Supplementary dataset 1: TAGGIT Microsoft Excel© macro.

```
'-------------------------------------------------------------------------
'
' TAGGIT Macro
'
' (c) Simon Holdsworth 2005, 2006
'
' This macro is free software; you can redistribute it and/or modify
' it as you please, but you must include this header in any distributed or
' modified version.
'
' Any useful additions to the macro should be submitted to the author.
'
' This package is distributed in the hope that it will be useful,
' but WITHOUT ANY WARRANTY; without even the implied warranty of
' MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
'
' Bug fixes, suggestions and comments should be sent to: hldswrth@hotmail.com
'
'-------------------------------------------------------------------------
'
' Instructions:
'
' Tag a list given a set of tags.  The search list may consist of one
' or more columns.  The tags are searched for within each of the columns in
' turn.  The first matching tag is placed in the next column after the search
' list.
'
' The macro also calculates the tag hit rate, and places a summary at the
' specified location.
'
' The currently selected cell and those to the right contain the parameters
' for the macro:
'
' Cell of 1st entry in search list
' Cell of 1st entry in tag list
' Cell of 1st entry in tag hit rate report (empty for no report)
' Number of columns in the search list
' Tags and corresponding search terms in rows 'R' or columns 'C'.
'
'-------------------------------------------------------------------------
'
' Updates:
'
' v7: 13/03/2006: Added search columns parameter to allow for more than two columns.
' v7(1): 27/07/2006: Added ability to have tags/search terms in columns or rows.
'
'-------------------------------------------------------------------------

' Main subroutine.
' Gets the parameters from the currently selected cell.
Sub Taggit()

    'On Error GoTo errorHandler

    Dim paramCell As Range
    Dim listCell As Range
    Dim listRange As Range
    Dim tagCell As Range
    Dim tagRange As Range
    Dim hitRateCell As Range
    Dim generateHitRate As Boolean
    Dim searchColumns As Integer
    Dim tagsAndTermsInColumns As Boolean

    ' Check that parameters are supplied.
    ' Assume that these are in the currently selected cell
    ' and those to the right.
    Set paramCell = ActiveCell
    If paramCell.Value = "" Or _
       paramCell.Offset(0, 1).Value = "" Then
        MsgBox "Error: current cell or the one to the right is empty"
        Exit Sub
    End If

    ' Get parameter values.
    Set listCell = Range(paramCell.Value)
    Set tagCell = Range(paramCell.Offset(0, 1).Value)

    If paramCell.Offset(0, 2).Value <> "" Then
        Set hitRateCell = Range(paramCell.Offset(0, 2).Value)
        generateHitRate = True
    End If

    searchColumns = 2
    If paramCell.Offset(0, 3).Value <> "" Then
        searchColumns = Int(Val(paramCell.Offset(0, 3).Value))
    End If

    tagsAndTermsInColumns = True
    If paramCell.Offset(0, 4).Value = "R" Then
        tagsAndTermsInColumns = False
    End If

    Set listRange = Range(listCell, listCell.Offset(-1, 0).End(xlDown))

    ' Tag range depends on whether tags and search terms are in rows or columns.
    If tagsAndTermsInColumns Then
        Set tagRange = Range(tagCell, tagCell.Offset(0, -1).End(xlToRight))
    Else
        Set tagRange = Range(tagCell, tagCell.Offset(-1, 0).End(xlDown))
    End If

    '  Generate the tags and hit rate.
    TagList listRange, tagRange, hitRateCell, generateHitRate, searchColumns, _
            tagsAndTermsInColumns

    Exit Sub

errorHandler:
    MsgBox "Error: did you make sure that the right sheet was active?"

End Sub

' Tag the given list and generate the hit rates if required.
Sub TagList(listRange As Range, _
            tagsRange As Range, _
            rateCell As Range, _
            generateRate As Boolean, _
```

```
          searchColumns As Integer, _
          tagsAndTermsInColumns As Boolean)

    ' Determine the number of tags, and create an array of
    ' collections to hold the search terms.
    Dim numTags As Integer
    If tagsAndTermsInColumns Then
        numTags = tagsRange.Columns.Count
    Else
        numTags = tagsRange.Rows.Count
    End If

    ReDim searchTerms(0 To numTags) As Collection
    ReDim tags(0 To numTags) As String

    ' Read in the tags and search terms.
    ReadTagsAndSearchTerms tagsRange, tags, searchTerms, numTags, tagsAndTermsInColumns

    ' Initialize the hit rate calculation
    If generateRate Then
        InitializeHitRate rateCell, listRange.Rows.Count, tags, numTags
    End If

    ' Tag the list
    TagUsingSearchTerms listRange, tags, searchTerms, numTags, rateCell, generateRate,
searchColumns

End Sub

' Read in the tags and search terms.
Sub ReadTagsAndSearchTerms(tagsRange As Range, _
                        tags, _
                        searchTerms, _
                        numTags As Integer, _
                        tagsAndTermsInColumns As Boolean)

    ReDim Preserve searchTerms(0 To numTags) As Collection
    ReDim Preserve tags(0 To numTags) As String

    xOffset = 0
    yOffset = 0
    If tagsAndTermsInColumns Then
        xOffset = 1
    Else
        yOffset = 1
    End If

    ' Go through the tags range pulling out the tag name and
    ' search terms.
    Dim tagNumber As Integer
    tagNumber = 0
    Dim terms As Collection

    For Each Cell In tagsRange.Cells
        tags(tagNumber) = Cell.Value

        Set terms = New Collection
        Set Cell = Cell.Offset(xOffset, yOffset)

        ' Store each search term.
        While Cell.Value <> ""

            ' Adjust the search term so that we can just look for
            ' it in an entry.
            term = Trim(Cell.Value)

            ' For now, strip off wildcards
            'If Right(term, 1) = "*" Then term = Left(term, Len(term) - 1)
            'If Left(term, 1) = "*" Then term = Right(term, Len(term) - 1)

            ' Check for a wildcard at the end.
            ' If no wildcard, add a space to force a complete word.
            If Right(term, 1) = "*" Then
                term = Left(term, Len(term) - 1)
            Else
                term = term & " "
            End If

            ' Check for a wildcard at the start.
            ' If no wildcard, add a space to force a complete word.
            If Left(term, 1) = "*" Then
                term = Right(term, Len(term) - 1)
            Else
                term = " " & term
            End If

            ' Lower case the search term.
            term = LCase(term)

            ' MsgBox "Adding search term '" & term & "' from value '" & Cell.Value & "'"

            terms.Add term
            Set Cell = Cell.Offset(xOffset, yOffset)
        Wend

        ' Store the set of search terms.
        Set searchTerms(tagNumber) = terms

        ' Look for the next tag.
        tagNumber = tagNumber + 1
    Next

End Sub

' Read in the tags and search terms.
Sub TagUsingSearchTerms(listRange As Range, _
                        tags, _
                        searchTerms, _
                        numTags As Integer, _
                        rateCell As Range, _
                        generateRate As Boolean, _
                        searchColumns As Integer)

    ReDim Preserve searchTerms(0 To numTags) As Collection
    ReDim Preserve tags(0 To numTags) As String
    Dim columnsSearched As Integer

    ' Set the Tags heading.
    listRange.Rows(0).Offset(0, searchColumns).Value = "Tag"
    listRange.Rows(0).Offset(0, searchColumns).Font.Bold = True
    ' Go through each cell in the list.
    For Each Cell In listRange.Cells

        ' Search each column in turn, starting with the leftmost.
        ' Stop as soon as a match has been found.
        columnsSearched = 0
        tagNumber = -1
        While tagNumber = -1 And columnsSearched < searchColumns
            tagNumber = FindFirstSearchTerm(Cell.Offset(0, columnsSearched).Value, numTags,
searchTerms)
            columnsSearched = columnsSearched + 1
        Wend

        ' If a tag was found, put the value in the end column.
        If tagNumber > -1 Then
            Cell.Offset(0, searchColumns).Value = tags(tagNumber)

            ' If generating hit rates, update the count for the tag.
            If generateRate Then
                rateValue = rateCell.Offset(tagNumber + 1, 1).Value
                If rateValue = "" Then
                    rateCell.Offset(tagNumber + 1, 1).Value = 1
                Else
                    rateCell.Offset(tagNumber + 1, 1).Value = Int(Val(rateValue)) + 1
                End If
            End If

        ' If not found, clear the cell.
        Else
            Cell.Offset(0, searchColumns).Value = ""
        End If
    Next

End Sub

' Find the first matching search term.
Function FindFirstSearchTerm(entry As String, _
                             numTags As Integer, _
                             searchTerms) As Integer

    ReDim Preserve searchTerms(0 To numTags) As Collection

    ' Add a space at the start and end of the entry, so that we don't
    ' have to special case hits at the ends.
    ' Also convert the entry into lower case so that we don't have to do
    ' a case-insensitive search, which is slower.
    '
    searchEntry = " " & LCase(entry) & " "

    ' Go through the search terms for each tag.
    ' There is a MUCH faster way to do this by constructing a
    ' search tree from all the search terms, rather than hunting
    ' through all the lists - maybe I'll implement that next.
    For i = 0 To numTags - 1
        Dim terms As Collection
        Set terms = searchTerms(i)
        For Each term In terms
            ' If the term is found, then return this tag index.
            ' Use a "textual" compare, i.e. case insensitive.
            If InStr(1, searchEntry, term, vbBinaryCompare) > 0 Then
                FindFirstSearchTerm = i
                Exit Function
            End If
        Next
    Next

    ' Return an indication that the term was not found
    FindFirstSearchTerm = -1

End Function

' Initialize the hit rate summary.
Sub InitializeHitRate(hitCell As Range, _
                      listSize As Integer, _
                      tags, _
                      numTags As Integer)

    ReDim Preserve tags(0 To numTags) As String

    ' Clear out the target.
    ' Clear out an extra 100 rows in case number of tags has reduced.
    Range(hitCell, hitCell.Offset(numTags + 104, 2)).Value = ""

    ' Initialise totals to zero.
    Range(hitCell.Offset(0, 1), hitCell.Offset(numTags, 1)).Value = 0

    ' Set up column titles.
    hitCell.Value = "Tags"
    hitCell.Font.Bold = True
    hitCell.Offset(0, 1).Value = "Count"
    hitCell.Offset(0, 1).Font.Bold = True

    ' Set up totals.
    hitCell.Offset(numTags + 1, 0).Value = "Unclassified"
    hitCell.Offset(numTags + 1, 1).Value = "=" & hitCell.Offset(numTags + 2, 1).Address & _
                                           "-" & hitCell.Offset(numTags + 3, 1).Address
    hitCell.Offset(numTags + 1, 0).Font.Bold = True
    hitCell.Offset(numTags + 1, 1).Font.Bold = True

    hitCell.Offset(numTags + 2, 0).Value = "Total genes in lists"
    hitCell.Offset(numTags + 2, 1).Value = listSize
    hitCell.Offset(numTags + 2, 0).Font.Bold = True
    hitCell.Offset(numTags + 2, 1).Font.Bold = True

    hitCell.Offset(numTags + 3, 0).Value = "Total genes classified above"
    hitCell.Offset(numTags + 3, 1).Value = "=SUM(" & _
                                           hitCell.Offset(1, 1).Address & ":" & _
                                           hitCell.Offset(numTags, 1).Address & ")"
    hitCell.Offset(numTags + 3, 0).Font.Bold = True
    hitCell.Offset(numTags + 3, 1).Font.Bold = True

    hitCell.Offset(numTags + 4, 0).Value = "Percentage classified"
    hitCell.Offset(numTags + 4, 1).Value = "=ROUND(" & hitCell.Offset(numTags + 3, 1).Address & _
                                           "/" & hitCell.Offset(numTags + 2, 1).Address & _
                                           " * 100,2)"
    hitCell.Offset(numTags + 4, 0).Font.Bold = True
    hitCell.Offset(numTags + 4, 1).Font.Bold = True

    ' Copy tag names.
    For i = 0 To numTags - 1
        hitCell.Offset(i + 1).Value = tags(i)
    Next

End Sub
```