Vowel normalization and plotting with the phonR package*

Daniel R. McCloy

Department of Linguistics, University of Washington
drmccloy@uw.edu

1 Introduction

The phonR package is a collection of functions for the R language (R Development Core Team, 2012) intended to meet the needs of phoneticians and phonologists. Currently the package includes functions for vowel normalization and vowel plotting; other functions for tasks like generating standard consonant charts and manipulating distinctive feature vectors for phone inventories are under development. This document describes phonR's vowel plotting and normalization capabilities.

The vowel plotting capabilities in phonR were developed to enhance the flexibility and visual appeal of vowel plots, especially plots with multiple groups' data on a single graph. To that end, several options are available in phonR that are unavailable or difficult to achieve with other vowel plotting tools (the vowels package in R (Kendall & Thomas, 2009) and its associated website NORM (Thomas & Kendall, 2007) being a popular alternative). In particular, the phonR package implements confidence interval plotting based on bivariate normal density contours, allowing for ellipsoids in F2×F1 space whose major and minor axes are not restricted to the horizontal or vertical (compare to NORM's crosshairs-style plotting of standard deviations in the F1 and F2 dimensions). Additionally, phonR introduces support for automatic drawing of the vowel polygon as well as the option to leave certain vowels unconnected to the polygonal line (e.g., central vowels like [ə]). Finally, both vowel means and individual vowel tokens may be plotted with IPA glyphs or geometric symbols (or may be omitted), with smart defaults for various color, linestyle, and shape options.

The phonR package can be downloaded from the Comprehensive R Archive Network (http://cran.r-project.org/package=phonR). The source code is hosted at https://github.com/drammock/phonR, and is released under GPL-3 (http://www.gnu.org/licenses/gpl.html). The package's plotting functionality depends on two other R packages: mixtools (Benaglia et al., 2009) and Cairo (Urbanek & Horner, 2011). The Cairo R package in turn depends on the Cairo graphics library (Worth & Esfahbod, 2012); installation of the Cairo graphics library is described in Section 4. Plotting with IPA glyphs instead of geometric shapes may also require a typeface with Unicode-IPA symbols; the package defaults to the Charis-SIL font (SIL International, 2011), but future versions will be more flexible regarding font choice.

*Development of the phonR package was funded in part by the National Institutes of Health, grant # R01DC006014 to Pamela Souza. The author would like to thank August McGrath for her assistance in developing and debugging early versions of the `plotVowels` function.

Like all R packages, phonR can be installed from within R with the `install.packages` command. Alternatively, downloaded source code can be installed from a command line console with the command `R CMD INSTALL phonR -l /path/to/unzipped/source/of/phonR/`, and some R frontends also have GUIs for package installation. Regardless of installation method, the package is loaded with the R command `library(phonR)`.

2 Normalizing vowels with phonR

The phonR function `normalizeVowels` implements several of the most common normalization algorithms, which are listed in Table 1. Depending on the normalization method chosen, the function can operate on a single vector of values, or may require both F1 and F2 (Watt-Fabricius method) or all of f0, F1, F2, and F3 (Nearey-2 method). Additionally, the Watt-Fabricius, Nearey-1, Nearey-2 and Lobanov normalization methods can make use of an optional argument `grouping.factor`, which allows normalization to occur speaker-intrinsically (other normalization methods are automatically speaker-intrinsic by virtue of their formulae).

Normalization method	**Source**	**Formula**
Bark	Traunmüller (1990)	$\frac{26.81 \times F_n}{1960 + F_n} - 0.53$
Equivalent Rectangular Bandwidth (erb)	Glasberg & Moore (1990)	$21.4 \times \log_{10}(1 + F_n \times 0.00437)$
Mel	Stevens & Volkmann (1940)	$2595 \times \log_{10}(1 + \frac{F_n}{700})$
Log	n/a	$\log_{10}(F_n)$
Lobanov (z-transform, z-score)	Lobanov (1971)	$\frac{F_n - \mu(F_n)}{\sigma(F_n)}$
Watt-Fabricius (s-centroid)[1]	Watt & Fabricius (2002)	$\frac{F_n}{centroid}$
Nearey-1 (logmean)	Nearey (1977)	$\log_e(F_n) - \mu(\log_e(F_n))$
Nearey-2	Nearey (1977)	$\log_e(F_n) - \sum_{n=0}^{3} \mu(\log_e(F_n))$

Table 1: Normalization methods available via phonR's `normalizeVowels` function

[1] In the phonR package, the centroid is defined as the point $\langle \frac{\min(\overline{F2_{vowel}}) + \max(\overline{F2_{vowel}})}{2}, \frac{2 \times \min(\overline{F1_{vowel}}) + \max(\overline{F1_{vowel}})}{3} \rangle$. This varies slightly from the formula in Watt & Fabricius (2002), since the phonR implementation simply calculates which vowel has the highest mean F1 value and designates it as low corner of the triangle, rather than asking the user to expressly specify the *trap* or *start* vowel. Similarly, the phonR implementation simply calculates which vowel has the highest mean

3 Generating vowel plots with phonR

The comprehensive function call for vowel plotting is shown in Listing 1, and the resulting plot is shown in Figure 1. Of special note here are the arguments `match.axes`, `poly.order`, and `poly.include`. The first of these, `match.axes`, determines whether all graphs will have the same axis limits; possible values are `'absolute'` (all graphs have the same dimensions), `'relative'` (all graphs have the same scale and aspect ratio, but their absolute limits may vary), or `'none'` (graphs may vary in their scale, aspect ratio, and/or absolute limits). This argument applies only when a grouping factor is supplied and `single.plot=FALSE`, so it was ignored when drawing the plot in Figure 1, but the effect of `match.axes='absolute'` (the default value) can be seen in Figure 3.

```
plotVowels(data=myDataFrame, f1='F1_column_name', f2='F2_column_name',
  f3='F3_column_name', f0='f0_column_name', vowel='Vowel_column_name',
  grouping.factor='Gender_column_name', norm.method='mel', match.unit=
  TRUE, match.axes='absolute', points='shape', points.alpha=0.3,
  means='text', means.alpha=1, ignore.hidden=TRUE, ellipses=TRUE,
  ellipse.size=0.3173, polygon=TRUE, poly.order=c('i','e','a','o','u'),
  poly.include=NULL, single.plot=TRUE, axis.col='#666666FF', titles=
  'auto', grayscale=FALSE, vary.shapes=TRUE, vary.lines=TRUE, legend=
  TRUE, uniform.style=FALSE, aspect.ratio=NULL, plot.dims=c(6.5,6.5),
  plot.unit='in', output='pdf')
```

Listing 1: A comprehensive function call to phonR's `plotVowels` function

The `poly.order` argument specifies the order in which vowels should be connected if `polygon=TRUE`; the default value for this argument is the set {i, ɪ, e, ɛ, æ, a, ɑ, ɔ, o, ʊ, u, ʌ}, and it is only necessary to explicitly provide this argument if you prefer the polygon be drawn in a different order or if your data contains vowels not in the default set. The `poly.include` argument takes an integer indicating how many of the vowels present in the data should be connected by the polygon segments, allowing certain vowels to be omitted (for example, one might choose to connect only the short vowel in a series of long/short vowel pairs, in which case the long vowels should be placed at the end of the `poly.order` vector, and `poly.include` should be the index of the last vowel to be connected to the polygonal line). Also shown in Figure 1 is the automatic axis labeling with regard to the chosen normalization method, variation in plot symbol shape and linestyle, and the effect of the transparency arguments (`points.alpha` and `means.alpha`). The line styles cycle through the default linestyles in R, but the plot symbols are restricted to a subset of the symbols available through `pch`, namely the set ○ □ △ ◇ ● ■ ▲ ◆ + × ▽.

A more typical function call includes fewer arguments, as in Listing 2; the output of this function call is shown in Figure 2. This example illustrates a few of the intelligent defaults of phonR's `plotVowels` function. For example, graphical options like line style or shape are automatically held constant if the plot is in color, or if each subject/group is plotted on a separate graph. Listing 2 also illustrates that the `data` argument may be omitted if the other required inputs are supplied as separate vectors (note the lack of quotation marks on the `f1`, `f2`, `vowel`, and `grouping.factor` arguments).

F2 value and uses that to calculate the upper left corner, rather than expressly looking for the mean of the "point-vowel" /i/. The upper right corner is, as in the original method, derived from the other two.

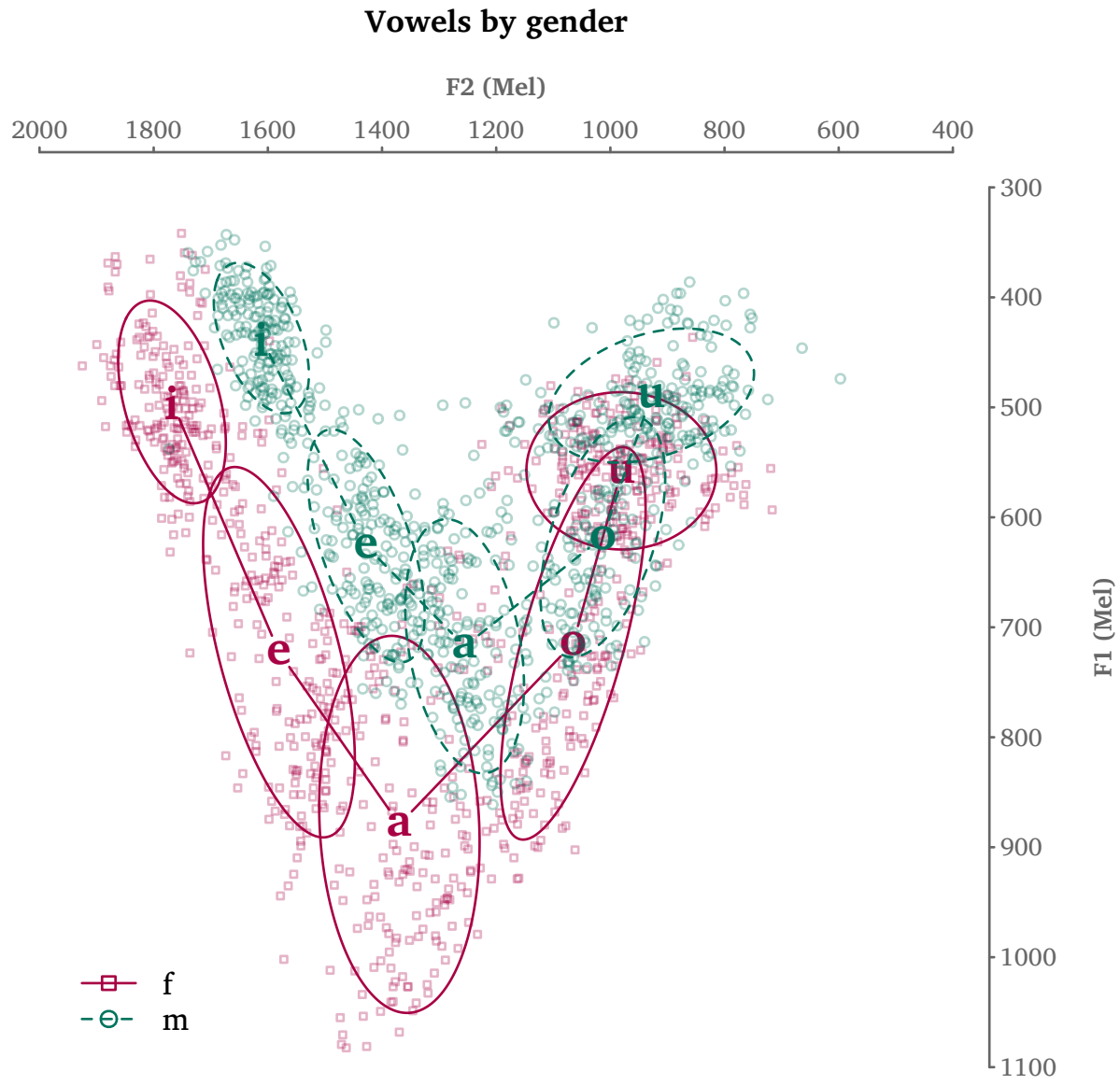

Figure 1: Vowel plot based on the maximal function call in Listing 1

For colored plots, `plotVowels` generates colors of equal chroma and luminance, with hues equally spaced around the color circle in HCL space (Zeileis et al., 2009). A custom title is also included here; the `titles` argument accepts either a string or vector of strings, and will recycle or discard strings as needed (with a warning to the user) if there is a mismatch between the number of graphs to be generated and the number of strings provided. Finally, Figure 2 illustrates the `match.unit=FALSE` argument, which transforms the data points using the specified normalization method (in this case, the Bark transformation) and plots them in the transformed space, but labels the axes with values equispaced in Hz, highlighting the non-linearity of the transform (note the unequal spacing of tickmarks that are numerically equidistant).

```
plotVowels(data=myDataFrame, f1='F1_column_name', f2='F2_column_name',
  vowel='Vowel_column_name', grouping.factor='subj', norm.method=
  'bark', match.unit=FALSE, points='none', points.alpha=0.6, means=
  'text', means.alpha=1, ignore.hidden=TRUE, ellipses=FALSE, polygon=
  TRUE, single.plot=TRUE, grayscale=FALSE, legend=TRUE, titles=
  'Indonesian vowel spaces (means and polygons)', aspect.ratio=1.5,
  plot.dims=c(6.5,6.5), plot.unit='in', output='pdf')
```

Listing 2: A typical function call to phonR's `plotVowels` function

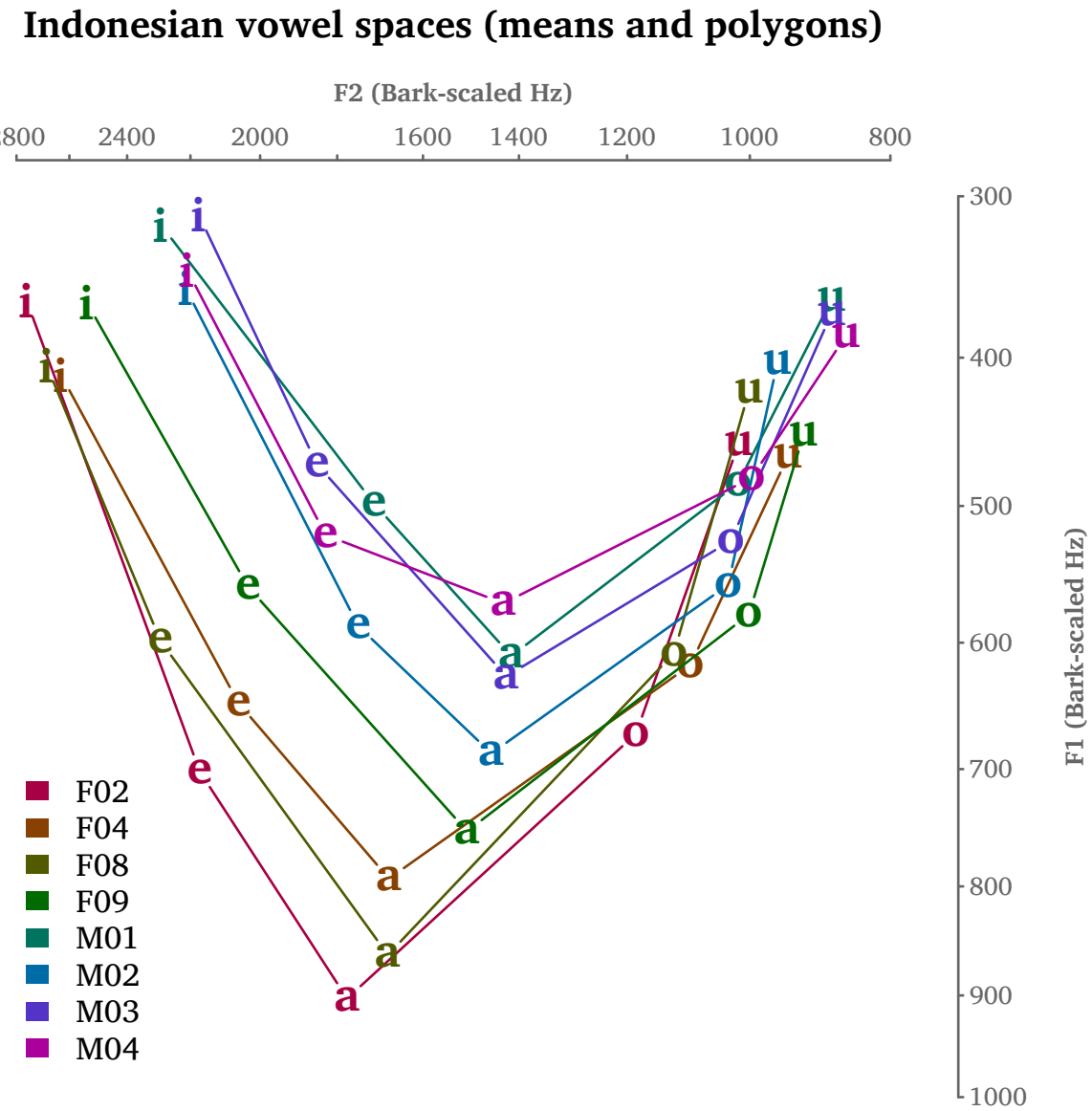

Figure 2: Vowel plot based on the typical function call in Listing 2

A minimal function call may include only `f1`, `f2`, and `vowel` arguments. The four main plot elements (points, means, ellipses, and polygons) each have default values and thus may be omitted if the defaults are suitable; omitting `grouping.factor` treats all points as belonging to the same subject or group.

```
plotVowels(f1=f1Vector, f2=f2Vector, vowel=vowelVector,
  grouping.factor=groupingVector, points='none', means='text',
  ellipses=TRUE, polygon=TRUE, single.plot=FALSE, output='screen')
```

Listing 3: A minimal function call to phonR's `plotVowels` function

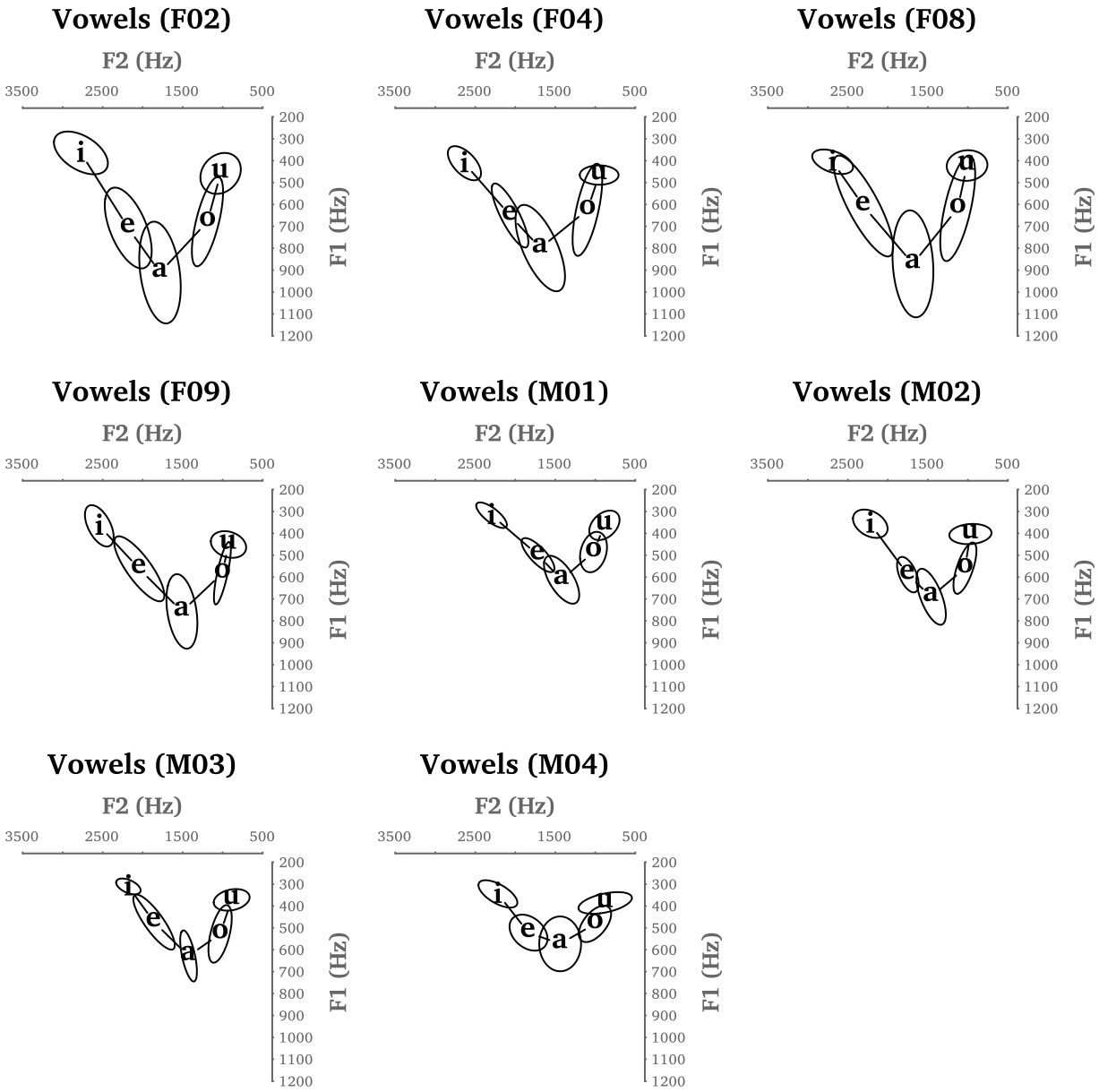

Figure 3: Vowel plot based on the minimal function call in Listing 3

Listing 3 and its corresponding plot in Figure 3 show another unique feature of the `plotVowels` function, which is that the argument `single.plot=FALSE` behaves differently depending on output method. If the output method is `'pdf'` or `'jpg'`, then a separate PDF or JPG file will be created for each group. In contrast, if the output method is `'screen'` then the graphs for each group are arranged in a grid so they can be easily inspected side-by-side. It also illustrates the automatic titling functionality, which labels each graph with the grouping factor value (in this case, subject number, though this could just as easily be task type, dialect, ethnicity, etc). Finally, as mentioned above, Figure 3 illustrates the effect of the argument `match.axes='absolute'`, which forces all graphs to the same dimensions and axis limits, allowing visual comparisons between graphs of the relative size and position of different speakers' vowel spaces.

4 Appendix: Installing the Cairo graphics library

Most versions of Linux distribute the Cairo graphics library through standard package management systems, and in most cases Cairo will already be installed. On Ubuntu-like systems you should check for the presence of packages "libcairo2" and "libcairo2-dev". On OS-X, installing Cairo is trivially accomplished through either Fink or MacPorts (though I'm told the installation of MacPorts itself is sometimes finicky). See http://www.cairographics.org/download/ for instructions. Getting the Cairo graphics library to work on Windows is tricky, and the guidance on the Cairo Graphics website was incomplete at time of writing. A method that is known to work is to download the run-time files from the GTK website and manually move them to the proper folder within your R system install path. Specifically, go to http://www.gtk.org/download/ and click on "Windows" (either 32-bit or 64-bit depending on your computer). Then scroll down to "Required third party dependencies" and download the run-time files for `zlib`, `cairo`, `libpng`, `fontconfig`, `freetype`, and `expat`. Uncompress them and gather up all the DLL files from the "bin" folders that were in each zipped package. Then move/copy those DLL files into the `/bin/x64/` folder[2] within your R program directory (usually this is C:/Program Files/R/R-2.14.0 or similar). Also move or copy "fonts.conf" from the `fontconfig` zip file, and put that into the `/etc/x64/fonts/` folder[3] within the R program directory. Of course, you will still need to install the Cairo R package from within R to make use of these libraries for plotting.

[2] or on 32-bit systems, the `/bin/i386/` folder.

[3] or on 32-bit systems, the `/etc/i386/fonts/` folder.

References

Benaglia, T., Chauveau, D., Hunter, D. R., & Young, D. (2009). mixtools: An R package for analyzing finite mixture models. *Journal of Statistical Software*, *32*(6), 1–29.

Glasberg, B. R., & Moore, B. C. J. (1990). Derivation of auditory filter shapes from notched-noise data. *Hearing Research*, *47*(1-2), 103–138.

Kendall, T., & Thomas, E. R. (2009). Vowels: Vowel manipulation, normalization, and plotting in R. http://cran.r-project.org/web/packages/vowels/index.html.

Lobanov, B. M. (1971). Classification of russian vowels spoken by different speakers. *The Journal of the Acoustical Society of America*, *49*(2), 606–608.

Nearey, T. M. (1977). *Phonetic feature systems for vowels*. Doctoral dissertation, University of Alberta.

R Development Core Team (2012). R: A language and environment for statistical computing. http://www.R-project.org/.

SIL International (2011). Charis SIL. http://scripts.sil.org/CharisSILfont.

Stevens, S. S., & Volkmann, J. (1940). The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, *53*(3), pp. 329–353.

Thomas, E. R., & Kendall, T. (2007). NORM: The vowel normalization and plotting suite. http://ncslaap.lib.ncsu.edu/tools/norm/index.php.

Traunmüller, H. (1990). Analytical expressions for the tonotopic sensory scale. *The Journal of the Acoustical Society of America*, *88*(1), 97–100.

Urbanek, S., & Horner, J. (2011). Cairo: R graphics device using cairo graphics library for creating high-quality bitmap (PNG, JPEG, TIFF), vector (PDF, SVG, PostScript) and display (X11 and win32) output. http://CRAN.R-project.org/package=Cairo.

Watt, D., & Fabricius, A. H. (2002). Evaluation of a technique for improving the mapping of multiple speakers' vowel spaces in the F1 ~ F2 plane. *Leeds Working Papers in Linguistics and Phonetics*, *9*, 159–173.

Worth, C., & Esfahbod, B. (2012). Cairo. http://www.cairographics.org/.

Zeileis, A., Hornik, K., & Murrell, P. (2009). Escaping RGBland: selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, *53*(9), 3259–3270.