Distruct:
a program for the graphical display of population structure

Noah A. Rosenberg [1]

Center for Computational Medicine and Biology
Department of Human Genetics
University of Michigan
100 Washtenaw Ave
Ann Arbor MI 48109, USA

The *distruct* software[2] is available at
`http://rosenberglab.bioinformatics.med.umich.edu/distruct.html`

June 27, 2007

[1]Comments can be emailed to me at rnoah@umich.edu.
[2]This manual is the companion to version 1.1 of the software.

Contents

1 Introduction

The clustering software *structure* (Pritchard *et al.*, 2000a; Falush *et al.*, 2003) provides an iterative algorithm that places individuals into K clusters. K is a parameter that is chosen in advance, but that can be varied in independent *structure* runs. Individuals are given "membership coefficients" for each cluster, such that the estimated membership coefficients of an individual sum to 1 across the K clusters. The matrix of membership coefficients, where the number of individuals is the number of rows and K is the number of columns, is referred to here as the *individual Q-matrix*. For each population, membership coefficients for each cluster can be averaged across individuals to create a *population Q-matrix*.

A convenient way to display *structure* results is to show each individual as a line segment. This segment is partitioned into K colored components, which represent the individual's estimated membership coefficients in the K clusters. The program *distruct* provides a variety of options for creating figures based on this general idea.

It is possible to make *distruct* figures without having used *structure* to generate the individual Q-matrix and the population Q-matrix; if other programs are used, the output from those programs must simply be formatted to match the *distruct* input. For the remainder of the manual, I assume that *structure* was in fact used to generate the input for *distruct*.

1.1 Availability

The *distruct* program was first used to generate Figures 1 and 2 in ROSENBERG *et al.* (2002). The website for *distruct* is `http://rosenberglab.bioinformatics.med.umich.edu/distruct.html`. When using *distruct*, please cite:

N. A. Rosenberg (2004). *Distruct*: a program for the graphical display of population structure. *Molecular Ecology Notes* 4: 137-138.

The *structure* software is available at `http://pritch.bsd.uchicago.edu`. Some familiarity with *structure* is assumed in this document. Users of *structure* and *distruct* may also be interested in *CLUMPP*, available at `http://rosenberglab.bioinformatics.med.umich.edu/clumpp.html`.

1.2 Basic overview

The *distruct* program is written in C, and compiled versions are available for Linux, MacOSX, and Windows. It reads data files based on output from *structure*. It allows additional optional files that permit the user to control the left-to-right order in which populations are displayed, the labels printed atop and/or below the figure, the bottom-to-top order of clusters, and the colors used. Output is printed in PostScript format, and the figure produced can be displayed using such programs as GhostView. If the figure is unsatisfactory, changes can often be made directly to the PostScript code without having to rerun *distruct* (indeed, a look at the PostScript can often help if the program does not seem to be doing what you want it to do).

Program settings are specified in the file *drawparams*, although some can be given with command-line arguments. Variables in all-capitals in this document are used in *drawparams*.

2 Input files

The data to be plotted are specified in ASCII text-formatted files derived from *structure* output. To allow for ongoing changes in the *structure* code, the *structure* output file itself is not used. Instead, the program takes a file with the population Q-matrix (required), and a separate file with the individual Q-matrix (optional), both printed in the format of the *structure* output. From *structure* output files, these files are easily produced, for example, by cutting and pasting.

The example input file used in this document is modified from the $K = 5$ graph shown for populations of Central/South Asia in Figure 2 of ROSENBERG *et al.* (2002). Along with the *distruct* code, the *distruct* package contains seven files related to this data set, as well as the file *drawparams*. The file *casia_f* is an output file from *structure*. The file *casia.ps* is the output of *distruct* when applied to data and settings in the other five files, which appear in Tables 1-5. A variety of color schemes for *distruct* plots are available in the ColorBrewer directory.

2.1 Population Q-matrix file

If NUMPOPS is the number of predefined populations and K is the number of clusters, *distruct* expects a file with NUMPOPS rows and $K + 2$ columns per row. This file is stored in INFILE_POPQ. Blank lines and extra whitespace are tolerated. In the example in Table 1, NUMPOPS=9 and $K = 5$. In Table 1, each row represents results for one population. The first

Table 1: Population Q-matrix (*casia.popq*).

50:	0.646	0.276	0.060	0.012	0.007	25
51:	0.900	0.062	0.018	0.011	0.009	25
52:	0.024	0.065	0.874	0.023	0.014	25
54:	0.058	0.133	0.139	0.661	0.009	25
56:	0.022	0.024	0.012	0.004	0.938	25
57:	0.572	0.337	0.064	0.013	0.015	25
58:	0.156	0.574	0.183	0.059	0.027	25
59:	0.234	0.585	0.143	0.021	0.016	25
629:	0.053	0.147	0.282	0.493	0.025	10

Table 2: Individual Q-matrix (*casia.indivq*).

1	1297	(3)	629	:	0.173	0.194	0.014	0.499	0.121
2	1298	(4)	629	:	0.077	0.048	0.377	0.497	0.002
3	1299	(4)	629	:	0.007	0.008	0.571	0.398	0.016
4	1300	(2)	629	:	0.076	0.453	0.018	0.442	0.010
5	1301	(4)	629	:	0.016	0.046	0.511	0.400	0.027
...									
206	203	(3)	59	:	0.005	0.977	0.006	0.004	0.008
207	205	(2)	59	:	0.263	0.616	0.038	0.004	0.078
208	206	(13)	59	:	0.218	0.654	0.016	0.054	0.058
209	208	(2)	59	:	0.499	0.025	0.424	0.029	0.023
210	210	(3)	59	:	0.050	0.877	0.017	0.014	0.041

column is an integer that gives a code number for the population and is followed by a colon. The next K columns are membership coefficients for clusters 1, 2, ..., K (real numbers in [0,1]). Ideally the numbers in these K columns sum to 1; in case they do not, the program normalizes them by their sum. The final column gives the sample size for the population (an integer).

While most applications will probably show the individual Q-matrix, some applications might wish to show only the population Q-matrix (for example, Table 2 of Wilson *et al.* (2001)). Thus, if PRINT_INDIVS is set to zero, *distruct* will display only the population Q-matrix.

2.2 Individual Q-matrix file

If NUMINDS is the number of individuals and PRINT_INDIVS is set to 1, *distruct* expects a file with NUMINDS rows and at least $K+6$ columns per row. This file is stored in INFILE_INDIVQ. Blank lines and extra whitespace are tolerated. Also, some options of *structure* print additional columns for confidence intervals; these columns are ignored. In the example shown in Table 2, NUMINDS=210. Each row shows the membership coefficients for one individual. Column 2 gives a code number for the individual. Column 4 gives the code number for the population to which the individual belongs. Columns 1, 3, and 5 are ignored. Columns 6 to $K+5$ show membership coefficients for clusters 1, 2, ..., K. Ideally the numbers in these K columns sum to 1; in case they do not, the program normalizes them by their sum.

Individuals are automatically grouped by population, with the left-to-right order of individuals in the figure being the same as the top-to-bottom order of individuals in the input file. Examples that show the individual Q-matrix are in Figures 1 and 2 of Rosenberg *et al.* (2002).

Table 3: Labels below the figure (*casia.names*).

50	Balochi
51	Brahui
57	Makrani
59	Sindhi
58	Pathan
52	Burusho
54	Hazara
629	Uygur
699	Yakut
56	Kalash

Table 4: Labels atop the figure (*casia.languages*).

50	Indo-European
51	Dravidian
57	Indo-European
59	Indo-European
58	Indo-European
52	Linguistic isolate
54	Indo-European
629	Altaic
699	Altaic
56	Indo-European

2.3 Labels below the figure

To place labels below the figure, set PRINT_LABEL_BELOW to 1. The program will search for the file specified by INFILE_LABEL_BELOW. The default is to print the population codes as labels. If the file is found, the input order of populations on the rows of the file will be used for the left-right order of graphing of populations. The first column contains the population code (an integer); the remaining columns contain the text that is to be printed below the figure.

Note that additional populations not found in the data can be included in the file. These additional lines will simply be ignored.

2.4 Labels atop the figure

Labels atop the figure are placed analogously to those below it using PRINT_LABEL_ATOP=1. The program will search for the file specified by INFILE_LABEL_ATOP. The default is to print the population codes as labels. If the file is found, the input order of populations on the rows of the file will be used for the left-right order of graphing of populations. If labels are requested both atop and below the figure, the entries in INFILE_LABEL_ATOP and INFILE_LABEL_BELOW should be listed in the same order. The first column contains the population code (an integer); the remaining columns contain the text that is to be printed atop the figure. As with the labels below the figure, this text can consist of multiple columns.

2.5 Vertical cluster order and cluster colors

The permutation that describes the vertical order of the clusters can be given in the file specified INFILE_CLUST_PERM. The colors to be used for the figure are also specified here. If no file is detected, the left-to-right order of the clusters in INFILE_POPQ is used. The names of some of the allowed colors are shown in Figure 1. Colors may vary considerably across printers. The first column of the file must contain a permutation of the integers in $\{1, 2...K\}$. The entries in the second column must be colors taken from the (case-insensitive) allowed list.

The maximum number of clusters is set at 60, and if no INFILE_CLUST_PERM file is specified, then the colors for the clusters will be taken in order from the top two rows of Figure 1. If GRAYSCALE is set to 1, real numbers in [0,1] are entered in place of the names of colors. In this scheme, 0 corresponds to black and 1 corresponds to white.

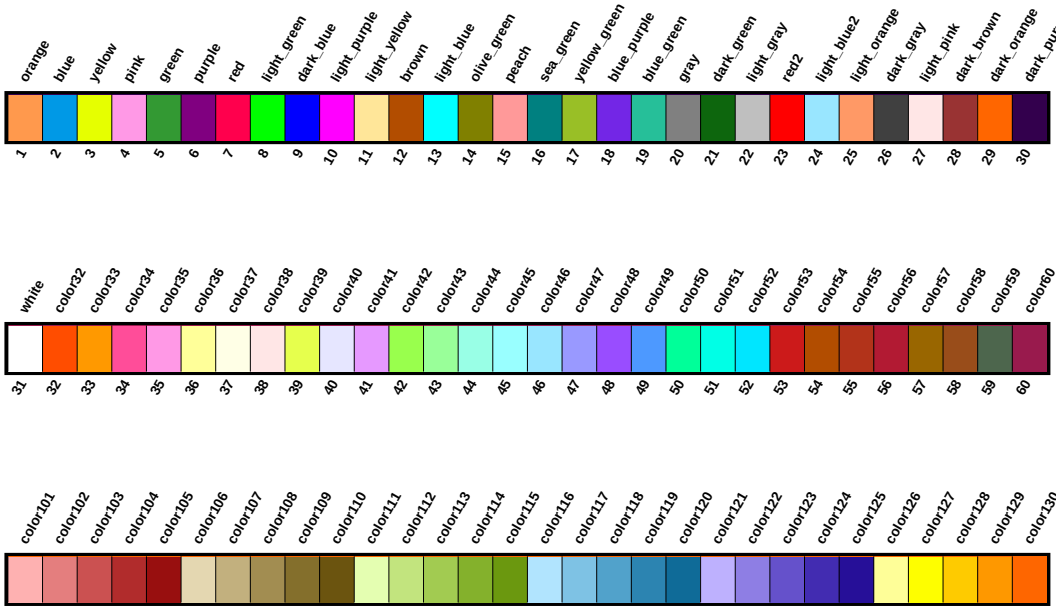

Figure 1: List of colors recognized by *distruct*. In the top two rows, the number below the figure represents the cluster for which the color will be used as the default. The label atop the figure is the string that is recognized as the color name. The string "black" is also recognized. The bottom row represents a stepped-sequential color scheme adapted from the scheme at `http://geography.uoregon.edu/datagraphics/color_scales.htm`.

Table 5: Permutation of the clusters, and colors (*casia.perm*).

5	yellOw
4	Pink
1	Red
2	green
3	blue_Purple

The *distruct* program also recognizes color schemes implemented in ColorBrewer (Brewer *et al.*, 2003; Harrower and Brewer, 2003). These color schemes, which contain 3-12 colors, are shown in Figures 2-9. A *distruct* figure can be produced using a particular ColorBrewer color scheme, or a color scheme can be created from a combination of the colors shown in Figures 1-9. ColorBrewer schemes are of three types — qualitative (Figures 2 and 3), diverging (Figures 4 and 5), and sequential (Figures 6-9). Qualitative schemes are likely to be best-suited to most depictions of population structure. As an example, to use the ColorBrewer color scheme *Accent_5_qual* instead of the scheme in *casia.perm*, replace *casia.perm* with *ColorBrewer/Accent_5_qual* for the value of INFILE_CLUST_PERM in the *drawparams* file. The rows in *Accent_5_qual* can be rearranged to produce the same bottom-to-top cluster order as in *casia.perm*.

The ColorBrewer website (`http://colorbrewer.org`) provides additional information about each of its schemes, including their suitability for colorblind viewers. A simulator for colorblind viewers is located at (`http://vischeck.com`).

Figure 2: ColorBrewer qualitative color schemes with 3-8 colors. Each row of graphs represents a different family of color schemes, and each column represents a different number of colors. The labels below the graphs represent the color names recognized by *distruct*.

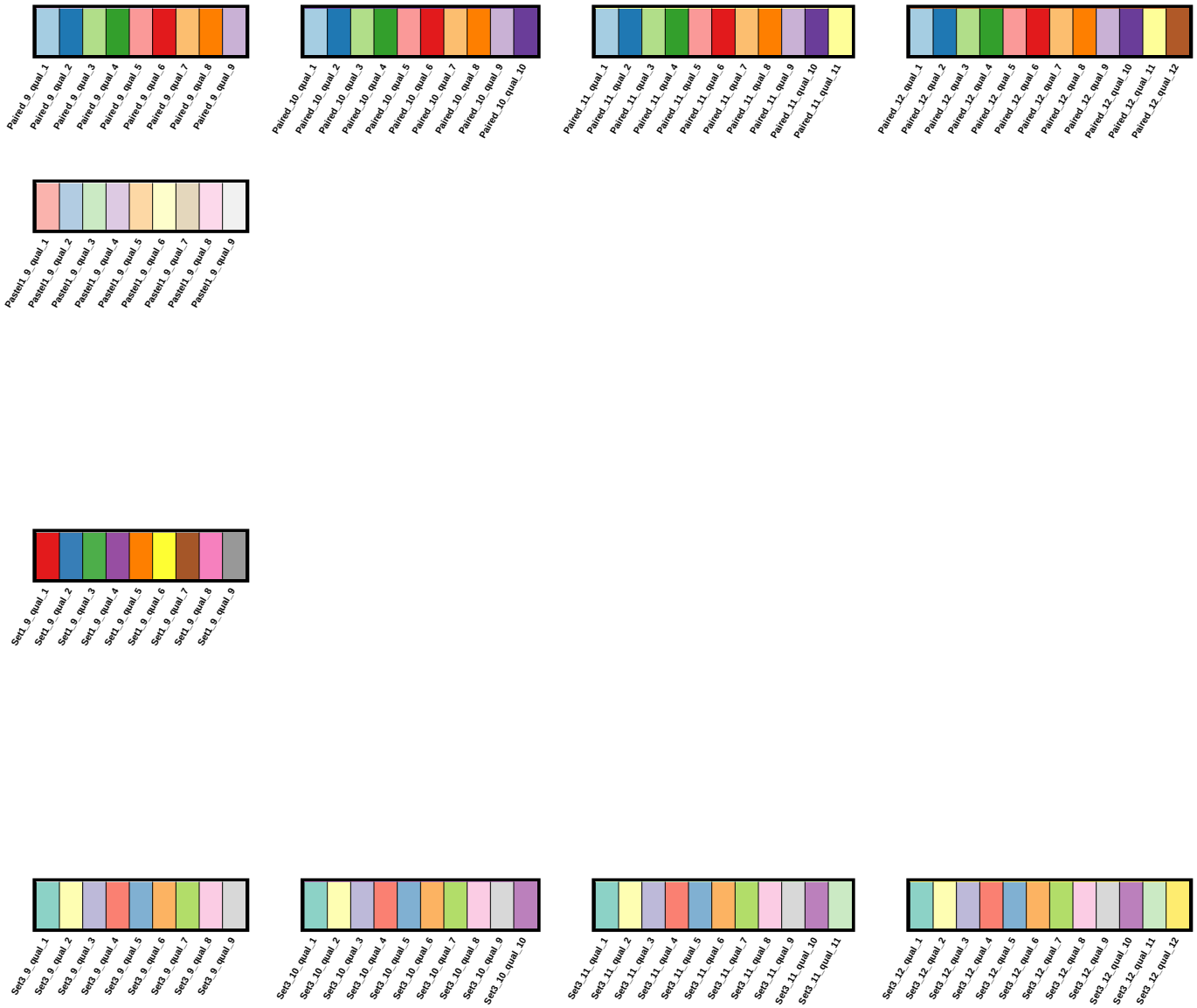

Figure 3: ColorBrewer qualitative color schemes with 9-12 colors (continued from the schemes with 3-8 colors in Figure 2). Each row of graphs represents a different family of color schemes, and each column represents a different number of colors. The labels below the graphs represent the color names recognized by *distruct*.

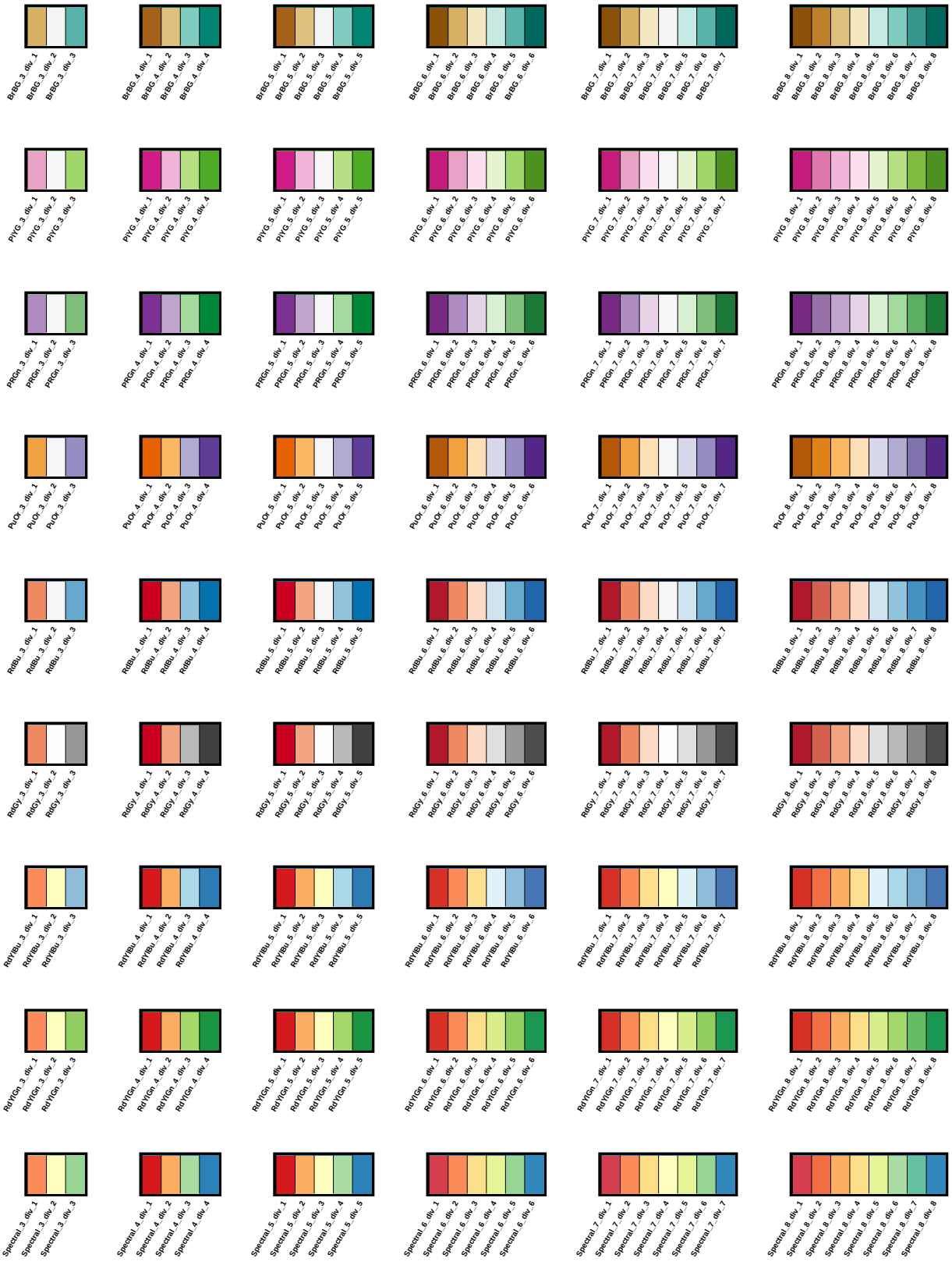

Figure 4: ColorBrewer diverging color schemes with 3-8 colors. Each row of graphs represents a different family of color schemes, and each column represents a different number of colors. The labels below the graphs represent the color names recognized by *distruct*.

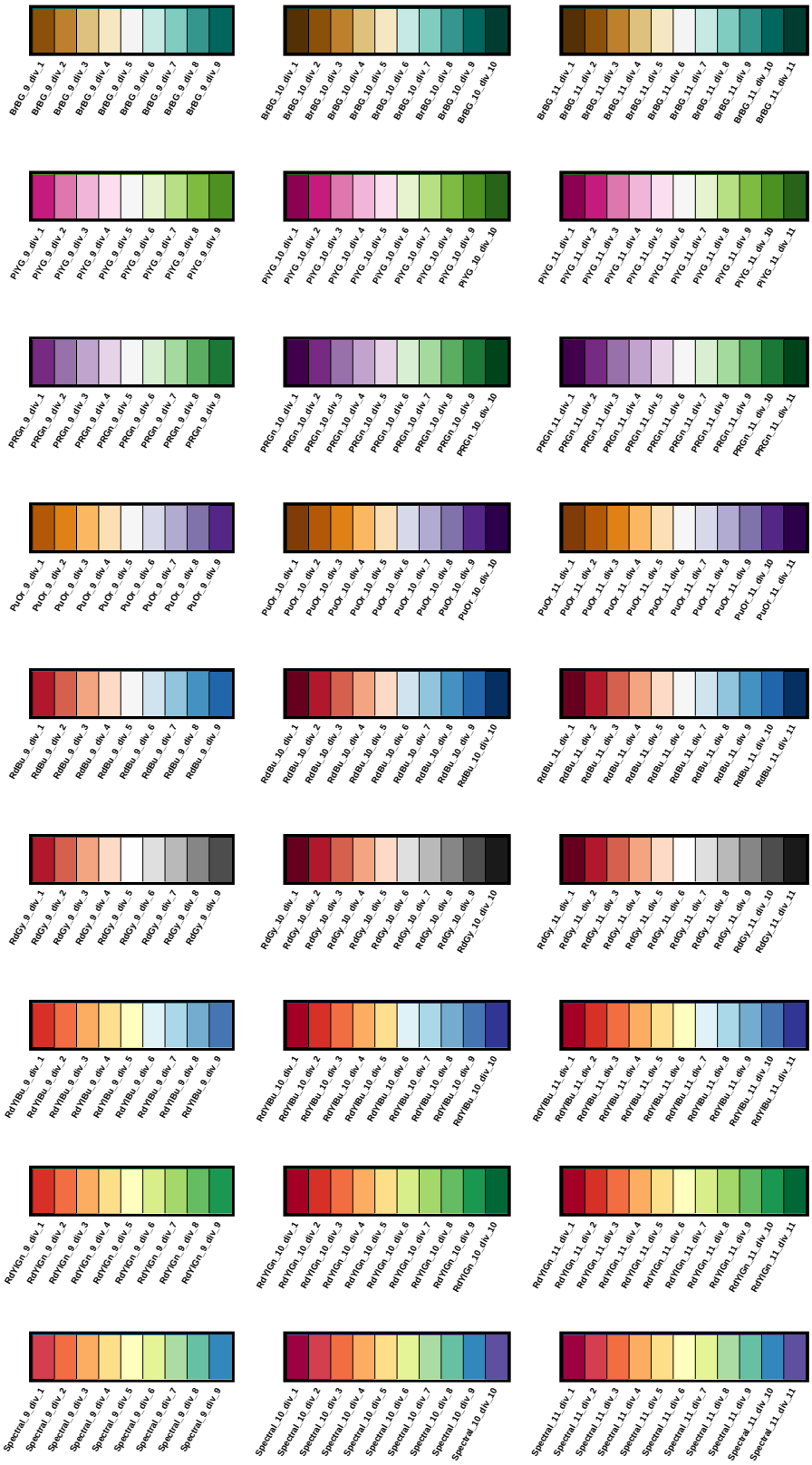

Figure 5: ColorBrewer diverging color schemes with 9-11 colors (continued from the schemes with 3-8 colors in Figure 4). Each row of graphs represents a different family of color schemes, and each column represents a different number of colors. The labels below the graphs represent the color names recognized by *distruct*.

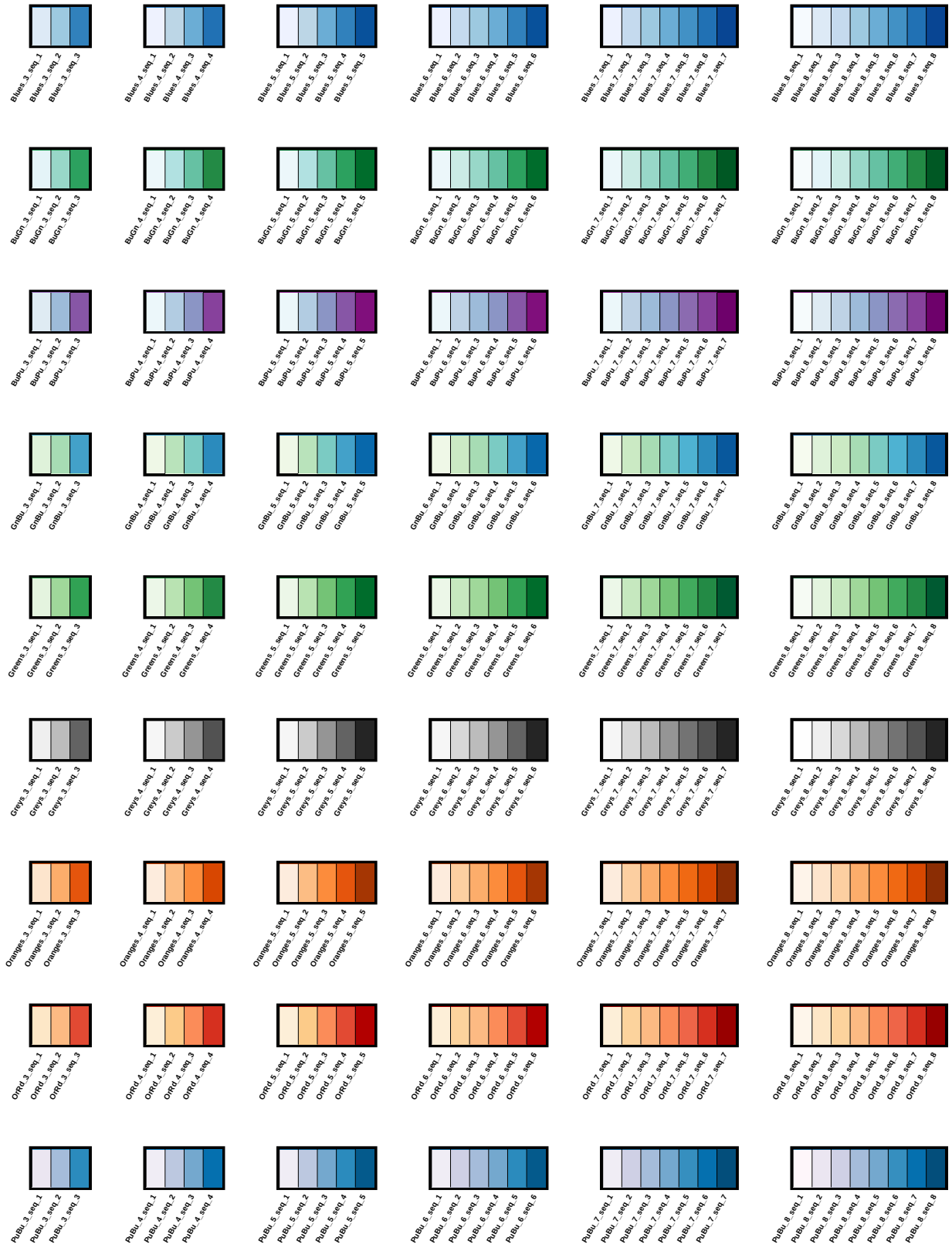

Figure 6: ColorBrewer sequential color schemes with 3-8 colors. Each row of graphs represents a different family of color schemes, and each column represents a different number of colors. The labels below the graphs represent the color names recognized by *distruct*.

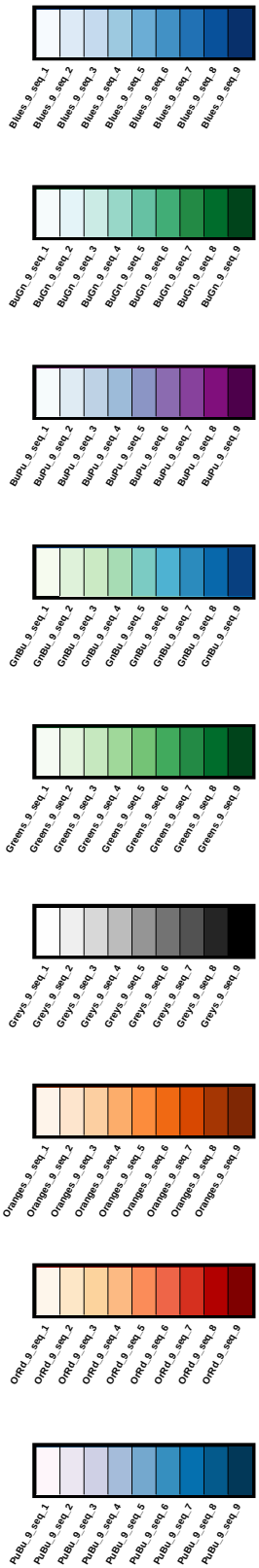

Figure 7: ColorBrewer sequential color schemes with 9 colors (continued from the schemes with 3-8 colors in Figure 6). Each row of graphs represents a different family of color schemes, and each column represents a different number of colors. The labels below the graphs represent the color names recognized by *distruct*.

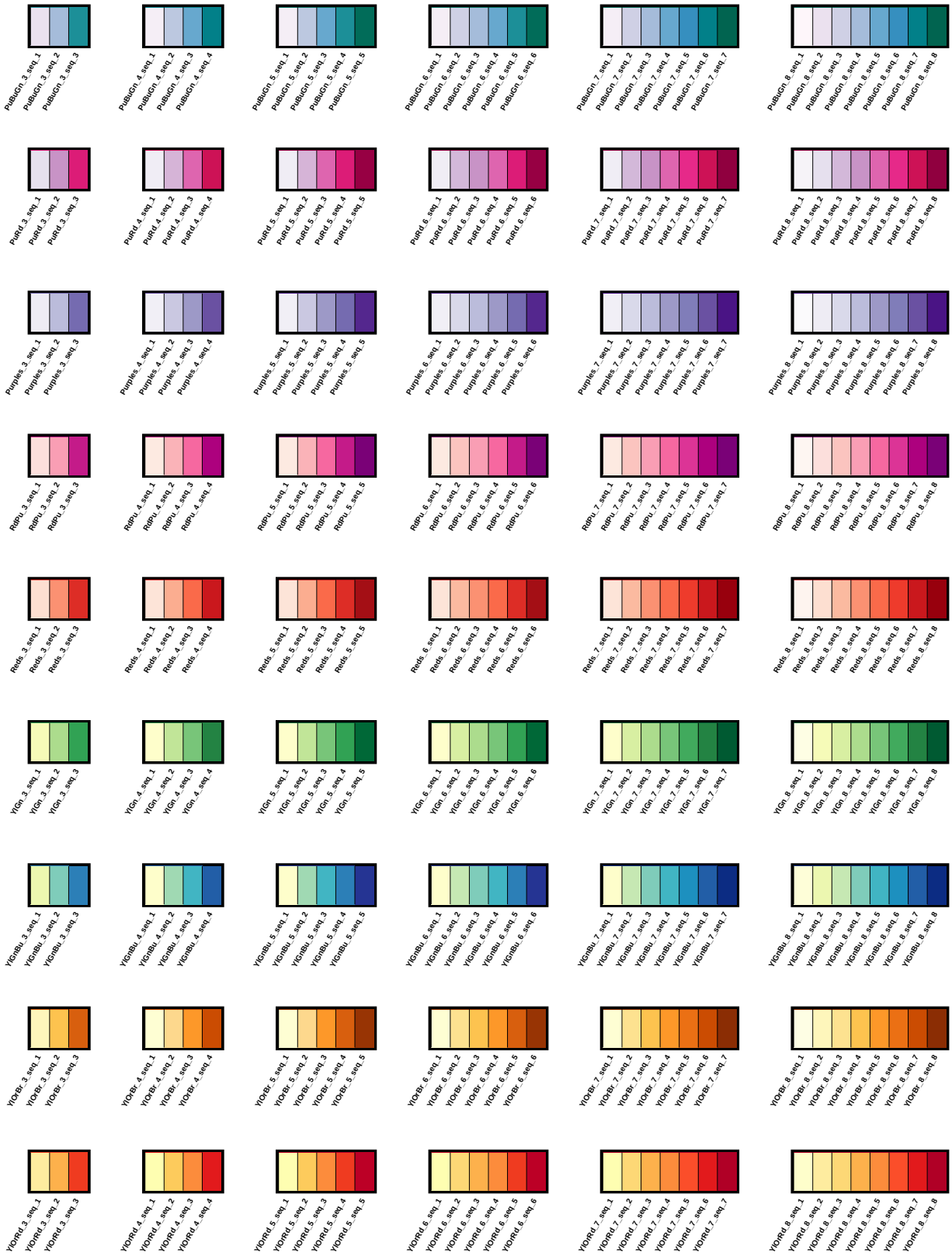

Figure 8: ColorBrewer sequential color schemes with 3-8 colors. Each row of graphs represents a different family of color schemes, and each column represents a different number of colors. The labels below the graphs represent the color names recognized by *distruct*.

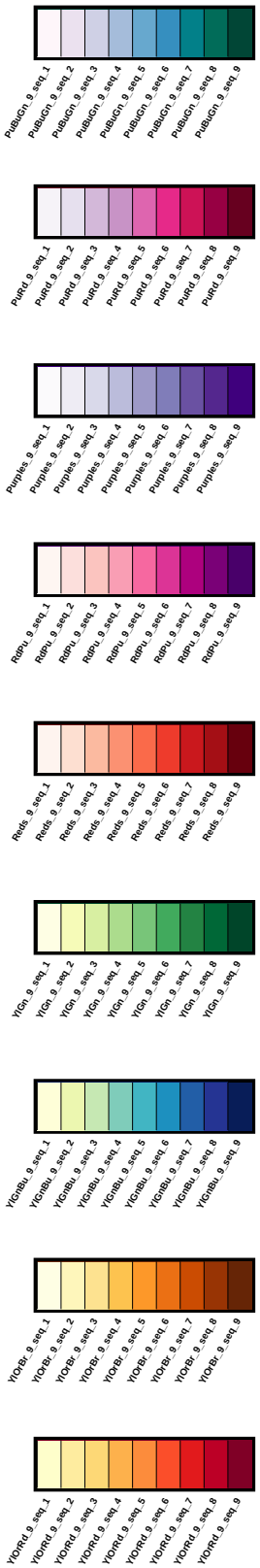

Figure 9: ColorBrewer sequential color schemes with 9 colors (continued from the schemes with 3-8 colors in Figure 8). Each row of graphs represents a different family of color schemes, and each column represents a different number of colors. The labels below the graphs represent the color names recognized by *distruct*.

2.6 Formatting errors

As in *structure*, the program attempts to verify that the input files are correctly formatted and reports errors that it finds. The program exits if a serious "Error" is discovered. In some cases, problems are not serious, a "Note" or "Warning" is issued, and the program will execute. There may be circumstances in which the desired usage of the program produces notes or warnings.

Editing the files to be used in *distruct* may introduce hidden formatting characters at the ends of lines or at the ends of files. Similarly to *structure*, this is one of the most frequent causes of errors when input files otherwise seem to be correctly formatted. In UNIX systems, the dos2unix function can remove many of these characters.

3 Usage options

It is best to place *distruct* and all files needed for its execution in the same directory. Every time *distruct* is run, it reads its instructions from the file *drawparams*. The format of this file is analogous to the *mainparams* and *extraparams* files of *structure*. Each parameter is printed in all-caps in the file, preceded by the word "#define" (parameters are also printed in all-caps throughout this document). The value is taken from the field that immediately follows the parameter name (for example "#define FONTHEIGHT 10" sets the font height to 10).

Following each parameter definition is a comment (marked "//") that describes the parameter. The comment indicates what type of value is expected: "(str)", for string (used for names of input and output files); "(int)", for integer; "(d)", for double (a real number - integers included); and "(B)", for Boolean (1 for TRUE, 0 for FALSE).

The program is insensitive to the order of the parameters, so they can be rearranged. The values of all parameters used for a given run are printed at the end of the output file. Several of the parameters in *drawparams* can be changed from the command line (Section 3.3).

3.1 Data settings and main options

The main settings concern the input files, the amount of data, and the items to be printed.

INFILE_POPQ (string) Name of input file for population Q-matrix (maximum length of 50 characters or possibly less, depending on the operating system). This file is required.

INFILE_INDIVQ (string) Name of input file for individual Q-matrix (maximum length of 50 characters or possibly less, depending on the operating system). This file is required if PRINT_INDIVS=1.

INFILE_LABEL_ATOP (string) Name of input file for list of labels to be printed atop the figure (maximum length of 50 characters or possibly less, depending on the operating system). This file is optional, but is expected if PRINT_LABEL_ATOP=1.

INFILE_LABEL_BELOW (string) Name of input file for list of labels to be printed below the figure (maximum length of 50 characters or possibly less, depending on the operating system). This file is optional, but is expected if PRINT_LABEL_BELOW=1.

INFILE_CLUST_PERM (string) Name of input file for permutation of clusters and color specifications (maximum length of 50 characters or possibly less, depending on the operating system). This file is optional.

OUTFILE (string) Name of output file (maximum length of 50 characters or possibly less, depending on the operating system). An existing file with this name will be overwritten.

K (int) Number of clusters. Maximum of 60.

NUMPOPS (int) Number of populations in population Q-matrix. Maximum of 5000.

NUMINDS (int) Number of individuals in individual Q-matrix. Maximum of 5000.

PRINT_INDIVS (Boolean) 1 to plot the individual Q-matrix, 0 to plot the population Q-matrix.

PRINT_LABEL_ATOP (Boolean) 1 to print labels atop the figure. If INFILE_LABEL_ATOP cannot be found, the default is to print the population codes. If this is set to 0, the labels will be printed as comments in the PostScript file.

PRINT_LABEL_BELOW (Boolean) 1 to print labels below the figure. If INFILE_LABEL_BELOW cannot be found, the default is to print the population codes. If this is set to 0, the labels will be printed as comments in the PostScript file.

PRINT_SEP (Boolean) 1 to print black vertical lines to separate the results for different populations.

3.2 Figure appearance and additional options

Most of the following options control the figure appearance. The coordinate plane treats the lower-left corner as the origin; 1 unit is 1/72 inches.

FONTHEIGHT (double) The size of the font used for printing labels atop and below the figure. The font is Helvetica-Bold.

DIST_ABOVE (double) The distance above the figure at which to put beginnings of labels (labels are left-justified).

DIST_BELOW (double) The distance below the figure at which to put ends of labels (labels are right-justified).

BOXHEIGHT (double) The vertical height of the figure.

INDIVWIDTH (double) The width allotted to the results for each individual.

ORIENTATION (int) The orientation for the figure on the page. The default choice is 0, "horizontal orientation," in which the figure is printed from left to right. In discussing items "atop" and "below" the figure, this document assumes the default choice. If another orientation is desired, such as if the figure is too big to be displayed in a "portrait" mode,

it is likely to be easiest to use the default orientation and to apply isometries to the figure in another application. There may be some exceptions: for example, if a PostScript figure converted to PDF and is then cut and pasted from a PDF file to a Microsoft PowerPoint presentation, it may be difficult in some PowerPoint versions to rotate the figure once in PowerPoint. In this case it might be desirable to set ORIENTATION to 3, "reverse vertical orientation," and rotate the figure in a PDF viewer before copying to PowerPoint. Other possibilities include 1 for "vertical orientation" and 2 for "reverse horizontal orientation." For orientation 0, a sensible choice for (XORIGIN, YORIGIN) is (72, 288); for 1, (360, 72); for 2, (540, 504); for 3, (288, 720).

XORIGIN (double) The x-coordinate of the lower left corner of the figure.

YORIGIN (double) The y-coordinate of the lower left corner of the figure.

XSCALE (double) Dilation factor for the x-direction. This factor is applied to all aspects of the figure but does not affect the origin.

YSCALE (double) Dilation factor for the y-direction. This factor is applied to all aspects of the figure but does not affect the origin.

ANGLE_LABEL_ATOP (double) Angle in degrees at which to print the labels atop the figure (in [0,180]).

ANGLE_LABEL_BELOW (double) Angle in degrees at which to print the labels below the figure (in [0,180]).

LINEWIDTH_RIM (double) The width of the line that comprises the border of the box.

LINEWIDTH_SEP (double) The width of the lines that separate results for different populations (relevant only if PRINT_SEP is 1). This parameter should be small compared to INDIVWIDTH.

LINEWIDTH_IND (double) The width of the lines used in printing the results. Results are filled boxes, so the value of this parameter should not have any impact.

GRAYSCALE (Boolean) Set to 1 to make a grayscale figure instead of a color figure. If this is 1, real numbers in [0,1] can be entered in INFILE_CLUST_PERM in place of the names of colors. The default scheme is to use $k/(K+1)$ for the kth cluster.

ECHO_DATA (Boolean) Set to 1 to print to the screen the first few and last few lines of the files INFILE_POPQ and (if appropriate) INFILE_INDIVQ.

REPRINT_DATA (Boolean) Set to 1 to print the population Q-matrix (and the individual Q-matrix, if applicable) as a comment in the PostScript file.

PRINT_INFILE_NAME (Boolean) Set to 1 to print the name of INFILE_POPQ above the figure. This option is meant for use only with ORIENTATION=0.

PRINT_COLOR_BREWER (Boolean) Set to 1 to print the ColorBrewer settings in the output file. This option is required if using colors from ColorBrewer.

3.3 Command-line arguments

Command-line flags enable the user to input certain parameters from the command line. The values entered with all flags except '-d' will overwrite the value in the *drawparams* file. The '-d' option enables use of a different parameter file in place of *drawparams*.

-d (drawparams) Read a different parameter input file instead of *drawparams*.

-K (K) Change the number of clusters.

-M (NUMPOPS) Change the number of predefined populations.

-N (NUMINDS) Change the number of individuals.

-p (input file) Read a different input file for the population Q-matrix.

-i (input file) Read a different input file for the individual Q-matrix.

-a (input file) Read a different input file for the labels to be printed atop the figure.

-b (input file) Read a different input file for the labels to be printed below the figure.

-c (input file) Read a different input file for the permutation of clusters and the colors.

-o (output file) Print results to a different output file.

The argument for a flag follows the flag and is separated from the flag by a space. The flags can be used in any order. For example, to change the number of individuals to 317 and the labels atop the figure to a file named *continents*, the appropriate command (in Unix) is:

```
./distruct -N 317 -a continents
```

If all parameters in *drawparams* are satisfactory, the program is called with `./distruct`. If the program lies in a different directory from the current one, it will search for *drawparams* and input files in the current directory. That is, `self/distruct` will search for input files in the present working directory rather than in the directory `self`. In Windows, the program is run most easily by typing `distruct` from a Dos prompt, in a directory that contains the program and the input files.

4 Examples

Consider the files in Tables 1-5 with the default settings in the file *drawparams*. When we run `./distruct`, we obtain Figure 10. Note the slight differences from the figure that was shown for the same data in Rosenberg *et al.* (2002). In Figure 10, the bottom-to-top order of the clusters was (5,4,1,2,3), while in Rosenberg *et al.* (2002) the order used was (1,2,3,4,5). The figure in Rosenberg *et al.* (2002) was smaller and did not print labels atop the figure.

Suppose that we wish to display only the population Q-matrix. We set PRINT_INDIVS=0. We also remove the separators between populations (PRINT_SEP=0). We then obtain Figure 11. Additional examples using other data sets are shown in Figures 12 and 13.

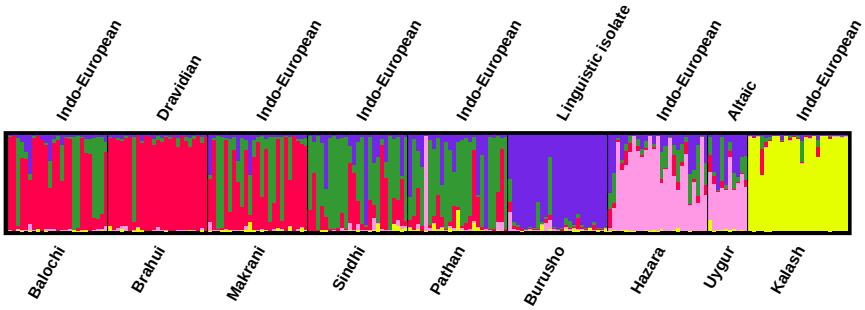

Figure 10: Graph of individual Q-matrix in Table 2 using information from Tables 1, 3, 4, 5.

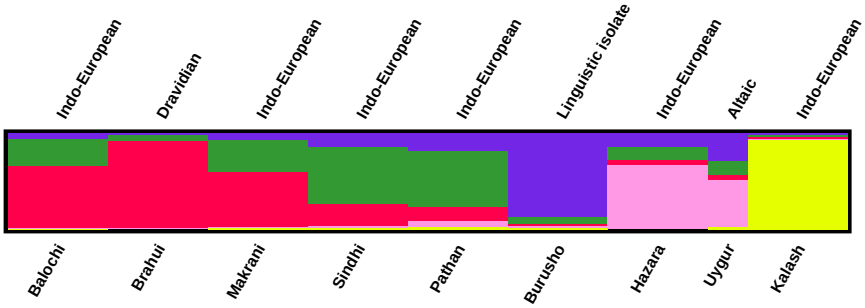

Figure 11: Graph of population Q-matrix in Table 1, using information from Tables 3, 4, 5.

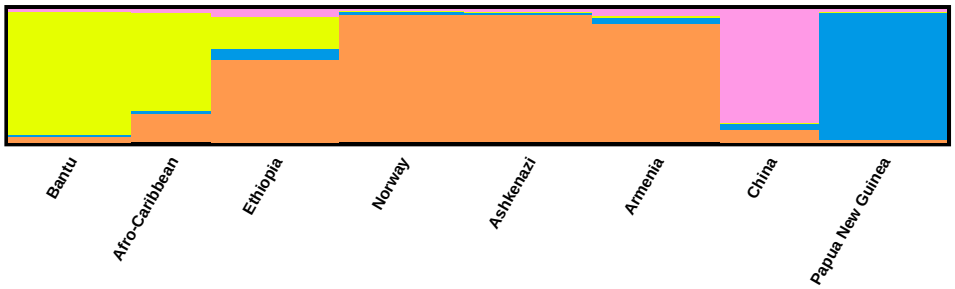

Figure 12: Graph of Table 2 from Wilson *et al.* (2001). The default permutation (1,2,3,4) was used with the default colors. For these data, $K = 4$, NUMPOPS=8, and NUMINDS=352. INDIVWIDTH was set to 1.

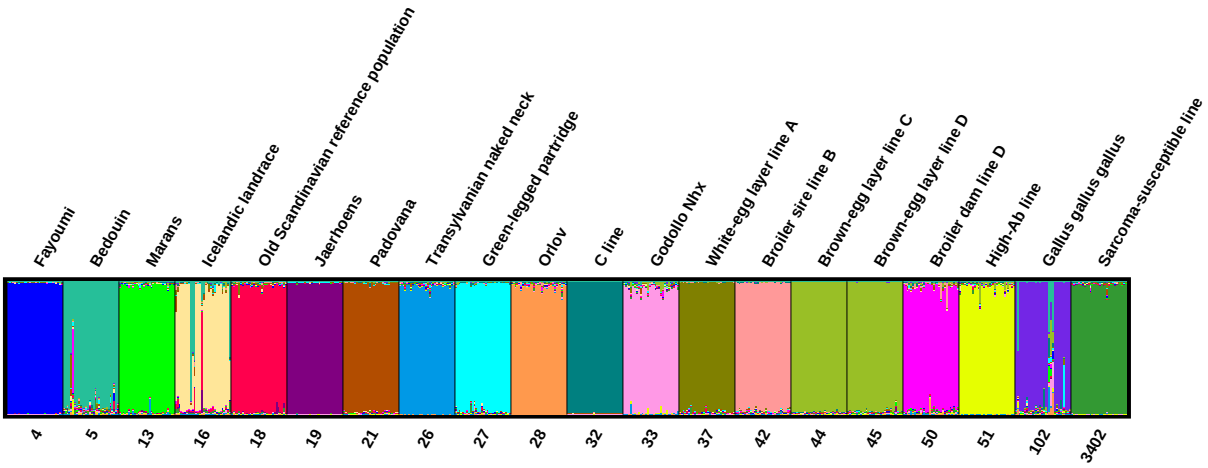

Figure 13: Graph of the *structure* run of highest estimated posterior probability among 100 runs shown in Table 2 of ROSENBERG *et al.* (2001). The default permutation was used with the default colors. Below the figure are breed codes and above the figure are breed names. $K = 19$, NUMPOPS=20, and NUMINDS=600. INDIVWIDTH was set to 0.7.

The *distruct* program has been designed to provide a display format for *structure* that is useful for any value of K. Especially for small K (such as $K = 2, 3$), results might be best displayed in various other forms not accommodated by *distruct*; some early examples from the literature include BEAUMONT *et al.* (2001), KOSKINEN *et al.* (2002), and PRITCHARD *et al.* (2000a,b). At the same time, *distruct* may be useful in situations other than for the display of population structure; one such example is provided in Figure 7 of CONRAD *et al.* (2006).

5 Frequently asked questions

Scenario 1: *In the Windows version of distruct, I double click on the icon to run the program. A window pops up and immediately disappears.*

Solution: The program opens a new window, runs, and closes the window upon completion. The output file will be created and will be placed in the appropriate directory. A more informative way to run *distruct* is from a Dos prompt. Go to the Start Menu, click on Run, and type `cmd`. Using `cd`, change directories to the directory where *distruct* is located, and then type `distruct`. Alternatively, double click on the `distruct.bat` MS-Dos batch file instead of the `distruct.exe` application. This command will open a window with a Dos prompt and will then run *distruct* in the window.

Scenario 2: *In the Windows version of distruct, the PostScript output file is not recognized. What software do I need to view the output?*

Solution: GhostView, `http://pages.cs.wisc.edu/~ghost`, is free software that can view PostScript files. Download both GSView and Ghostscript. Alternatively, programs such as Acrobat Distiller and Illustrator can read PostScript files.

Scenario 3: *When opening the PostScript output file with a PostScript viewer, the page is blank. However, no error messages were produced when running distruct, and the size of the output file is nonzero.*

Solution: Sometimes this problem arises from errors in the settings of the PostScript viewer that change the size of the page. Try running *distruct* with smaller values of XORIGIN and YORIGIN, or converting the PostScript to PDF.

References

Beaumont, M., E. M. Barratt, D. Gottelli, A. C. Kitchener, M. J. Daniels, J. K. Pritchard and M. W. Bruford, 2001 Genetic diversity and introgression in the Scottish wildcat. Mol. Ecol. **10:** 319–336.

Brewer, C. A., G. W. Hatchard and M. A. Harrower, 2003 ColorBrewer in print: a catalog of color schemes for maps. Cartogr. Geogr. Inform. Sci. **30:** 5–32.

Conrad, D. F., M. Jakobsson, G. Coop, X. Wen, J. D. Wall, N. A. Rosenberg and J. K. Pritchard, 2006 A worldwide survey of haplotype variation and linkage disequilibrium in the human genome. Nature Genet. **38:** 1251–1260.

Falush, D., M. Stephens and J. K. Pritchard, 2003 Inference of population structure using multilocus genotype data: Linked loci and correlated allele frequencies. Genetics **164:** 1567–1587.

Harrower, M. A., and C. A. Brewer, 2003 ColorBrewer.org: an online tool for selecting color schemes for maps. Cartogr. J. **40:** 27–37.

Koskinen, M. T., P. Sundell, J. Piironen and C. R. Primmer, 2002 Genetic assessment of spatiotemporal evolutionary relationships and stocking effects in grayling (*Thymallus thymallus*, Salmonidae). Ecol. Lett. **5:** 193–205.

Pritchard, J. K., M. Stephens and P. Donnelly, 2000a Inference of population structure using multilocus genotype data. Genetics **155:** 945–959.

Pritchard, J. K., M. Stephens, N. A. Rosenberg and P. Donnelly, 2000b Association mapping in structured populations. Am. J. Hum. Genet. **67:** 170–181.

Rosenberg, N. A., 2004 *Distruct*: a program for the graphical display of population structure. Mol. Ecol. Notes **4:** 137–138.

Rosenberg, N. A., T. Burke, K. Elo, M. W. Feldman, P. J. Freidlin, M. A. M. Groenen, J. Hillel, A. Mäki-Tanila, M. Tixier-Boichard, A. Vignal, K. Wimmers and S. Weigend, 2001 Empirical evaluation of genetic clustering methods using multilocus genotypes from 20 chicken breeds. Genetics **159:** 699–713.

Rosenberg, N. A., J. K. Pritchard, J. L. Weber, H. M. Cann, K. K. Kidd, L. A. Zhivotovsky and M. W. Feldman, 2002 Genetic structure of human populations. Science **298:** 2381–2385.

Wilson, J. F., M. E. Weale, A. C. Smith, F. Gratrix, B. Fletcher, M. G. Thomas, N. Bradman and D. B. Goldstein, 2001 Population genetic structure of variable drug response. Nature Genet. **29:** 265–269.