```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<!-- stylesheet of the data format version 6 -->

 <xsl:template match="/">
  <html>
   <head>
    <title/>

<!-- xxxx  style definitions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxx -->

    <style type="text/css">
      h1 {font-family:Verdana; font-size:25px; color:#00c;}
      h3 {font-family:verdana; color:#00c}
      p {font-family:Verdana; font-size:14px; font-weight:bold;}
      span {font-family:Verdana; font-size:14px; font-weight:normal;}
      table {font-family:verdana; font-size:14px; border=0;
border-collapse:collapse;}
      th {text-align:left; border:1px solid gray; padding-left:5px;
padding-right:5px}
      th.without {text-align:left; border:0px; padding-left:5px;
padding-right:5px}
      th.without_ind {color:#00c; text-align:left; border:0px; padding-left:5px;
padding-right:5px}
      th.matrix {text-align:left; vertical-align:text-top; border:1px solid
gray; padding-left:0px; padding-right:5px}
      td {text-align:left; border:1px solid gray; padding-left:5px;
padding-right:5px}
      td.without {text-align:left; border:0px; padding-left:5px;
padding-right:5px}
      td.matrix {text-align:left; border:1px solid gray; padding-left:5px;
padding-right:5px;}
      td.genotype {font-family:Courier; font-size:18px; text-align:left;
border:1px solid gray; padding-left:5px; padding-right:5px}
      td.genotype2 {font-family:Courier; font-size:18px; text-align:left;
border:0px; padding-left:5px; padding-right:5px}
    </style>

   </head>

   <body>
    <xsl:apply-templates/>
   </body>
  </html>
 </xsl:template>



<!-- xxxx  header
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx -->

 <xsl:template match="header">
    <h1>
      <xsl:value-of select="@title"/>
    </h1>

    <table>
      <tr>
        <th class="without"> Organism:</th>
        <td class="without">
          <xsl:value-of select="organism"/>
        </td>
      </tr>

      <tr>
        <th class="without">Number of populations:</th>
```

```
        <td class="without">
          <xsl:value-of select="numPop"/>
        </td>
      </tr>

      <xsl:if test="numReads">
        <tr>
          <th class="without"> Number of reads: </th>
          <td class="without">
            <xsl:value-of select="numReads"/>
          </td>
        </tr>
      </xsl:if>

      <xsl:if test="aligned">
        <tr>
          <th class="without"> Are sequences aligned: </th>
          <td class="without">
            <xsl:value-of select="aligned"/>
          </td>
        </tr>
      </xsl:if>

      <xsl:if test="missing">
        <tr>
          <th class="without"> Symbol for missing data: </th>
          <td class="without">
            <xsl:value-of select="missing"/>
          </td>
        </tr>
      </xsl:if>

      <xsl:if test="gap">
        <tr>
          <th class="without"> Symbol for a gap: </th>
          <td class="without">
            <xsl:value-of select="gap"/>
          </td>
        </tr>
      </xsl:if>

      <xsl:if test="gameticPhase">
        <tr>
          <th class="without"> Gametic phase: </th>
          <td class="without"> known </td>
        </tr>
      </xsl:if>

      <xsl:if test="recessiveData">
        <tr>
          <th class="without"> recessive data: </th>
          <td class="without"> yes </td>
        </tr>
      </xsl:if>

   </table>
 </xsl:template>




<!-- xxxx  loci
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx -->

<xsl:template match="loci">
```

```
<h3>
 <br />
 Loci:
</h3>
<p>
  <xsl:if test="lociNum">
    Number of loci:
    <span>
      <xsl:value-of select="lociNum" />
      <br />
    </span>
 </xsl:if>

 <xsl:if test="lociDataType">
   Data type of loci:
   <span>
     <xsl:value-of select="lociDataType" />
     <br />
   </span>
 </xsl:if>
</p>

<table>
 <xsl:for-each select="locus">
  <tr>
    <th class="without_Ind">
      id: "<xsl:value-of select="@id" />"
    </th>
  </tr>

  <xsl:if test="locusDataType">
    <tr>
      <td class="without"> data type: </td>
      <td class="without">
        <xsl:value-of select="locusDataType" />
      </td>
    </tr>
  </xsl:if>

  <xsl:if test="locusChromosom">
    <tr>
      <td class="without"> at chromosom: </td>
      <td class="without">
        <xsl:value-of select="locusChromosom" />
      </td>
    </tr>
  </xsl:if>

  <xsl:if test="locusLocation">
    <tr>
      <td class="without"> location: </td>
      <td class="without">
        <xsl:value-of select="locusLocation" />
      </td>
    </tr>
  </xsl:if>

  <xsl:if test="locusGenic">
    <tr>
      <td class="without"> locus is genic: </td>
      <td class="without">
        <xsl:value-of select="locusGenic" />
      </td>
    </tr>
  </xsl:if>

  <xsl:if test="locusLength">
    <tr>
      <td class="without"> length: </td>
      <td class="without">
```

```
          <xsl:value-of select="locusLength" />
        </td>
      </tr>
    </xsl:if>

    <xsl:if test="locusLinks">
      <tr>
        <td class="without"> links: </td>
        <td class="without">
          <xsl:value-of select="locusLinks" />
        </td>
      </tr>
    </xsl:if>

    <xsl:if test="locusComments">
      <tr>
        <td class="without"> comments: </td>
        <td class="without">
          <xsl:value-of select="locusComments" />
        </td>
      </tr>
    </xsl:if>

    <tr>
      <td class="without">
        <br />
      </td>
    </tr>

   </xsl:for-each>
  </table>
</xsl:template>




<!-- xxxx  population
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxx -->

 <xsl:template match="population">
  <h3>
   <br />
   Population: <xsl:value-of select="@name"/>
  </h3>
  <p> Population size:
   <span>
    <xsl:value-of select="popSize" />
    <br />
   </span>

   <xsl:if test="popGeogCoord">
     Geographic coordination of the population:
     <span>
       <xsl:value-of select="popGeogCoord"/>
     </span>
     <br />
   </xsl:if>

   <xsl:if test="popLingGroup">
     Linguistic group of the population:
     <span>
       <xsl:value-of select="popLingGroup"/>
     </span>
     <br />
   </xsl:if>
```

```
<!-- xxxx loci begin for alignment (variable setting) xxxxxxxxxx -->

     <xsl:variable name="min">
       <xsl:for-each select="ind/read/start">
         <xsl:sort select="." />
         <xsl:if test="position() = 1">
           <xsl:value-of select="."/>
         </xsl:if>
       </xsl:for-each>
     </xsl:variable>

<!-- xxxxxxxxxxxxxx -->


   <xsl:if test="popNumReads">
     Number of reads:
     <span>
       <xsl:value-of select="popNumReads"/>
     </span>
     <br />
   </xsl:if>


   <xsl:if test="popLocus">
     locus:
     <span>
       <xsl:value-of select="popLocus" />
     </span>
     <br />
     <xsl:if test="ind/read/start">
       locus start:
       <span>
         <xsl:value-of select="$min" />
       </span>
       <br />
     </xsl:if>
   </xsl:if>


<!-- xxxx variable if indFreq tag exist or not xxxxxxxx -->

   <xsl:variable name="Freq">
     <xsl:choose>
       <xsl:when test="ind/indFreq">
         1
       </xsl:when>
       <xsl:otherwise>
         0
       </xsl:otherwise>
     </xsl:choose>
   </xsl:variable>

<!-- xxxxxxxxxxxxxxxxxxxxxx -->


  </p>
   <table>
     <xsl:apply-templates select="id" />

     <xsl:apply-templates select="ind">
       <xsl:with-param name="minimum" select="$min + 1" />
       <xsl:with-param name="frequency" select="$Freq" />
     </xsl:apply-templates>
   </table>
 </xsl:template>


<!--
 xxxx  id
```

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

 <xsl:template match="id">
   <xsl:if test="position()=1">
     <tr>
       <th>identifier</th>
       <th>frequency</th>
         <xsl:if test="data">
           <th>
             data

              xxx test if the popLoci siblings before the current node exist xxx

             <xsl:if test="preceding-sibling::popLoci">
               <span>
                 <br />
                 (<xsl:value-of select="preceding-sibling::popLoci" />)
               </span>
             </xsl:if>
           </th>
        </xsl:if>
     </tr>
   </xsl:if>

   <tr>
    <td> <xsl:value-of select="@name"/> </td>
    <td> <xsl:value-of select="idFreq" /> </td>
    <xsl:if test="data">
      <td class="genotype">
        <xsl:for-each select="data">
          <xsl:value-of select="." />
          <br />
        </xsl:for-each>
      </td>
    </xsl:if>
   </tr>
 </xsl:template>

-->



<!-- xxxx  ind
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx -->

 <xsl:template match="ind">
  <xsl:param name="minimum" />
  <xsl:param name="frequency" />



<!-- xxx align over one individual (different loci) xxxxxxxxxxxxxxxxxxxx-->

   <xsl:choose>
     <xsl:when test="indLocus|indLoci">
       <tr>
         <th class="without_Ind"> Individual <xsl:value-of select="@name" />:
</th>
       </tr>

       <xsl:if test="indGeogCoord">
         <tr>
           <td class="without"> geographic coordination: </td>
           <td class="without">
             <xsl:value-of select="indGeogCoord" />
           </td>
         </tr>
       </xsl:if>
```

```
        <xsl:if test="indLingGroup">
          <tr>
            <td class="without"> linguistic group: </td>
            <td class="without">
              <xsl:value-of select="indLingGroup" />
            </td>
          </tr>
        </xsl:if>

        <xsl:if test="indLocus">
          <tr>
            <td class="without"> locus: </td>
            <td class="without">
              <xsl:value-of select="indLocus" />
            </td>
          </tr>
        </xsl:if>


<!-- xxxx loci start for alignment (variable setting) xxxxxxx -->

        <xsl:variable name="min">
          <xsl:for-each select="read/start">
            <xsl:sort select="." />
            <xsl:if test="position() = 1">
              <xsl:value-of select="."/>
            </xsl:if>
          </xsl:for-each>
        </xsl:variable>

<!-- xxxxxxxxxxxxxxxxxxx -->

        <xsl:if test="$min &gt; 0">
          <tr>
            <td class="without"> locus start: </td>
            <td class="without">
              <xsl:value-of select="$min" />
            </td>
          </tr>
        </xsl:if>

        <xsl:if test="indFreq">
          <tr>
            <td class="without"> frequency of this individual: </td>
            <td class="without">
              <xsl:value-of select="indFreq" />
            </td>
          </tr>
        </xsl:if>

        <xsl:if test="indNumReads">
          <tr>
            <td class="without"> number of strains: </td>
            <td class="without">
              <xsl:value-of select="indNumReads" />
            </td>
          </tr>
        </xsl:if>


        <tr>
          <td class="without"> data: </td>



<!-- xxxx data alignment xxxxxxxxxxxxxxxx -->
          <td class="genotype2">
            <xsl:choose>
              <xsl:when test="read">
```

```
              <xsl:apply-templates select="read">
                <xsl:with-param name="minimum" select="$min + 1" />
              </xsl:apply-templates>
            </xsl:when>

            <!-- aligned data -->
            <xsl:otherwise>
              <xsl:for-each select="data">
                <xsl:value-of select="." />
                <br />
              </xsl:for-each>
            </xsl:otherwise>
          </xsl:choose>
        </td>
      </tr>

      <tr>
        <td class="without">
          <br />
        </td>
      </tr>

    </xsl:when>



<!-- xxx align over one population (ind same loci) xxxxxxxxxxxxxxxxxxxxxx-->

    <xsl:otherwise>

      <xsl:if test="position()=1">
        <tr>
          <th>individual</th>

          <xsl:if test="indGeogCoord">
            <th> geographic coordination </th>
          </xsl:if>

          <xsl:if test="indLingGroup">
            <th> linguistic group </th>
          </xsl:if>

          <xsl:if test="$frequency = 1">
            <th> frequency </th>
          </xsl:if>

          <xsl:if test="indNumReads">
            <th> number of reads </th>
          </xsl:if>

          <xsl:if test="data">
            <th>
              data

              <!-- xxx test if the popLoci siblings before the current node
exist xxx -->
              <xsl:if test="preceding-sibling::popLoci">
                <span>
                  <br />
                  (<xsl:value-of select="preceding-sibling::popLoci" />)
                </span>
              </xsl:if>
            </th>
          </xsl:if>

          <xsl:if test="read/data">
            <th> data </th>
          </xsl:if>
```

```
        </tr>
      </xsl:if>

      <tr>
        <td> <xsl:value-of select="@name"/> </td>

        <xsl:if test="indGeogCoord">
          <td>
            <xsl:value-of select="indGeogCoord" />
          </td>
        </xsl:if>

        <xsl:if test="indLingGroup">
          <td>
            <xsl:value-of select="indLingGroup" />
         </td>
        </xsl:if>

        <xsl:if test="$frequency = 1">
          <td>
            <xsl:choose>
              <xsl:when test="indFreq">
                <xsl:value-of select="indFreq" />
              </xsl:when>
              <xsl:otherwise>
                1
              </xsl:otherwise>
            </xsl:choose>
         </td>
        </xsl:if>

        <xsl:if test="indNumReads">
          <td>
            <xsl:value-of select="indNumReads" />
          </td>
        </xsl:if>

        <xsl:if test="data">
          <td class="genotype">
            <xsl:for-each select="data">
              <xsl:value-of select="." />
              <br />
            </xsl:for-each>
          </td>
        </xsl:if>

        <xsl:if test="read">
          <td class="genotype">
            <xsl:apply-templates select="read">
              <xsl:with-param name="minimum" select="$minimum" />
            </xsl:apply-templates>
          </td>
        </xsl:if>

      </tr>

    </xsl:otherwise>
  </xsl:choose>
</xsl:template>



<!-- xxxx template read xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx -->

 <xsl:template match="read">
  <xsl:param name="minimum" />

      <xsl:call-template name="align">
        <xsl:with-param name="min" select="$minimum" />
        <xsl:with-param name="beginning" select="start" />
```

```
       </xsl:call-template>

       <xsl:value-of select="data" />
       <br />

 </xsl:template>



<!-- xxxx template align (for alignment) xxxxxxxxxxxxxxx -->

 <xsl:template name="align">
  <xsl:param name="min" />
  <xsl:param name="beginning" />

   <xsl:if test="$beginning &gt; $min"> <xsl:call-template name="align">
       <xsl:with-param name="min" select="$min" />
       <xsl:with-param name="beginning" select="$beginning - 1" />

     </xsl:call-template>
   </xsl:if>
 </xsl:template>







<!-- xxxx  structure
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxx -->

 <xsl:template match="structure">
  <h3>
    <br />
    Structure: <xsl:value-of select="@name"/>
  </h3>
  <p>

    <xsl:if test="numGroups">
      Number of groups:
      <span>
        <xsl:value-of select="numGroups"/>
      </span>
      <br />
    </xsl:if>

    <xsl:if test="group">
      <xsl:for-each select="group">
        <xsl:number value="position()" format="1" />. group:
        <span>
          <xsl:value-of select="."/>
        </span>
        <br />
      </xsl:for-each>
    </xsl:if>

  </p>
 </xsl:template>






<!-- xxxx  distance matrix
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
xxxxxxx -->

 <xsl:template match="distanceMatrix">
   <h3>
     <br />
     Distance matrix: <xsl:value-of select="@name"/>
   </h3>
   <p>

     <xsl:if test="matrixSize">
       Size of the distance matrix:
       <span>
         <xsl:value-of select="matrixSize"/>
       </span>
       <br />
     </xsl:if>

     Distance matrix:
   </p>

   <table>
     <xsl:call-template name="cr2br">
       <xsl:with-param name="text" select="matrix" />
       <xsl:with-param name="label" select="matrixLabels" />
     </xsl:call-template>

     <td class="without" />
     <xsl:call-template name="cr2Komma2">
       <xsl:with-param name="text" select="matrixLabels" />
     </xsl:call-template>

   </table>
   <br />
 </xsl:template>



<!-- xxxx  cr2br: seperate string after "line break"  xxxxxxxxxxxxxx -->

 <xsl:template name="cr2br">
  <xsl:param name="text" />
  <xsl:param name="label" />

  <xsl:choose>
   <xsl:when test="contains($text, '&#xA;')">
    <tr>

      <xsl:choose>
        <xsl:when test="contains($label, ',')">
          <th class="without">
            <xsl:value-of select="substring-before($label, ',')" />
          </th>
        </xsl:when>
        <xsl:otherwise>
          <th class="without">
            <xsl:value-of select="$label" />
          </th>
        </xsl:otherwise>
      </xsl:choose>


      <xsl:call-template name="cr2Komma">
        <xsl:with-param name="text" select="substring-before($text, '&#xA;')" />
      </xsl:call-template>

    </tr>

    <xsl:call-template name="cr2br">
      <xsl:with-param name="text" select="substring-after($text, '&#xA;')" />
      <xsl:with-param name="label" select="substring-after($label, ',')" />
```

```
    </xsl:call-template>

   </xsl:when>
   <xsl:otherwise>

     <tr>

       <th class="without">
         <xsl:value-of select="$label" />
       </th>

       <xsl:call-template name="cr2Komma">
         <xsl:with-param name="text" select="$text" />
       </xsl:call-template>
     </tr>

   </xsl:otherwise>
  </xsl:choose>
 </xsl:template>



<!-- xxxx  cr2br: seperate string after "comma"  xxxxxxxxxxxxx -->

 <xsl:template name="cr2Komma">
  <xsl:param name="text" />

  <xsl:choose>
   <xsl:when test="contains($text, ',')">

     <td class="without">
       <xsl:value-of select="substring-before($text, ',')" />
     </td>

     <xsl:call-template name="cr2Komma">
       <xsl:with-param name="text" select="substring-after($text, ',')" />
     </xsl:call-template>
   </xsl:when>
   <xsl:otherwise>

     <td class="without">
       <xsl:value-of select="$text" />
     </td>

   </xsl:otherwise>
  </xsl:choose>
 </xsl:template>



<!-- xxxx  cr2br: seperate string after "comma" (for header)  xxxxxxxxxxxxx -->

 <xsl:template name="cr2Komma2">
  <xsl:param name="text" />
  <xsl:choose>
   <xsl:when test="contains($text, ',')">


     <th class="without">
       <xsl:value-of select="substring-before($text, ',')" />
     </th>

     <xsl:call-template name="cr2Komma2">
       <xsl:with-param name="text" select="substring-after($text, ',')" />
     </xsl:call-template>
   </xsl:when>
   <xsl:otherwise>

     <th class="without">
       <xsl:value-of select="$text" />
```

```
      </th>

    </xsl:otherwise>
   </xsl:choose>
  </xsl:template>


</xsl:stylesheet>
```