OMR Exam System: Major formulas used in OMR2003

Prepared by Saša Bošnjak, May 2004

Standard Deviation:

Standard deviation

$$SD = \sqrt{\frac{\sum_{i=1}^{n} (Xi - \bar{X})^2}{n - 1}}$$

where

SD = Standard deviation

n = number of students in the class

Xi = score for student i

$\bar{X}$ = mean score for the class

The code that calculates the variance is:

```
float CTaskReport::CalculateVariance(vector<float> &data)
{
  if(data.size() < 2)
  {
    MessageBox(NULL, "The number of students is less then 2! The standard variance will not
be calculated!",
      "Warning", MB_OK | MB_SYSTEMMODAL);
    return 0.0;
  }

  float mean = 0.0;
  float variance = 0.0;

  for(vector<float>::iterator iter = data.begin(); iter != data.end(); iter++)
  {
    mean += *iter;
  }
```

```
  mean /= data.size();

  for(iter = data.begin(); iter != data.end(); iter++)
  {
    variance += (*iter - mean) * (*iter - mean);
  }

  return variance/(data.size() - 1);
}
```

The standard deviation is then calculated as a square root of variance.

Point biserial:

Point biserial

$$r_{pb} = \frac{Y_I - Y}{SD} \sqrt{\frac{n_I\, n}{(n - n_I)\,(n - 1)}}$$

where :

- n = total number of students
- n_I = number who answered questions correctly
- Y = mean score for all the students
- Y_I = mean score for those who answered question correctly
- SD = standard deviation

The code snippet that calculates point biserial is:

```
        // calculate point biserial for the question as
        // ((Yi - Y)/s)sqrt((Ni * N)/((N - Ni)(N - 1)))
        if(numCorrect != 0)
        {
          flMeanScoreCorrect /= numCorrect;
        }

        if(numTotalCount == numCorrect || numTotalCount == 1 || m_flStdDevScoreRaw == 0.0)
        {
          flPointBiserial = 0.0;
        }
        else
        {
          flPointBiserial = (((float)numCorrect * (float)numTotalCount) / (((float)numTotalCount -
(float)numCorrect)
              * ((float)numTotalCount - 1)));
          flPointBiserial = sqrt(flPointBiserial);
```

```
        flPointBiserial = ((flMeanScoreCorrect - m_flMeanScoreRaw) / m_flStdDevScoreRaw)
* flPointBiserial;
    }
```

PDiff:

PDiff = Number of correct answers / number of answers

The code snippet that calculates the PDiff coefficient is:

```
    //@PDIFF
    HRCALL(pXmldoc->createAttribute(_bstr_t("PDIFF"), &pAttr));

    if(!numTotalCount)
    {
      HRCALL(pAttr->put_value(CComVariant(0)));
    }
    else
    {
      temp.Format("%.3f", (float)numCorrect / (float)numTotalCount);
      HRCALL(pAttr->put_value(CComVariant(temp)));
    }
```

Percentile:

The percentile coefficient is defined as sum of scores for people with score less then current score and half of sum of scores for people with score equal to the current score The result is then divided by number of students. The percentile coefficient is calculated for each student in the class

Percentile :

$$\text{Percentile} = \frac{\sum_{i=1}^{L} S_i + \frac{1}{2} \sum_{j=1}^{E} S_j}{n}$$

where :

n = total number of students in the class

S_1 = Student with score less then current student

L = number of students with score less then current student

E = nuber of students with score equal to current student

S_j = student with score equal to the score for the current student

The code snippet that calculates the percentile coefficient is:

```
    // Calculate percentile
    // percentile is defined as add number of people with score less then current and add to it
    // half people with the same score. Divide the result with total number of students
    int below = 0;
    int equal = 0;

    for(int j = 1; j <= m_pExam->GetStudentExamCount(); j++)
    {
      CStudExam *pSEPerc = m_pExam->GetStudentExam(j);

      if(!pSEPerc)
      {
        throw COMRException("CTaskOutFinalReports::DoStudentsReport(), Cannot locate
student exam for percentile calculations!");
      }

      // skip if the version is not ok
      if(!pSE->IsOK())
      {
        continue;
      }

      if(pSE->GetScore() > pSEPerc->GetScore())
      {
        below++;
      }
      else if(pSE->GetScore() == pSEPerc->GetScore() && i != j)
      {
        equal++;
      }
    }
    // make sure you are not dividing by 0
    if(m_pExam->GetStudentExamCount() == m_pExam->GetNoOfExamsWithNoVersion())
    {
      throw COMRException("CTaskOutFinalReports::DoStudentsReport(), You have no exams
with correct version number!");
    }

    temp.Format("%.2f", ((below + 0.5 * equal) / (m_pExam->GetStudentExamCount() -
m_pExam->GetNoOfExamsWithNoVersion())) * 100);
    HRCALL(pTextNode->put_nodeValue(CComVariant(temp)));
    HRCALL(pSubStudentNode->appendChild(pTextNode, &pTextNode));
    HRCALL(pStudentNode->appendChild(pSubStudentNode, &pSubStudentNode));

    HRCALL(pTextNode->Release());
    HRCALL(pSubStudentNode->Release());
```

Z-Score:

Z - Score :

$$Z = \frac{S - M}{SD}$$

where :

Z = Z - Score

S = score for the student

M = mean score for the class

SD = Standard deviation

The code snippet that calculates Z-Score:

```
    //********** add zscore node
    // the individual z score is calculated as:
    // (Score - Mean)/StdDev
    HRCALL(pXmldoc->createNode(CComVariant(NODE_ELEMENT), _bstr_t("ZSCORE"),
_bstr_t(""), &pSubStudentNode));
    HRCALL(pXmldoc->createNode(CComVariant(NODE_TEXT), _bstr_t(""), _bstr_t(""),
&pTextNode));
    float zscore;

    if(m_flStdDevScoreRaw == 0.0)
    {
      zscore = 0.0;
    }
    else
    {
      zscore = (pSE->GetScore() - m_flMeanScoreRaw) / m_flStdDevScoreRaw;
    }

    temp.Format("%.2f", zscore);
    HRCALL(pTextNode->put_nodeValue(CComVariant(temp)));
    HRCALL(pSubStudentNode->appendChild(pTextNode, &pTextNode));
    HRCALL(pStudentNode->appendChild(pSubStudentNode, &pSubStudentNode));

    HRCALL(pTextNode->Release());
    HRCALL(pSubStudentNode->Release());
```

T-Score:

T-Score:

T = 50 + 10 Z
Where:

T = T-Score
Z = Z-Score

The code snippet that calculates the T score:

```
    //********** add tscore node
    // tscore is calculated as:
    // t = 50 + 10z
    // where t = t score
    // z = z score
    HRCALL(pXmldoc->createNode(CComVariant(NODE_ELEMENT), _bstr_t("TSCORE"),
_bstr_t(""), &pSubStudentNode));
    HRCALL(pXmldoc->createNode(CComVariant(NODE_TEXT), _bstr_t(""), _bstr_t(""),
&pTextNode));

    temp.Format("%.2f", 50 + 10 * zscore);
    HRCALL(pTextNode->put_nodeValue(CComVariant(temp)));
    HRCALL(pSubStudentNode->appendChild(pTextNode, &pTextNode));
    HRCALL(pStudentNode->appendChild(pSubStudentNode, &pSubStudentNode));

    HRCALL(pTextNode->Release());
    HRCALL(pSubStudentNode->Release());

    HRCALL(pChildNode->appendChild(pStudentNode, &pStudentNode));
    HRCALL(pRootNode->appendChild(pChildNode, &pChildNode));
    HRCALL(pStudentNode->Release());
```