

Your Graphs and Tables at Their Best on the Web with ODS

LeRoy Bessler, Bessler Consulting & Research, Milwaukee, USA, bessler@execpc.com

Francesca Pierri, Università di Perugia - C.A.S.I., Perugia, Italy, frc@unipg.it

Abstract

You can create and deploy informative and influencing graphs and tables for the web, and link them to each other, with SAS/GRAPH®, SAS®, and the Output Delivery System (ODS), even without knowledge of HTML.

There are a variety of web deployment and linking options and tools, for which trade-offs between advantages and disadvantages need to be understood. And there are design concerns unique to presentation on the web. This paper covers these matters, and the presentation will show working examples.

You will learn: how to link and/or provide drill-down for trend lines, bar charts, pie charts, maps, text slides, and tables via clickable points, areas, legends, labels, and navigational hot spots; how to partition a web page to display in a single frame both a table of contents and any one of the graphs or tables listed therein; how to use the SAS/GRAPH WEBFRAME device driver; how to provide “flyover” pop-up labels as an alternative to permanent annotation; and how to use animation.

You will also learn how to use colors, fonts, PROC TEMPLATE and ODS styles, and the options and features of SAS/GRAPH, to create elegant and effective web pages.

This paper can help you to deliver communication-effective design, and to deploy graphs and tables on the web in a manner best suited to your needs and situation. It assumes at least novice knowledge of SAS/GRAPH, uses SAS Release 8.2, and is not *intended* to be specific to an operating system (any system dependencies or differences known to the authors will be discussed).

Introduction

The aim of this paper is to show you as much as possible about important SAS software capabilities for publishing graphs, tables, and slides on the Web.

Design-guided techniques have been used to override SAS defaults, to show drillable and linked graphs and tables in a web-page package in the most communication-effective way. Whether it is in the main body of the paper, or in the comments of the program code, or in the code itself, there are the best practices for dealing with titles, footnotes, legends, symbols, axes, tick marks, labels, colors, fonts, dimension, backgrounds, drill-down, cross-links, navigational hot spots, and the technical fine points of using ODS and SAS/GRAPH.

The objective is the creation of packages of web pages that are:

- (a) likely to look as intended—as seen during development by the original creator—with any other viewer’s web browser;
- (b) easily and quickly navigated; and
- (c) easy to read and interpret.

Strengths and limitations are identified for these facilities:

- WEBFRAME driver
- GIF driver
- GIFANIM driver
- Output Delivery System

Suggestions for their most effective use are provided in the following section.

Note: This paper, if in PDF format, is compatible with Acrobat Reader 5.0. Earlier versions will not display the illustrations well.

Design and Implementation

The programs shown in this paper are developed using Release 8.2 of SAS software on the PC. The particular operating system is Windows 98. No platform dependencies were intended.

The linked and drillable graphs, tables, and slides created can be viewed with either Internet Explorer 5.0 or Netscape Navigator 4.0 on a monitor with a diagonal as small as 14 inches (35.6 centimeters) and a resolution as low as 800 X 600. The commonest resolution setting for people browsing the web is 800 X 600.

The graph image has been sized to leave enough space for the browser's frame, and for any links that are required outside the graphic area, without causing a requirement for scrolling to view the entire image.

Only the GIF, GIFANIM, and WEBFRAME drivers are used to create graphs. Use of Java and ActiveX is avoided in order to deliver web pages that can be easily and readily browsed without requiring any extra features or facilities that might not be available for the viewer's web browser. Furthermore, well-designed graphs for data presentation should require no viewer/recipient modification for successful interpretation. Delivery of complex, multi-dimensional data, which may require the recipient to attempt to look at it different ways, is not the commonest data presentation need.

Browser-Safe Colors have been used. There are only 216 browser-safe colors. Despite its huge range of possible color choices, SAS/GRAPH cannot work with a palette greater than 255 colors at any one time. It is not an unreasonable limitation of design/construction freedom to commit to using only colors that any browser can be guaranteed to display as you intended.

Browser-Safe colors are RGB colors (mixtures of red, green, and blue) with SAS names of the form CXRRGGBB, where RR, GG, and BB are

restricted to the six possible hexadecimal values 00, 33, 66, 99, CC, and FF. This palette yields $6 \times 6 \times 6 = 216$ colors. All web browsers are guaranteed to be able to display these colors. Modern monitors and video display cards are rarely set to display fewer than 256 colors. It is a fact that many, if not most, people browsing the web have their displays set to view a maximum of 256 colors, regardless of the higher capabilities of their monitors and video cards. If non-browser-safe colors are used, such web page viewers will not necessarily see the same colors that the web page creator used and saw.

Combinations of background and foreground colors that maximize readability are used as much as possible. Avoided are those combinations that severely impair readability, such as black text on a dark background, or yellow text or lines on a white background.

TrueType fonts have been used. On a Unix machine, we would use the (quasi-) equivalent Unix system fonts. Both these choices are industry standards, which will be well rendered and familiar to a web page audience—unlike SAS/GRAPH-specific “software fonts”. Moreover, software fonts require extra processing for creation, and presumably yield larger web page files to transmit and display.

Whenever possible, the TrueType fonts used have been those developed by Matthew Carter for Microsoft. He designed his Georgia (serif) and Verdana (sans serif) fonts expressly for readability on screen and on the web.

The three-dimensional feature is not used to render images that do not show the relationship between three variables. It is a fact that, despite their popularity, 3D pie charts are deceptive, 3D bar charts can be more difficult to interpret, and 3D maps sometimes have the response for one geographic area hiding the response for another.

The presentation of the web pages is content-focused and message-focused. For instance, the web page is limited to one graph or table, so that

the viewer need not scroll up and down to find the image of interest. By default, the more common BY-processing, done for related images, usually spills all of them out into a web page that requires scrolling.

Multi-image scrollable web pages have two problems. First, the individual images are unable to capture focused attention of the viewer, except on a very small screen. Second, there is a situation that can cause viewer confusion. Suppose the viewer has been routed to one of the images via a hyperlink. If that image is at the bottom of the scrollable page, but the vertical space on the screen is taller than one image (e.g., it is large enough to display two successive images), then the viewer will naturally look at the image at the top of the page and think that something is wrong.

For presentation of shares of the whole when there would be too many pie chart slices to be readable, a customized ranked horizontal bar chart is provided, to show shares of the whole, not just the usual absolute response measures.

The web pages produced by the programs in this paper deliver images that are as sparse as possible, but which deliver all the required information. This focuses viewer attention on the meaning and message in the data, and may marginally decrease web page transmit and display times. The defaults for bar charts and trend plots are rooted in the hand-drawn, graph-paper- and pen-based tradition of axis lines, tick marks, and tick mark value labeling. However, the design and programs in this paper deliver web pages and images that tell the viewer only what, and everything that, she/he really needs to know, either statically and permanently, or dynamically upon request.

For trend lines, there are permanent labels displayed for the start and end values, but the intermediate values are available via web-enabled “flyover” pop-up labels, which are transient. Alternatively, all or selective intermediate points could have been

permanently annotated with the %TRNDANNO macro recently developed by one of the authors. Although SAS/GRAPH permits the use of area fill below plot lines, such fill adds no communication value, and is neither used here, nor recommended.

The examples include a few elegant customized uses of the SAS/GRAPH mapping capabilities and mapping support data sets.

Use of GOPTIONS NOGTITLE and/or NOGFOOTNOTE has been avoided. With those overrides, it is possible to specify HTML links in the SAS/GRAPH TITLE and FOOTNOTE statements, but have their display diverted to the “ODS/HTML area” outside the graph area. However, the vertical space that the title(s) and/or footnote(s) would require if displayed in the graph area is reserved, even though the links are displayed outside the graph area. The result is that the graph is artificially restricted. The empty white space may look strange, and the restriction can make it unnecessarily difficult to read the graph, especially if an annotated map.

Fortunately, circumvention is possible. You can execute an otherwise unnecessary PROC PRINT after the SAS/GRAPH PROC, but within the same HTML body file. The desired links are supplied via a TITLE statement(s) for the PROC PRINT. The data set that it “prints” contains one observation with one variable which is a blank and which has its label set to ‘00’X (which is undisplayable). Further, the custom ODS style built for this PROC PRINT is defined to display everything in white, which is invisible on the white background that is used. The design assures that the links in the title fit in the web browser window. If the unused invisible label and data do not fit, it does not matter.

A macro-language solution has been chosen to handle groups of graphs that differ only by the value of a classification variable instead of using ODS HTML ANCHOR= and BY processing. This solution, at least as coded, automatically assures that each graph or table is

a separate web page, the design benefits of which we have explained above.

The macro-based solution may be somewhat difficult for the novice to initially understand, but it is a powerful and reliable tool. Once its use is mastered, it eliminates error-vulnerable situation-specific hand coding. And there is no need to try to figure out how the index is being incremented from the anchor value for each step through the class (actually, “BY”) variable.

The programs (that use ODS) supply a custom frame title in the web browser banner line at the top of the screen.

For the real tables output by PROC PRINT (which could be any other PROC-generated tables as well), the custom ODS style built with PROC TEMPLATE makes the background for all the data cells and row and column labels white. Also, it makes invisible all the grid lines that ODS “adds” to the table if you use styles.default. There may be some situation in which these extras add communication value. If so, PROC TEMPLATE can be used to build a style for the backgrounds and/or grid elements to suit your design preferences.

In The CrossLink Example, by default, there would be a horizontal rule between the graph output and the PROC PRINT that is used to provide the non-drilldown links to other graphs or tables. The unneeded pseudo-pagebreak has been suppressed within the body file, using the custom style built with PROC TEMPLATE.

There are several customizations specifically for the Table of Contents example.

A custom title has been defined to replace the default “Table of Contents”, and the width available to the Table of Contents has been adjusted to prevent line breaks in the entries. For the Table of Contents entries, the PROCLABEL line has been blanked out, since it would be a no-added-value extra, and the not-useful link label text usually supplied by ODS

(such as “plot of y*x” or “print of lib.data”) has been replaced. The bullets that by default are to the left of the links, and to the left of the blanked-out PROC labels, have been suppressed. When the PROCLABEL line is blank, it still occupies useless white space, which causes a requirement for scrolling if the Table of Contents is long. That white space has been eliminated.

Three Ways to Organize and Link Graphs and Tables

Following sections present the SAS/GRAPH WEBFRAME driver, a Table of Contents on the left of your web page built with ODS and the GIF device driver, and the authors’ custom alternative, called “The CrossLink Example”—showing three ways in which you can use different SAS System capabilities to produce a communication-effective web application.

These methods are discussed starting with the simplest. Many **illustrations and much code** are presented **at the end of the paper**.

WEBFRAME Driver (Appendix 1)

This method is the easiest to implement, but applies only to graphs. The web frame provides a column of thumbnail images with clickable 8-character labels/links at the left side of the page. The serious limitation of this deployment is that it does not allow drill-down links or any other hyperlinks except from the thumbnail index column. Also, on a small screen, the thumbnails do occupy space, which may impair the viewability of your real images by requiring you to make them small enough to fit in the restricted real viewing area.

HTML Table of Contents (Appendix 2)

This is easy to implement. The program shows how to customize the index column. Like the case of the WEBFRAME solution described above, this deployment does take up space on

web browser screen. However, it does have the benefit of allowing the use of drill-down links.

The CrossLink Example (Appendix 3)

This is the most powerful solution, and the one that makes best use of available screen space.

For this solution, navigation and drill-down through the graphs and tables are absolutely customizable to suit your needs and preferences. The authors call it “The CrossLink Example” because it does not rely on a perhaps more familiar inverted tree network of images. Instead, with the drilldown options, the crosslinks, and the flyover text, one can explore one’s way through the information by a diversity of routes. A sufficiently curious or information-needful user can easily find all the available information, or at least any of the information that is of interest to her/him. The links could have been more descriptive, but have been designed for a small screen

This implementation uses no Table of Contents and no thumbnails. There is no loss of screen space at the side of the page. Of course, the non-drill-down links require some vertical space, but not very much. If vertical space is scarce, and the image is a graph, and you are very ambitious, it is possible to produce clickable hot spots at any open white space on the graph image, by use of ANNOTATE. Actually, our annotation of clickable city markers on maps is *somewhat* similar to what you would have to do.

Conclusions

After exercising, exploring, and comparing the capabilities of the SAS/GRAPH WEBFRAME driver, the customized ODS-supported Table of Contents implementation, and The CrossLink Example designed and developed by the authors, it is clear that the most flexible and powerful way to web-deploy and web-link data and image in tables and graphs that inform and influence is with the CrossLink implementation.

In any case, all of the illustrations used for this project employ a design that is intended to optimally use the software to communicate, not to decorate. When there eventually is increased presentation of tables and graphs via wireless communication and handheld devices, the minimal, but maximally informative and web-friendly, design presented here will be suitable for the small screen.

For now, this paper has compared tools currently available to SAS users, and has demonstrated what the authors suggest are the best practices for use of those tools today. However, the standards and features for web browsers, and the software capabilities of ODS and the SAS System, will evolve. The authors are continuing to explore and work.

Immediate further technical initiatives will be:

- (a) to see whether the ANCHOR= feature and traditional BY processing can be effectively used within the authors’ design framework;
- (b) to find a way to successfully incorporate high-quality scanned images with graphs;
- (c) to develop simple macros that can relieve the coding redundancy that occurs when assembling a package of web pages; and
- (d) to identify the simplest and most intuitive, but still adequate, way to use the complex PROC TEMPLATE facility to build the requisite styles.

Recommended software enhancements are:

- (a) support for the LINK= parameter (already available when using non-graphic PROCs) in the SAS/GRAPH TITLE, FOOTNOTE, and NOTE statements; and
- (b) increased white space between the link labels that are produced with LINK=, to improve the separability/readability of the link labels.

Acknowledgment

This really belongs at the beginning, not at the end. Thanks to the many people at SAS Institute Inc. who provide an excellent standard of competent, responsive, customer-oriented Technical Support.

Notices

SAS and SAS/GRAPH are trademarks or registered trademarks in the USA and other countries.
® denotes USA registration.

The Authors

Dr. LeRoy Bessler is a consultant, with over twenty years of experience with SAS software and other information technology, in a wide variety of applications and industries. He has a special interest in macro-based Software-Intelligent application development, to deliver solutions that are reliable, reusable, maintainable, and extendable. LeRoy has presented, in England, Ireland, and the USA, on topics in application development, graphic design for effective communication, data visualization, visual communication, color, and InfoGeographics.

LeRoy Bessler, PhD
Bessler Consulting & Research
PO Box 96
Milwaukee, WI 53201-0096, USA
1-414-351-6748
bessler@execpc.com

Dr. Francesca Pierri coordinates the Scientific Applications and Educational Assistance Branch at the University of Perugia. She works closely with university teachers and researchers on statistical data processing. Francesca is in charge of SAS software, and provides advice, technical support, and training for the SAS System. She has presented, in France, Ireland, Italy, and the USA, on topics in application development, graphic design, data visualization, and software performances. Francesca also is a consultant for Regione Umbria for the development of its Sanitary Information System. For that project she used SAS graphic visualization tools and developed an EIS application.

Francesca Pierri, PhD
Università di Perugia, C.A.S.I.
06125 Perugia, Italy
39-075-5853794
frc@unipg.it

Bibliography

Published in *Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference* by SAS Institute Inc. (Cary, N.C., USA, 2001):
“Your Graphs on the Web with SAS/GRAPH Version 8” by Francesca Pierri, and “The Power to Show: Using Information and Image to Reveal and Persuade” by LeRoy Bessler.

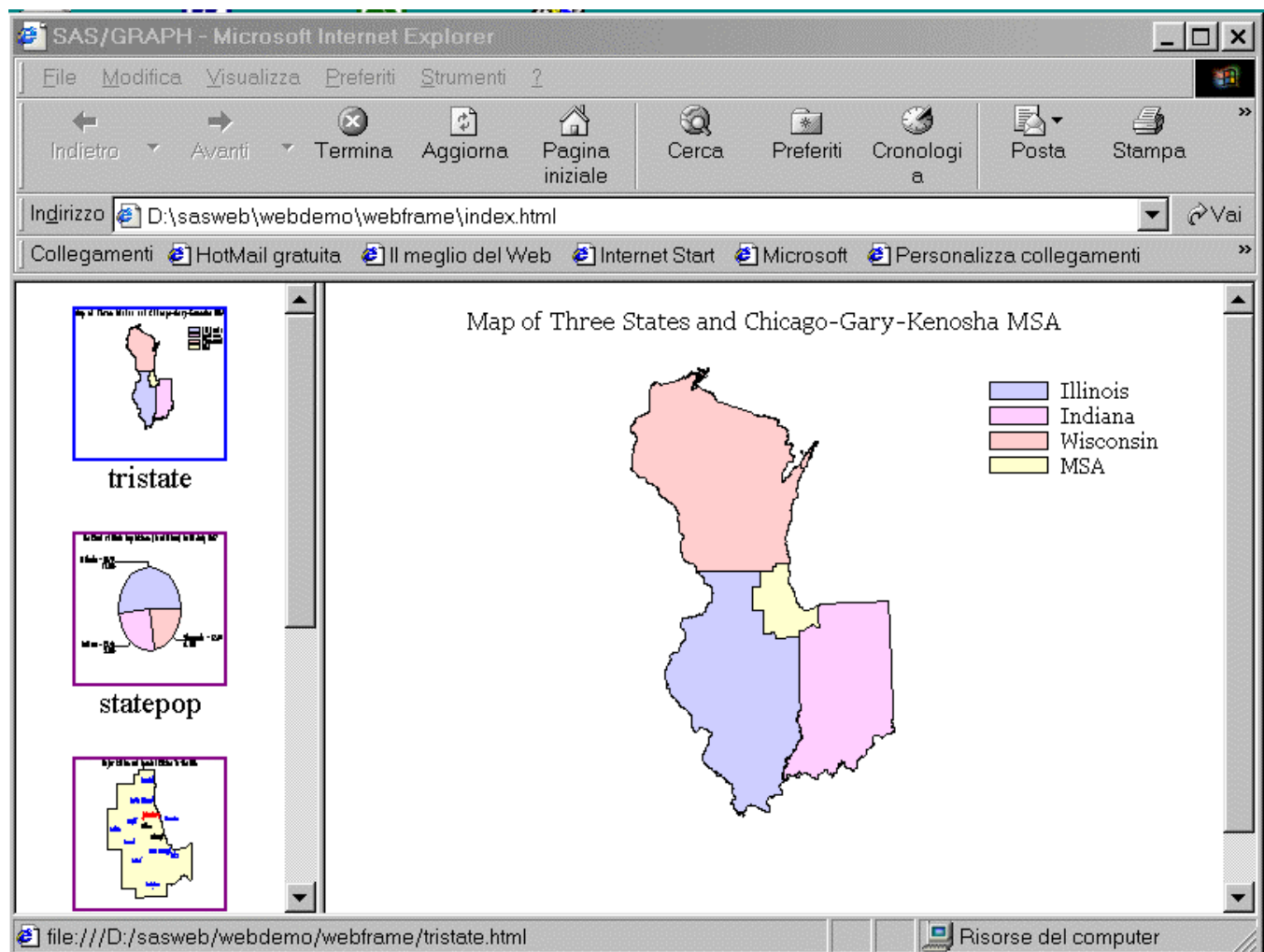
Also published by SAS Institute Inc. (Cary, N.C., USA, 1999): *The Complete Guide to the SAS Output Delivery System, Version 8* and *SAS/GRAPH Software: Reference, Version 8*.

Compatibility Notes

Netscape Navigator 4.0 does not support the flyover text as implemented with here ODS. Also, it shows grid lines for the tables, unlike Internet Explorer, which we used during application development. For a future edition of the code, we expect to find a way to eliminate the grid for both browsers.

Appendix 1. The WebFrame Example

The focus here is on code needed to use the webframe driver, not on graph-building code.



```
/* With the webframe driver, you cannot specify text
   for top left corner of web browser frame. To replace
   the default, which is "SAS/GRAPH", you must edit
   the output html file. */
```

* Same cautions about appearance as for CrossLink program. ;

```
%let PATH = c:\YourFolder\YourSubFolder3;
* Output goes into the above named folder. ;
* After running this program, go there, and click on the index.html file. ;
* You have no control over the name of that starting-point file. ;
```

```
libname DB 'c:\YourFolderForData';
/* access and prepare input data */
```

```
/* Clear the graphics catalog */
proc greplay igout=work.gseg nofs;
delete _all_;
run; quit;
```

```

filename webfrou t "&PATH";

goptions device=webframe
          xpixels=540 ypixels=405
          nodisplay /* graphs will go only to work.gseg catalog */
          gsfname=webfrou t /* used later */
          gsfmode=replace;

/* With the webframe driver, there can be no hyperlinks.
   So, the footnotes in the CrossLink and TOC examples,
   which either talked about drill-down options,
   or contained links to other web pages,
   take up no vertical space.
   So, increase xpixels= and ypixels= in goptions above. */

/* Now, insert all the code to invoke various SAS/GRAPH procs.
   The webframe driver cannot do anything with steps to build
   tabular listings (unless you use PROC GPRINT).
   Any link=, html=, html_legend= parameters will be ignored,
   if they do not actually get recognized/treated as errors.

   >>> Use name= (with limit of 8 characters, which will all
        be converted to lower case) in the SAS/GRAPH code
        where appropriate. What you assign will appear
        as the label of the thumbnail image. */

/* Here is just one example: */
/* pattern statements and legend1 definition here */
title1 h=14 pt f='Georgia'
"Map of Three States and Chicago-Gary-Kenosha MSA";
proc gmap data=DB.fourarea map=DB.fourarea;
id state;
choro state / discrete coutline=black legend=legend1
              name='tristate'; /* for the thumbnail label */
format state statnm.;
run; quit;
/* More graph creation PROCs follow. */

/* After all graphs have been built and stored in work.gseg,
   turn on the display */
goptions device=webframe display;

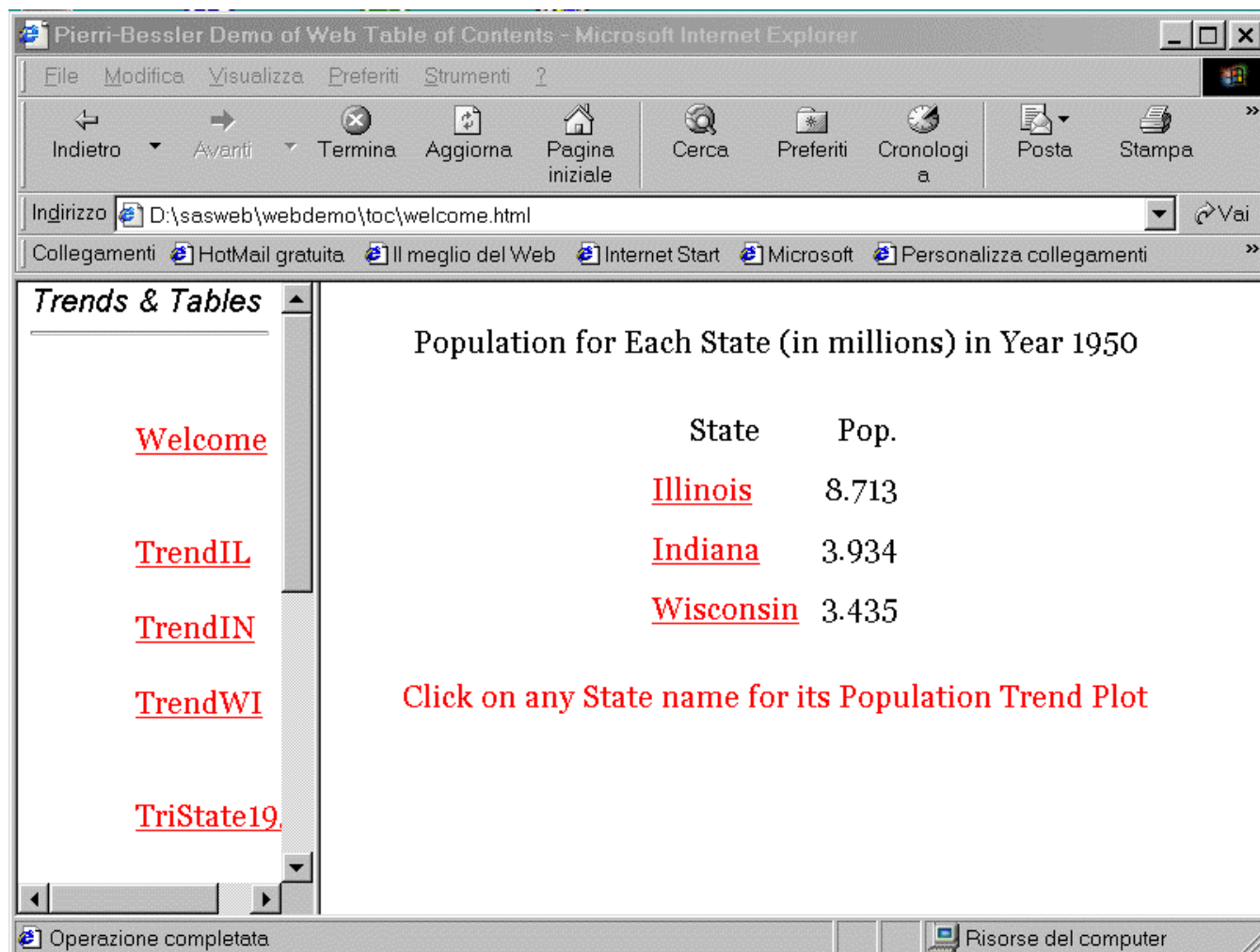
/* replay all the graphs from the work.gseg catalog
   to the previously defined gsfname=webfrou t destination */
proc greplay igout=work.gseg nofs;
replay _all_;
run;
quit;

filename webfrou t clear;

```

Appendix 2. The Table of Contents Example

The focus here is on ODS code for the TOC, not on graph/table-building code. Only one type of web page is shown. Screen capture utility would not permit showing the flyover text displayed when you “Rest mouse . . .”.



```

/* PROC TEMPLATE Code for Our Table of Contents Style */
/* Build custom style for web pages and tables */
/* for the situation where we are using a Table of Contents. */
/* Use AS THE PARENT */
/* the same styles.our_style as built for CrossLink program. */
/* THE styles.our_style MUST HAVE ALREADY BEEN BUILT */

proc template;
edit styles.our_style /* inherit everything from our_style */
    as styles.our_toc_style; /* but customize as below */
style Frame from Document /
    contentsize=24%; /* default 23% causes line breaks
                     in invisible, but "displayed",
                     useless information */
style Contents from Document /
    pagebreakhtml = _undef_; /* remove extra space
                             between TOC entries */
style ContentProcName from IndexProcName /
    bullet=none; /* delete bullet before (invisible) PROC name */
style ContentFolder /

```

```

bullet=none /* delete bullet before link label in TOC */
ListEntryAnchor=yes
PreHtml=_undef_ PostHtml=_undef_;
style ContentItem /
bullet=none /* delete bullet before link label in TOC */
ListEntryAnchor=yes
PreHtml=_undef_ PostHtml=_undef_;
style text / /* customize the title of the TOC */
'Content Title' = "Trends & Tables";
end;
run; quit;

/* Now build the web pages */

/* define the text to appear at top left of the web browser frame */
%let FRAMETTL =
Pierri-Bessler Demo of Web Table of Contents;

* Same cautions about appearance as for CrossLink program. ;

%let PATH = c:\YourFolder\YourSubFolder2;
* Output goes into the above named folder. ;
* After running this program, go there, and click on the welcome.html file.;

ods listing close;
ods noresults; /* stop output to the Results window */

libname DB 'c:\YourFolderForData';
/* access and prepare input data */

/* Build ODS style for Table of Contents implementation,
if not already done. */

/* Need all the same code as in CrossLink program to prepare
for use of %stplots and %yrtable macros.
Will not be repeated here. */

/* define a macro, no application processing yet */

%macro yrtable; /* ONLY SHOWING DIFFERENCES from CrossLink use */
%do i = 1 %to &yrcount;

ods html body = "Yr&&year&i..Tbl.html"
style = styles.our_toc_style; /* here use our TOC style */
ods proclabel ' '; /* extra line with PROC name or a customized
substitute adds no value in TOC */

/* title and footnote */

proc print data=DB.stathist(where=(year eq "&&year&i")) split='_'
contents="TriState&&year&i";
/* contents= is used in TOC link */
/* remainder of PROC invocation code same as in CrossLink program */
run;

%end;
%mend yrtable;

```

```

/* define another macro, no application processing yet */

%macro stplots; /* ONLY SHOWING DIFFERENCES from CrossLink use */
%do i = 1 %to &stcount;
  /* same macro code as in CrossLink program */
ods html body = "st&&state&i..Plt.html"
      style = styles.our_toc_style; /* here use our TOC style */
ods proclabel ' '; /* extra line with PROC name or a customized
      substitute adds no value in TOC */
  /* same macro code as in CrossLink program */
proc gplot data=DB.stathist(where=(statenam eq "&&state&i"));
plot pop*year / des="Trend&stabbrev"
      /* des= is used in TOC link */
  /* remainder of PROC invocation code same as in CrossLink program */
run; quit;
%end;
%mend stplots;

options reset=all;
/* empty the work.gseg default graphics catalog
to avoid entry-name conflicts */
proc greplay igout=work.gseg nofs;
delete _all_;
run; quit;

/* Create all the web pages now.
Each slide, table, or graph is a separate page.
Not all dumped into one big scrollable page,
as might happen with BY processing and ANCHOR= technique. */

ods html path = "&PATH" (url=none)
      contents = 'contents.html' (title="Contents Title"
      /* where used??? */)
      frame = 'welcome.html' (title="&FRAMETTL"
      /* prefix text at top left corner of browser */)
      style = styles.our_toc_style
      gtitle gfootnote
      body = 'welcbody.html' (title="Body Title"
      /* where used??? */);
ods proclabel ' '; /* label adds no value */

options reset=all;
options device=gif xpixels=440 ypixels=330;
proc gslide name='welcslid'
      des='Welcome'; /* des= is used in TOC link */
  /* title and note statements for the Welcome page */
run; quit;
options reset=all;
%stplots
run;
options reset=all;
%yrtables
run;
ods html close; /* DO NOT close the html destination until here */

```

Appendix 3. The CrossLink Example

Note: All titles and footnotes in the web pages shown here were designed and coded to be a uniform size. Variations appeared when viewing with a browser on a screen-capture computer other than the one that was used to create the web pages. However, at least one computer other than the creation computer displayed the web pages with no alteration of text sizes.

This is initializaton code only. Details for individual web pages are shown on following pages.

```
/* specify text for top left corner of web browser frame */
%let BODYTTL =
Pierri-Bessler Demo of Graphs & Tables on the Web;

* Appearance of output as designed by the authors is affected by screen size
  and resolution of the display monitor used for development. ;
  Here the monitor was 14-inch (35.6-cm) with resolution set to 800 X 600. ;
* Also a factor is web browser text size used y the authors when viewing
  results during development. For us, View Text Size was set to Medium. ;

%let PATH = c:\YourFolder\YourSubFolder1;
* output goes into the above named folder. ;
* After running this program, go there,
  and click on the welcome.html file. ;
* >>> If your browser does not have text size "Medium",
  results you see may be counter to our design.
  (Check by clicking View and then Text Size.) ;

ods listing close;
ods noresults; /* especially when creating many pages,
               stop output to the Results window */

libname DB 'c:\YourFolderForData';
/* access and prepare input data */

/* circumvent a problem with GIFANIM driver */
libname gdevice0 'c:\FP_LeRB\fixgdev';run;
proc gdevice nofs cat=gdevice0.devices;
delete fixanim;
copy gifanim from=sashelp.devices newname=fixanim;
modify fixanim hsize=0 vsize=0;
run; quit;

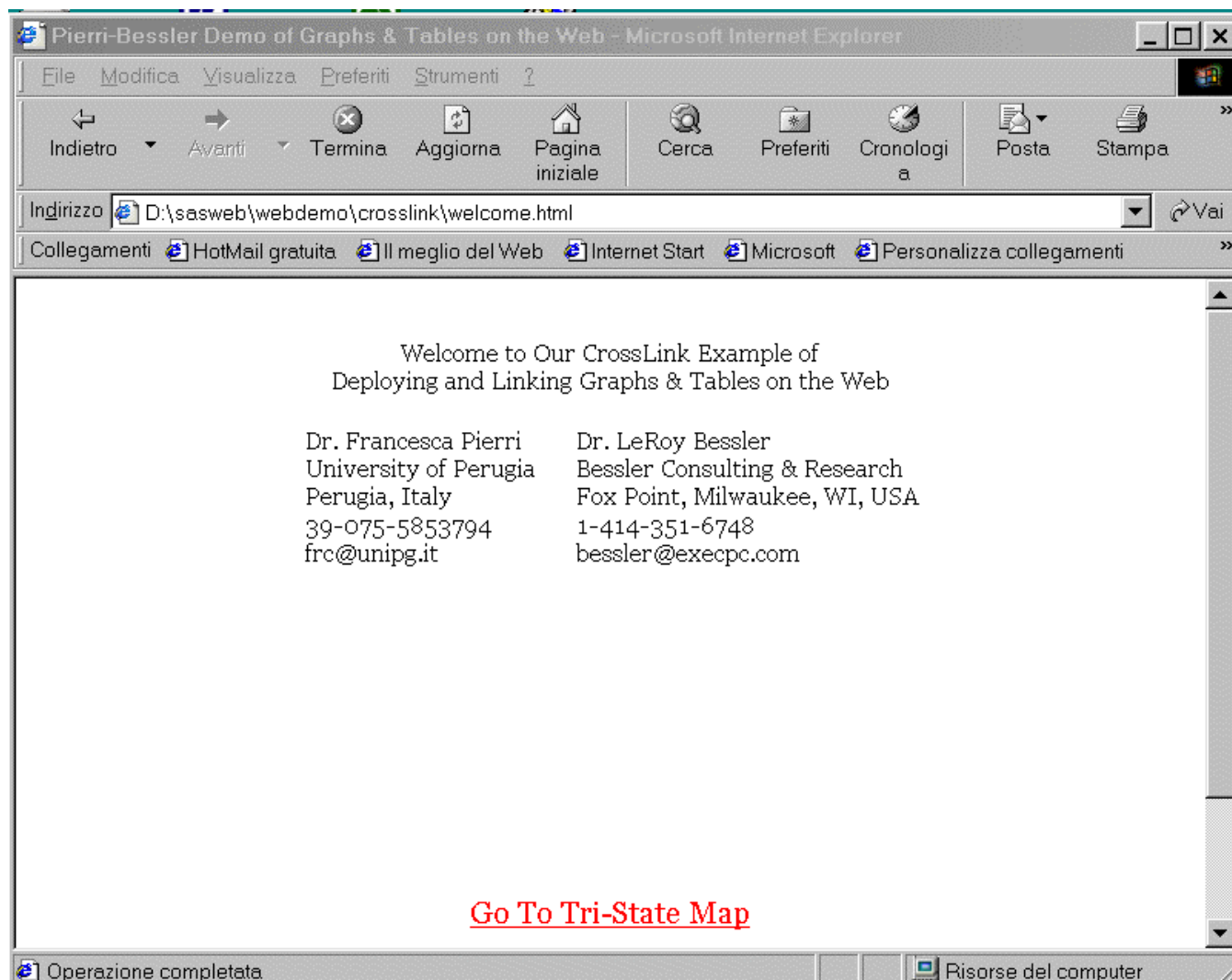
/* build a fake data set used for a trick later */
data DB.lnksonly;
length heading $ 1;
label heading='00'X;
heading = ' ';
run;

/* If not already built, use PROC TEMPLATE to build
  our style for web pages and tables, and the special
  style to be used with prints of DB.lnksonly */
```

In the web page examples that follow, many of the web pages are drillable and/or drilldown “destinations”. However, every web page has one or more clickable links at the bottom to take you to other pages. When the web page is a table, those links are defined with link= in a SAS FOOTNOTE statement. When the web page is a graph, the graph image is immediately followed by a PROC PRINT of the “data” in DB.lnksonly with a SAS TITLE statement that uses link=. The reason for this trick has been explained in the earlier “Design and Implementation” section of this paper.

Welcome Page: text-only version

This simply links to *next* web page in the application—it is not meant as a “main menu”.

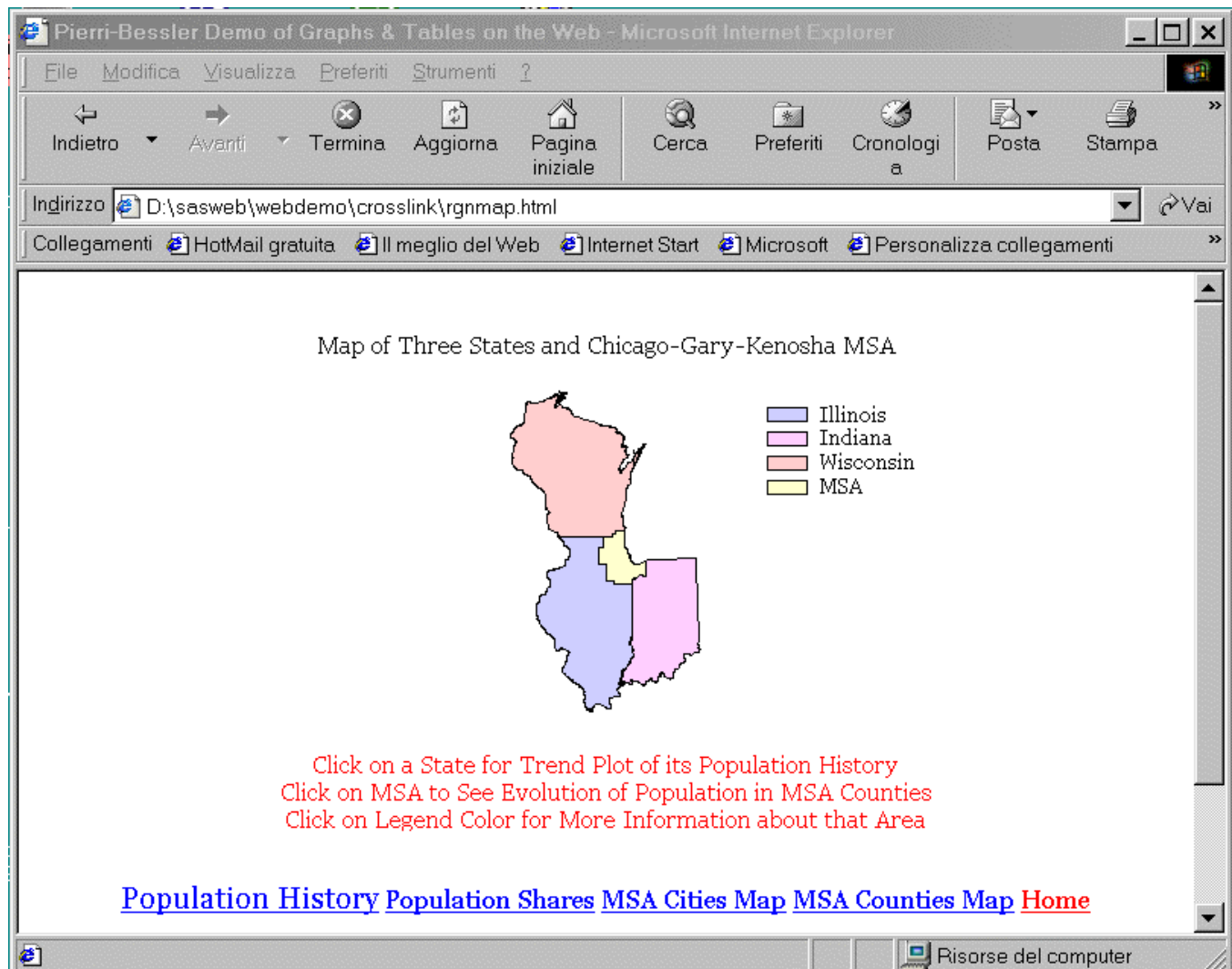


```
goptions reset=all;
ods html path = "&PATH" (url=none) /* folder for the body data set */
        body = 'welcome.html'
            (title="&BODYTTTL") /* custom title for browser frame */
            style = styles.our_style /* style built with PROC TEMPLATE */
            gtitle gfootnote; /* display this text inside the graph area */
goptions device=gif xpixels=440 ypixels=330;
proc gslide name='welcslid';
  /* titles, notes, footnotes for Welcome page */
run; quit;
  /* now provide link to next web page */
ods html style=styles.lnktable;
title1 h=10 pt f='Georgia'
link="tristate.html" /* define the link */
'Go To Tri-State Map'; /* label the link */
footnote1; /* erase any possible leftover footnotes */
proc print data=DB.linksdgd noobs;
run; /* no data or column label to show, just links */
ods html close;
```

Drillable Map of the TriState Area, with the Principal MSA as an Interstate Inset

Map areas link to tables of information about each county,

legend color blocks link to slides of information about each associated area.



```
/* define links to information about each map area */
/* define links to state population trend plots
   and to animated map of MSA county population change */
data DB.fourarea;
set DB.fourarea;
length imglink $ 40;
length infolink $ 40;
if state eq 17 then do;
    imglink = 'href="stIllinoisPlt.html"';
    infolink = 'href="stIllinoisInfo.html"';
end;
else
if state eq 18 then do;
    imglink = 'href="stIndianaPlt.html"';
    infolink = 'href="stIndianaInfo.html"';
end;
else
if state eq 55 then do;
```

```

    imglink = 'href="stWisconsinPlt.html"';
    infolink = 'href="stWisconsinInfo.html"';
end;
else
if state eq 99 then do;
    imglink = 'href="mapanim.html"';
    infolink = 'href="stmsaInfo.html"';
end;
run;
goptions reset=all;
ods html path = "&PATH" (url=none) /* folder for the body data set */
    body = 'tristate.html'
        (title="&BODYTTL") /* custom title for browser frame */
    style = styles.our_style /* style built with PROC TEMPLATE */
    gtitle gfootnote;
goptions device=gif xpixels=440 ypixels=330;
pattern1 v=msolid c=CXCCCCFF; /* browser-safe lightest blue */
pattern2 v=msolid c=CXFFCCFF; /* browser-safe lightest magenta */
pattern3 v=msolid c=CXFFCCCC; /* browser-safe lightest red */
pattern4 v=msolid c=CXFFFFCC; /* browser-safe lightest yellow */
legend1 across = 1
    label = none /* label would add no value */
    shape = bar (1 cm, 2.5 pct)
    value = (f='Georgia' /* web-friendly font */
        c=CX000000 /* browser-safe black */
        h=12 pt
        'Illinois' 'Indiana' 'Wisconsin' 'MSA')
    mode = protect
    position = (top right inside);
title1 h=14 pt f='Georgia'
"Map of Three States and Chicago-Gary-Kenosha MSA";
footnote1 h=14 pt f='Georgia' c=CXFF0000
'Click on a State for Trend Plot of its Population History';
footnote2 h=14 pt f='Georgia' c=CXFF0000
'Click on MSA to See Evolution of Population in MSA Counties';
footnote3 h=14 pt f='Georgia' c=CXFF0000
'Click on Legend Color for More Information about that Area';
proc gmap data=DB.fourarea map=DB.fourarea;
id state;
choro state / discrete coutline=black legend=legend1
    html=imglink /* link for click on area */
    html_legend=infolink /* link for click
                        on legend color */
    name='tristate';
format state statnm.;
run; quit;
/* now provide links to other web pages */
ods html style=styles.lnktable;
title1 h=10 pt f='Georgia'
link="st3tbl.html" 'Population History'
link="st3pie.html" 'Population Shares'
link="msacitymap.html" 'MSA Cities Map'
link="msacntypopmap.html" 'MSA Counties Map'
link="welcome.html" 'Home';
footnote1;
proc print data=DB.lnkonly noobs;
run; /* no data or column label to show, just links */
ods html close;

```

Drillable Table to Compare Population in the Three States over Several Decades

State row labels link to trend plots of population for each state.

History of State Populations (in millions)
Population of the Interstate MSA Has Been Apportioned

State	1997 Pop.	1990 Pop.	1980 Pop.	1970 Pop.	1960 Pop.	1950 Pop.
Illinois	11.896	11.431	11.427	11.114	10.081	8.713
Indiana	5.864	5.544	5.490	5.194	4.662	3.934
Wisconsin	5.170	4.892	4.706	4.418	3.952	3.435

Click on any State name for its Population Trend Plot
[Tri-State Population Shares](#) [Tri-State Map](#)

```
%macro makelink(data=,var=,varprefix=,  
                varsuffix=,htmlsuffix=,fmtname=);
```

```
/* This macro enables you to make the display of the value  
   of a variable into a link labelled with that value, if you  
   later invoke/apply the format where you want the link.  
   The macro dynamically creates the format, providing a link  
   for every distinct value of the variable in the data set. */
```

```
proc sort data=&data out=toprep nodupkey;  
by &var;  
run;
```

```
data toformat;  
length bodyname $ 40;  
set toprep;  
retain fmtname "&fmtname";  
value = &var;  
rename &var = start;
```

```

bodyname = "&varprefix"||value||"&varsuffix..html&htmlsuffix";
label =
'<a '||'href="'||compress(bodyname)||'">'||trim(&var)||'</a>';
run;

proc format cntlin=toformat;
run; /* to see result, use proc format fmtlib lib=work;run; */

%mend makelink;

/* %makelink MUST PRECEDE the creation of st3tbl.html
and the INVOCATION of %yrtabls */
%makelink(data=DB.statepop,var=statenam,varprefix=st,
varsuffix=Plt,htmlsuffix=,fmtname=$lnkplt)
run;

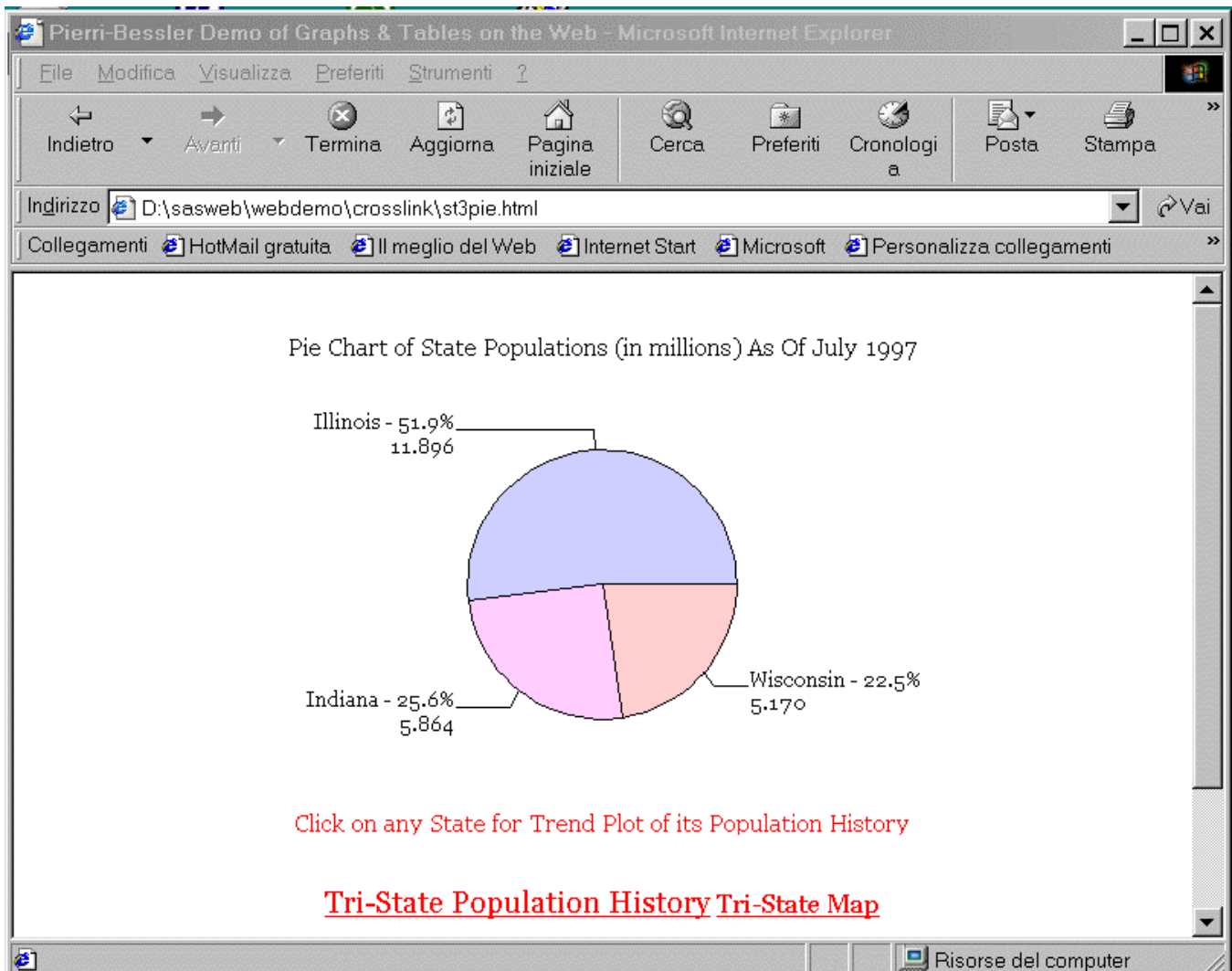
ods html path = "&PATH" (url=none) /* folder for the body data set */
body = 'st3tbl.html'
(title="&BODYTTL") /* custom title for browser frame */
style = styles.our_style; /* style built with PROC TEMPLATE */

title1 h=10 pt f='Georgia'
"History of State Populations (in millions)";
footnote1 h=10 pt f='Georgia' c=CXFF0000
'Click on any State name for its Population Trend Plot';
footnote2 h=10 pt f='Georgia'
link="st3pie.html" 'Tri-State Population Shares'
link="tristate.html" 'Tri-State Map';
proc print data=DB.statepop split='_';
id statenam;
var stpop97 stpop90 stpop80 stpop70 stpop60 stpop50;
format statenam $lnkplt.; /* this makes the state name label a drilldown link */
run;
ods html close;

```

Drillable Pie Chart of Population by Sliced by State

Pie slices link to trend plots of population for the states.



```
/* define links to each state's pop. trend plot */
data DB.pop4pie;
set DB.pop4pie;
length imglink $ 40;
if state eq 17 then imglink = 'href="stIllinoisPlt.html"';
else
if state eq 18 then imglink = 'href="stIndianaPlt.html"';
else
if state eq 55 then imglink = 'href="stWisconsinPlt.html"';
run;
options reset=all;
ods html path = "&PATH" (url=none) /* folder for the body data set */
        body = 'st3pie.html'
        (title="&BODYTTTL") /* custom title for browser frame */
        style = styles.our_style /* style built with PROC TEMPLATE */
        gtitle gfootnote; /* display this text inside the graph area */
options device=gif xpixels=440 ypixels=330;
/* control the text displayed for
   pie slice name, value, and percent */
```

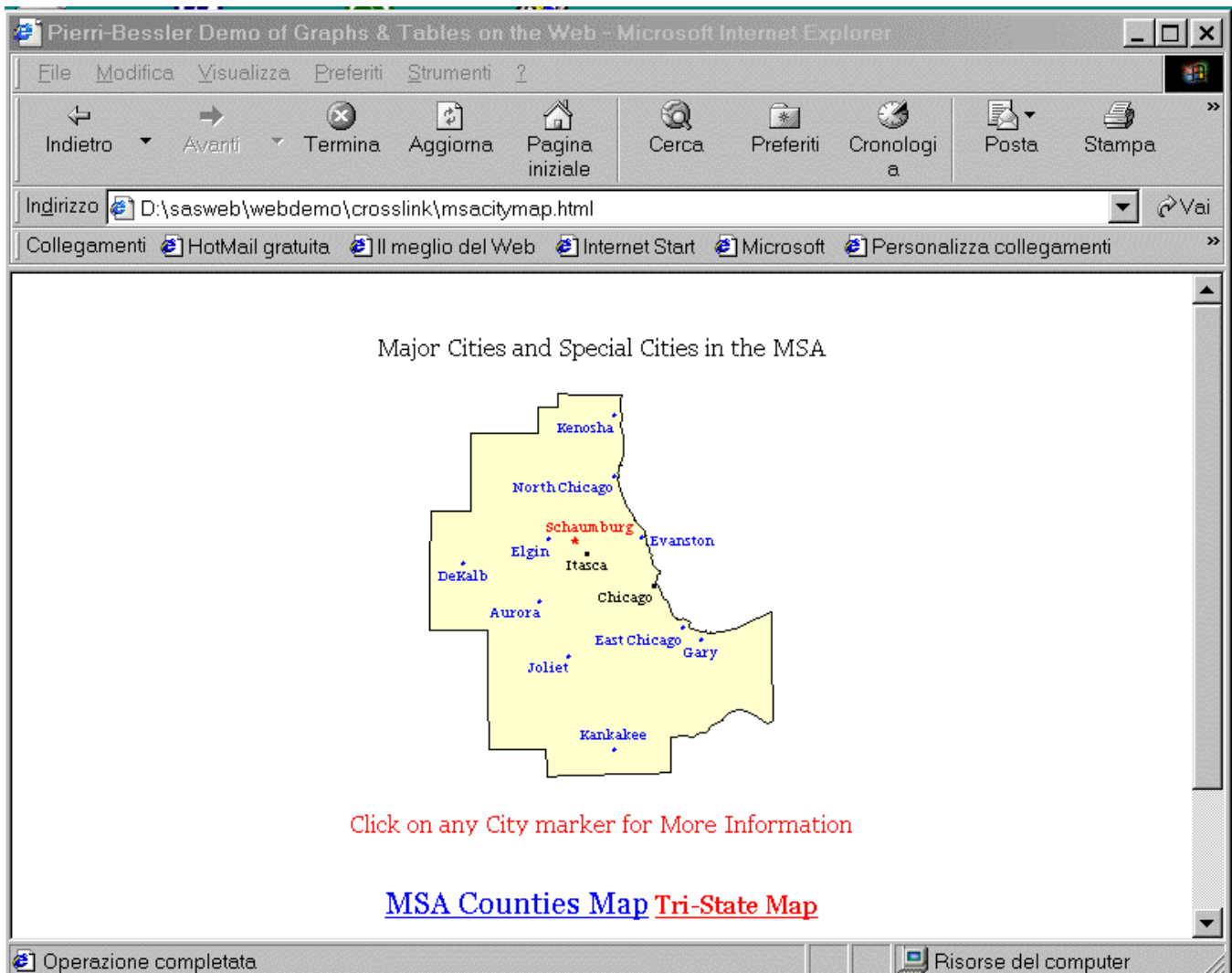
```

goptions ctext=CX000000 htext=12 pt ftext='Georgia';
/* make the colors used for each state in pie same as used in map */
goptions reset=pattern;
pattern1 v=psolid c=CXCCCCFF;
pattern2 v=psolid c=CXFFCCFF;
pattern3 v=psolid c=CXFFCCCC;
title1 h=14 pt f='Georgia'
"Pie Chart of State Populations (in millions) As Of July 1997";
footnote1 h=14 pt f='Georgia' c=CXFF0000
'Click on any State for Trend Plot of its Population History';
proc gchart data=DB.pop4pie;
pie statwpct / sumvar=stpop97 /* preprocessing built stpop97
                                as concatenation of state name
                                and state's percent of total */
                                descending /* rank the states, but adjust the
                                            order of patterns so that states
                                            have same color as in map --
                                            "color code" the states
                                            for communication consistency */
                                discrete
                                noheading /* fixed default heading has no value */
                                slice=arrow value=arrow
                                percent=none /* SAS/GRAPH format of percent
                                                is unacceptable */
                                coutline=black
                                html=imglink /* link each state to its pop. trend plot */
                                name='st3pie';
run; quit;
/* provide links to other web pages */
ods html style=styles.lnktable;
title1 h=10 pt f='Georgia'
link="st3tbl.html" 'Tri-State Population History'
link="tristate.html" 'Tri-State Map';
footnote1;
proc print data=DB.lnksonly noobs;
run;
ods html close;

```

Drillable Map of Cities of the MSA

with links to tables of information about each city.



```
/* Prior processing created an annotate data set
   with one record for each city name label
   and one record for each city site marker.
   Now make those markers clickable links (use html=)
   to tables of information about the cities. */
data DB.citylink;
set DB.annocity;
length html $ 40;
if length(text) eq 1 /* this is a marker label not a city label */
then html = 'href="city" || trim(left(city)) || 'Tbl.html"';
else html = ' ';
run;

goptions reset=all;

ods html path = "&PATH" (url=none) /* folder for the body data set */
        body = 'msacitymap.html'
              (title="&BODYTTTL") /* custom title for browser frame */
        style = styles.our_style /* style built with PROC TEMPLATE */;
```

```

gttitle gfootnote;

goptions device=gif xpixels=440 ypixels=330;

goptions reset=pattern;
pattern v=msolid c=CXFFFFCC; /* browser-safe lightest yellow
    provides maximum contrast non-white background for
    city marker and name annotation on the map */

title1      h=14 pt f='Georgia'
'Major Cities and Special Cities in the MSA';
footnote1 h=14 pt f='Georgia' c=CXFF0000
'Click on any City marker for More Information';

proc gmap map=DB.msamapnc data=DB.msamapnc
    anno=DB.citylink; /* annotated city markers are links */
id state;
choro fakevar / nolegend /* one area, no response variable,
    so no use for a legend */
    coutline=black
    name='citymap';
run; quit;

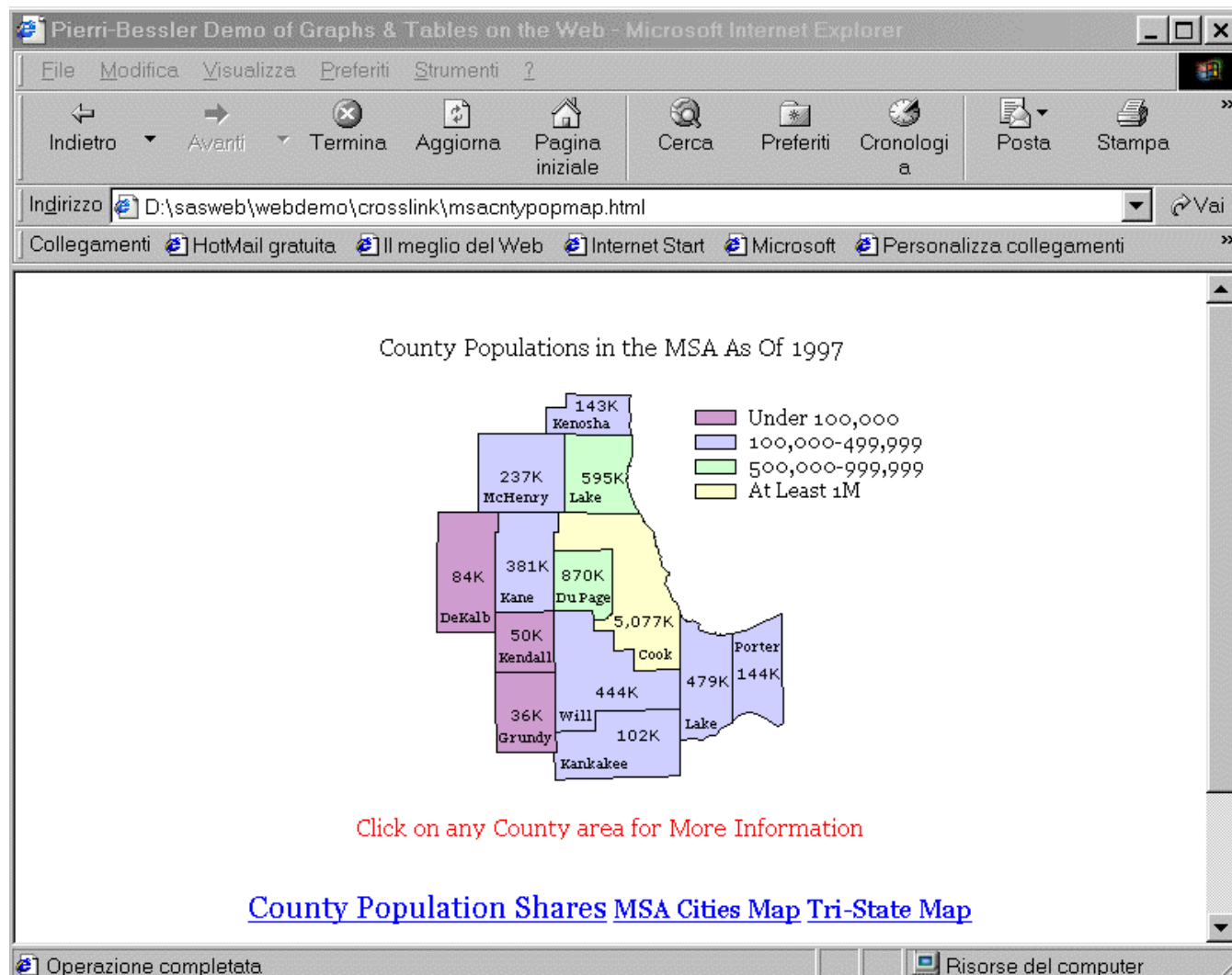
/* provide links to other web pages */
ods html style=styles.lnktable;
title1 h=10 pt f='Georgia'
link="msactypopmap.html" 'MSA Counties Map'
link="tristate.html"      'Tri-State Map';
footnote1;
proc print data=DB.lnksonly noobs;
run;

ods html close;

```

Drillable Map of Populations in Counties of the MSA

with links to tables of information about each county.



```
goptions reset=all;
```

```
ods html path = "&PATH" (url=none) /* folder for the body data set */
         body = 'msacntypopmap.html'
           (title="&BODYTTTL") /* custom title for browser frame */
         style = styles.our_style /* style built with PROC TEMPLATE */
         gtitle gfootnote;
```

```
goptions device=gif xpixels=440 ypixels=330;
```

```
legend1 across = 1
      label = none
      shape = bar (6 pct, 2.8 pct)
      value = (f='Georgia' c=CX000000 h=12 pt)
      mode = protect
      position = (top right inside);
```

```
pattern1 v=msolid c=CXCC99CC; /* browser-safe light purple */
pattern2 v=msolid c=CXCCCCFF; /* browser-safe lightest blue */
```

```

pattern3 v=msolid c=CXCCFFCC; /* browser-safe lightest green */
pattern4 v=msolid c=CXFFFFCC; /* browser-safe lightest yellow */

title1      h=14 pt f='Georgia'
"County Populations in the MSA As Of 1997";
footnote1 h=14 pt f='Georgia' c=CXFF0000
'Click on any County area for More Information';

proc gmap map=DB.msamapwpc data=DB.coupop anno=DB.annocnty;
id state county;
choro couJUL97 / discrete legend=legend1 coutline=black
                 html=coudrill /* link each county to its info table */
                 name='cnpopmap';
format couJUL97 rangtxt.;
run; quit;

/* provide links to other web pages */
ods html style=styles.lnktable;
title1 h=10 pt f='Georgia'
link="msacntypoprak.html" 'County Population Shares'
link="msacitymap.html"    'MSA Cities Map'
link="tristate.html"      'Tri-State Map';
footnote1;
proc print data=DB.lnksonly noobs;
run;

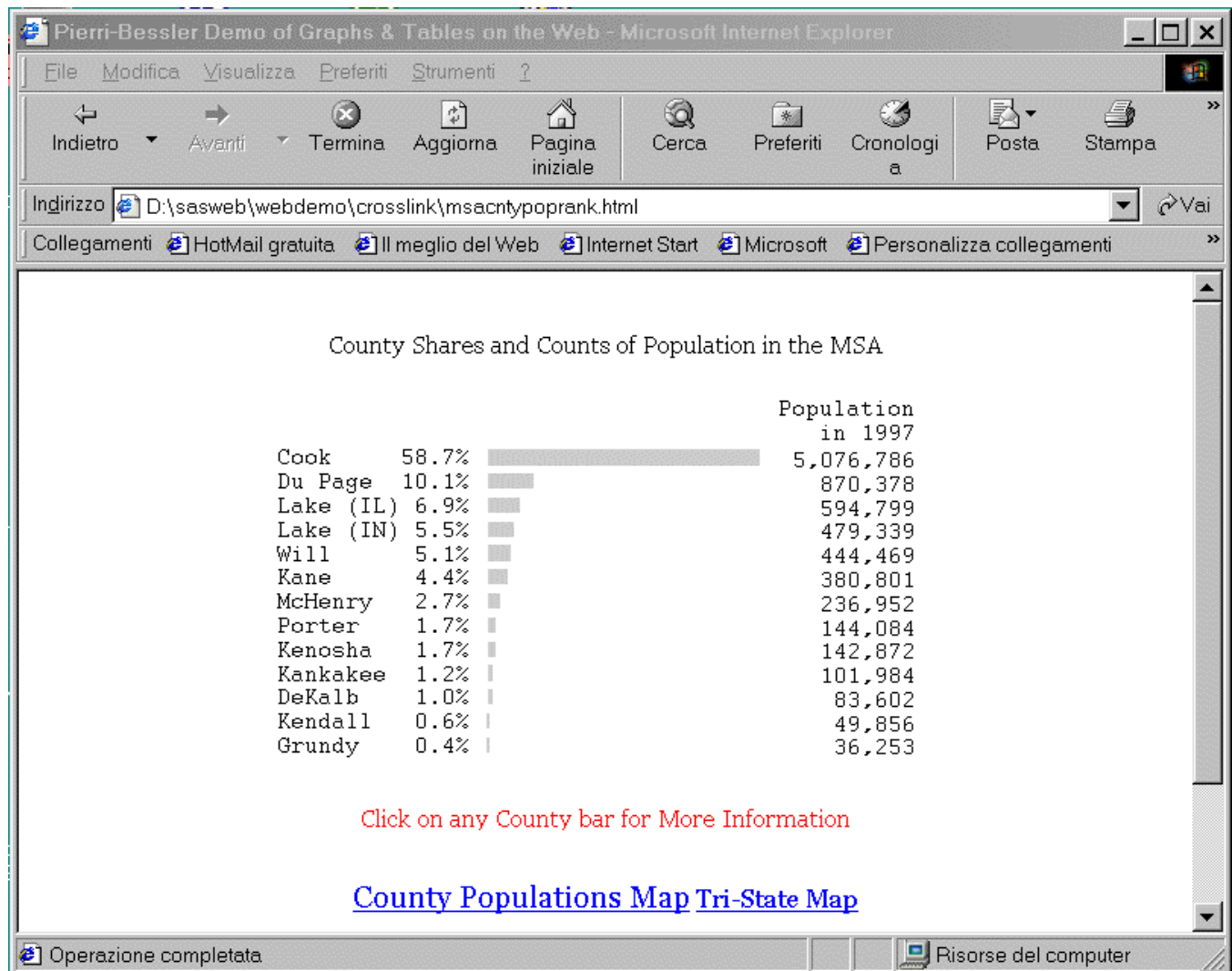
ods html close;

```

Drillable, Ranked, Maximally Informative, Horizontal Bar Chart

with links to tables of information about each county.

It also provides the pie chart function of shares of whole—there would be too many slices.



```
proc sort data=DB.coupop out=sorted;
by state county;
run;
data DB.stcoupop;
length barlabel $ 14;
if _N_ eq 1 then set msapop; /* get total pop. */
set sorted;
/* format beginning of bar label with county name */
barlabel = put(put(state,z2.)||put(county,z3.),$stcntnm.);
/* format end of bar label with percent of whole */
/* temporarily force leading zero, if needed,
to line up all percents */
substr(barlabel,10,5) =
put(((couJUL97/msapop) * 100),z4.1) || '%';
/* now replace leading zero, if any, with a blank */
if substr(barlabel,10,1) eq '0'
then substr(barlabel,10,1) = ' ';
run;
```

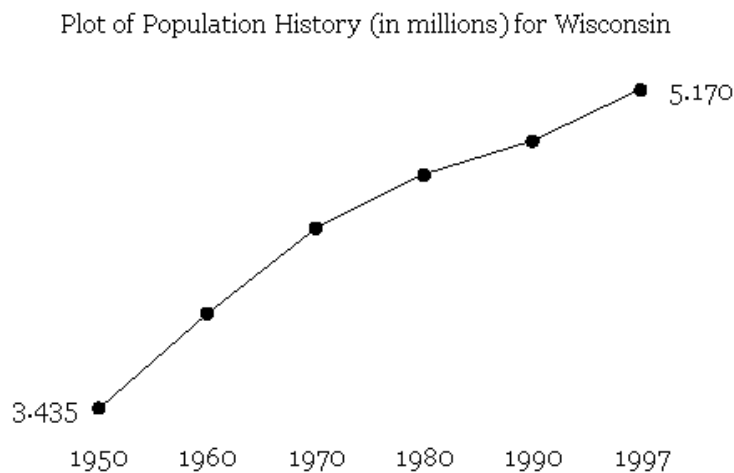
```

goptions reset=all;
ods html path = "&PATH" (url=none) /* folder for the body data set */
      body = 'msacntypoprank.html'
            (title="&BODYTTTL") /* custom title for browser frame */
            style = styles.our_style /* style built with PROC TEMPLATE */
            gtitle gfootnote;
goptions device=gif xpixels=440 ypixels=330;
/* Control the text displayed for
   Bar values at left and bar values in table at right side.
   hbar gives you a right-justified table at end of bars.
   Use a fixed-width font to line up the percents of whole
   in the bar labels, and digits in the table entries.
   Had a problem with Courier New. Used Courier instead. */
goptions ctext=CX000000 htext=14 pt ftext='Courier';
goptions reset=pattern;
pattern1 v=solid c=CXCCCCC; /* browser-safe lightest grey. */
/* Black bars would dominate the image.
   White bars with black outlines can create an image
   where the bars are confused with the spaces. */
/* no midpoint axis & no label for it */
axis1 style=0 label=none;
/* no response axis & no label for it */
axis2 style=0 label=none
/* no tick marks and no tick mark values */
major=none minor=none value=none
/* Start axis at zero.
   Retrieve the pre-computed-and-saved-as-global-macro-variable
   maximum county population to size the axis to fit. Define no
   intermediate tick marks that would be invisible anyhow */
order=(0 &maxJUL97);
title1 h=14 pt f='Georgia'
'County Shares and Counts of Population in the MSA';
footnote1 h=14 pt f='Georgia' c=CXFF0000
'Click on any County bar for More Information';
proc gchart data=DB.stcoupop;
hbar barlabel / sumvar=couJUL97 /* name and percent of whole */
               sumlabel='Population in 1997' /* heading for the
               column of bar values at right side of graph */
               descending /* rank the counties */
               noframe /* no added value */
               maxis=axis1 raxis=axis2
               width=0.7 /* try to match height of bar label */
               name='cnpopbar'
               space=0.5 /* adjust as you like */
               html=coudrill; /* link each bar to its county info table */
format couJUL97 comma9.;
run; quit;
/* now provide links to other web pages */
ods html style=styles.lnktable;
title1 h=10 pt f='Georgia'
link="msacntypopmap.html" 'County Populations Map'
link="tristate.html" 'Tri-State Map';
footnote1; /* erase any possible leftover footnotes */
proc print data=DB.lnksonly noobs;
run;
ods html close;

```

Plot points link to tables comparing population of the three states in that year.

Intermediate trend points display population for the year as “flyover” text, but the screen capture utility would not permit showing the flyover text displayed when you “Rest mouse . . .”.



Click any point for Table of that Year's Population by State

[Tri-State Population History](#) [Tri-State Population Shares](#) [Tri-State Map](#)

```
/* alt= below is the flyover text for trend points */  
/* href= below is the drill-down destination page for trend points */  
data DB.stathist(keep=statenam year pop yrdrill);  
length yrdrill $ 40;  
label statenam='State' year='Year' pop='Pop.';  
set DB.statepop;  
year = '1997'; pop = stpop97;  
yrdrill = 'href="Yr1997Tbl.html"'; /* start-point value on axis,  
no flyover text needed */  
  
output;  
year = '1990'; pop = stpop90;  
yrdrill = 'href="Yr1990Tbl.html" alt="||put(pop,6.3)||"' ;  
output;  
/* do 1980, 1970, 1960 analogously to 1990 */  
year = '1950'; pop = stpop50;  
yrdrill = 'href="Yr1950Tbl.html"'; /* end-point value on axis,  
no flyover text needed */
```

```

output;
run;

proc sort data=DB.stathist(keep=statenam) out=sorted nodupkey;
by statenam;
run;

data _null_;
set sorted end=lastone;
call symput('state' || trim(left(_N_)), trim(left(statenam)));
if lastone;
call symput('stcount', trim(left(_N_)));
run;

/* empty the work.gseg graphics catalog
to prevent entry-name conflict */
proc greplay igout=work.gseg nofs;
delete _all_;
run; quit;

goptions reset=all;

%macro stplots;
%do i = 1 %to &stcount;

proc means noprint data=DB.stathist(where=(statenam eq "&&state&i")) min max;
var pop;
output out=minmax min=minpop max=maxpop;
run;

data _null_;
set minmax;
call symput('realminy', minpop);
call symput('tickminy', trim(left(put(round(minpop, 0.001), 6.3))));
call symput('realmaxy', maxpop);
call symput('tickmaxy', trim(left(put(round(maxpop, 0.001), 6.3))));
call symput('range_y', (maxpop - minpop));
run;

ods html path = "&PATH" (url=none) /* folder for the body data set */
body = "st&&state&i..Plt.html"
(title="&BODYTTTL") /* custom title for browser frame */
style = styles.our_style /* style built with PROC TEMPLATE */
gtitle gfootnote; /* display this text inside the graph area */

%let stabbrev = %upcase(%substr(&&state&i, 1, 2));

goptions device=gif xpixels=440 ypixels=330;

axis1 label=none minor=none major=(color=white) style=0
value = (c=CX000000 h=14 pt f='Georgia');
axis2 label=none minor=none major=(color=white) style=0
order = &realminy to &realmaxy by &range_y
value = (c=CX000000 h=14 pt f='Georgia' "&tickminy" " ");
axis3 label=none minor=none major=(color=white) style=0
order = &realminy to &realmaxy by &range_y
value = (c=CX000000 h=14 pt f='Georgia' " " "&tickmaxy");

symbol1 v=dot i=join c=CX000000; /* the real plot line */

```

```

symbol2 v=none i=none c=CXFFFFFF; /* invisible line for plot2
      plot2 is needed to have the right-hand-side vertical axis */

title1      h=14 pt f='Georgia'
"Plot of Population History (in millions) for &&state&i";
footnote1 h=14 pt f='Georgia' c=CX0000FF
"Rest mouse on any point for value of that Year's Population";
footnote2 h=14 pt f='Georgia' c=CXFF0000
"Click any point for Table of that Year's Population by State";

proc gplot data=DB.stathist(where=(statenam eq "&&state&i"));
plot  pop*year / noframe haxis=axis1 vaxis=axis2
      name="st&stabbrev.plot"
      html=yrdrill; /* link each year's point to the
      table of population by state for that year */
plot2 pop*year / noframe haxis=axis1 vaxis=axis3;
run; quit;

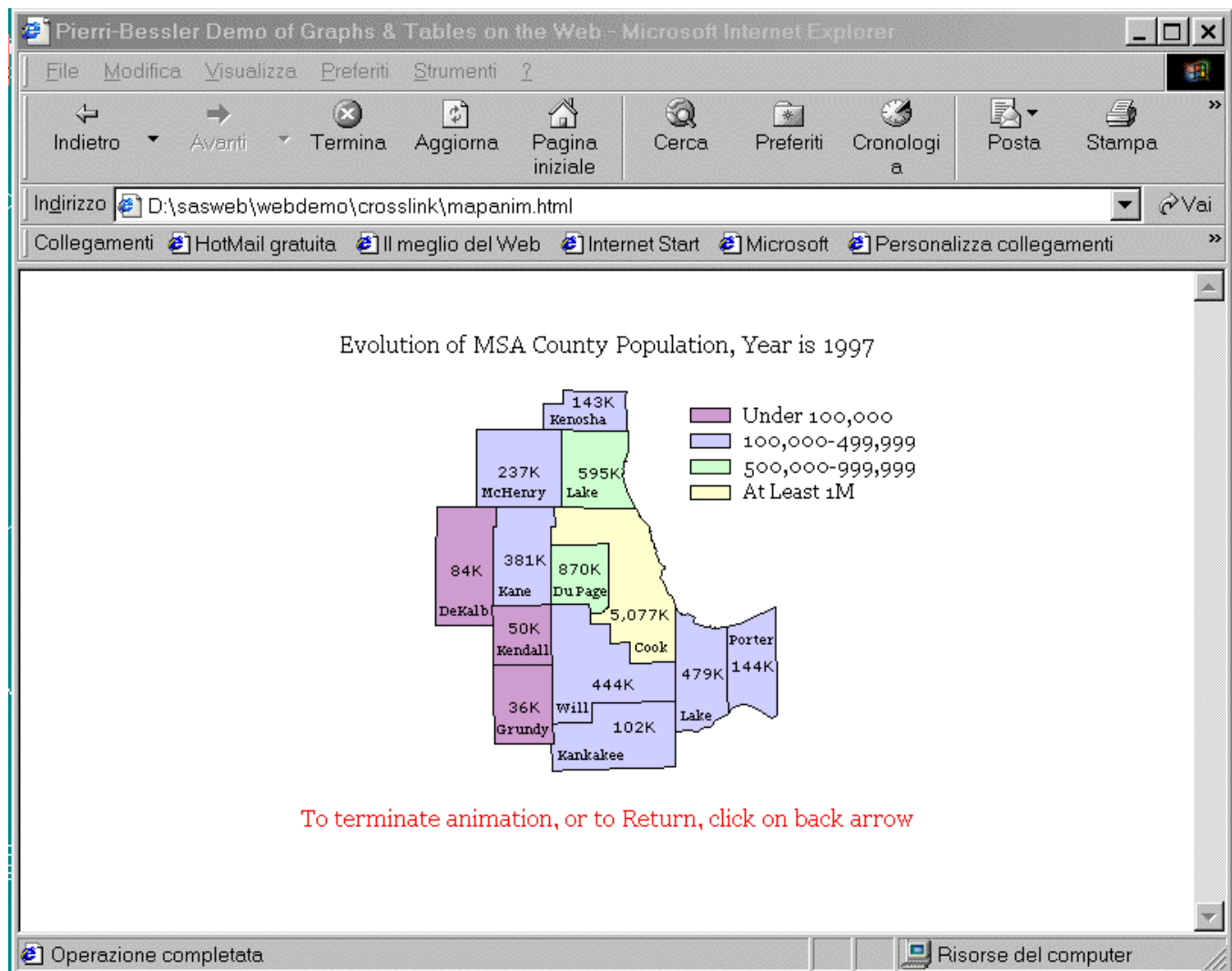
/* provide links to other web pages */
ods html style=styles.lnktable;
title1 h=10 pt f='Georgia'
link="st3tbl.html"      'Tri-State Population History'
link="st3pie.html"      'Tri-State Population Shares'
link="tristate.html"    'Tri-State Map';
footnote1;
proc print data=DB.lnkonly noobs;
run;

ods html close;
%end;
%mend stplots;

%stplots
run;

```

Animated Map for Display on the Web



```

goptions reset=all;
/* Clear the graphics catalog */
proc greplay igout=work.gseg nofs;
delete _all_;
run; quit;
ods html path = "&PATH" (url=none) /* folder for the body data set */
        body = 'mapanim.html'
            (title="&BODYTTTL") /* custom title for browser frame */
            style = styles.our_style /* style built with PROC TEMPLATE */
            gtitle gfootnote;
goptions device=gif xpixels=440 ypixels=330;
/* Create a placeholder. Title will not display. */
proc gslide name='mapanim';
title1 'animated gif placeholder';
run; quit;
ods html close;
goptions reset=all;
ods listing;
/* Create the animated gif using the same file */
filename animout "&PATH.\mapanim.gif";
goptions dev=fixanim xpixels=440 ypixels=330

```

```

        gsfname=animout
        iteration=1 delay=100;
    /* use county population range colors same as in static map */
pattern1 v=msolid c=CXCC99CC; /* browser-safe light purple */
pattern2 v=msolid c=CXCCCCFF; /* browser-safe lightest blue */
pattern3 v=msolid c=CXCCFFCC; /* browser-safe lightest green */
pattern4 v=msolid c=CXFFFFCC; /* browser-safe lightest yellow */
legend1 across    = 1
      label      = none
      shape      = bar (6 pct, 2.8 pct)
      mode       = protect
      position   = (top right inside)
      order      = ('1' '2' '3' '4')
      value      = (c=CX000000 h=12 pt f='Georgia'
                    'Under 100,000' '100,000-499,999'
                    '500,000-999,999' 'At Least 1M');
proc sort data=DB.cntyhist(keep=year) out=sorted nodupkey;
by year;
run;
data _null_;
set sorted end=lastone;
call symput('cyear' || trim(left(_N_)), put(year, $4.));
if lastone;
call symput('cyrcount', trim(left(_N_)));
run;
%macro yrmaps;
%do i = 1 %to &cyrcount;
%if &i eq 1 %then %do;
goptions gsfmode=replace;
%end;
%else %do;
goptions gsfmode=append;
%end;
title1      h=14 pt f='Georgia'
            "Evolution of MSA County Population, Year is &&cyear&i";
footnote1 h=14 pt f='Georgia' c=CXFF0000
          "To terminate animation, or to Return, click on back arrow";
proc gmap map=DB.msamapwc
      data=DB.cntyhist(where=(year eq "&&cyear&i"))
      anno=DB.annocouh(where=(year in ("&&cyear&i" ' ' ')));
id state county;
choro popcat / discrete legend=legend1 coutline=black
              midpoints='1' '2' '3' '4';
run; quit;
%end;
%mend yrmaps;
%yrmaps
run;
/* end the animation */
data _null_;
file animout recfm=n mod;
put '3B'x;
run;
filename animout clear;
goptions reset=all;
ods listing close;

```