssl.sacbee.com/onboard/homes.html
Browser: <Select><Copy>
Spreadsheet: <Edit><Paste Special><HTML>
Spreadsheet: <File><Save As> home_sales_$zipcode.csv

$ home_sales_reformat.sh home_sales_$zipcode.csv home_sales_$zipcode_reformatted.csv

> table = read.csv(file="home_sales_$zipcode_reformatted.csv", header=T)
> source("home_sales.R")
> f_min($sq_feet, $bedrooms, $bathrooms);
f($sq_feet, $bedrooms, $bathrooms);
f_max($sq_feet, $bedrooms, $bathrooms)

Files

http://timriley.net/statistics
http://timriley.net/download/appaserver