Visualizing Data with SAS®

Selected Topics

Foreword by
Robert Allison

Table of Contents

Free SAS® e-Books: Selected Topics

In this series, we have carefully curated a collection of topics that introduces and provides context to the various areas of analytics. Topics covered illustrate the power of SAS solutions that are available as tools for data analysis, highlighting a variety of commonly used techniques.

Discover more free SAS e-books!
support.sas.com/freesasebooks

sas.com/books
for additional books and resources.

§sas

THE POWER TO KNOW®

About This Book

What Does This Book Cover?

Creative data exploration and creative problem solving begin with visualizing your data. Data visualization is critical to grasp difficult concepts or identify new patterns that emerge from their data. Our data keeps getting bigger, and we need quicker, easier ways to convey it!

Topics covered in this free e-book illustrate the power of SAS software that are available as tools for data visualization, highlighting a variety of domains, including infographics, geomapping, and clinical graphs for the health and life sciences.

For many more helpful resources, please visit support.sas.com and support.sas.com/books.

We Want to Hear from You

SAS Press books are written *by* SAS Users *for* SAS Users. We welcome your participation in their development and your feedback on SAS Press books that you are using. Please visit https://support.sas.com/publishing to do the following:

- Sign up to review a book
- Request information on how to become a SAS Press author
- Recommend a topic
- Provide feedback on a book

Do you have questions about a SAS Press book that you are reading? Contact the author through saspress@sas.com or https://support.sas.com/author_feedback.

SAS has many resources to help you find answers and expand your knowledge. If you need additional help, see our list of resources: https://support.sas.com/publishing.

Foreword

Sometimes a picture is worth a thousand words, especially when you to need to present complex information in an easily consumable form. In the past, cavemen painted walls, the Egyptians carved hieroglyphs, printers used etching and engraving, and draftsmen created technical drawings–all to visually convey information.

These days, the amount of information has increased exponentially, and luckily we have computers to help us create our pictures–or as we data scientists like to call them, visualizations. *Visualization* encompasses all the graphical techniques that can be used to better understand your data and the results of your analyses. Visualization includes things like graphs and maps, and even a text table with visual cues such as colored traffic lighting.

SAS software provides many different techniques to visualize your data, and several excellent books have been written to demonstrate how to use these techniques. We have selected chapters from SAS Press books and relevant SAS Global Forum papers to introduce you to the topics and let you sample what each of the authors has to offer. If any of these chapters pique your interest, then those will probably be good books to add to your personal library:

1. I-Sah Hsieh and Alice Kassens – "Big Data and the Stories It Can Tell" - *Exploring Big Data with SAS Visual Analytics: Visualizing Trade Data* (Anticipated Fall 2017 – Sign up to be notified when it is available.)

   Big data is everywhere. To many, defining big data is not the important issue. Rather, it is how we use it. Structured and unstructured data can tell stories of fraud, need, and want with the right technologies and analytic tools. In this excerpt, we explore a few examples and scenarios to illustrate the storytelling process.

2. Tricia Aanderud, Ryan Kumpfmiller, and Rob Collum – "The Where of Data" - *An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data*

   Traditional business intelligence systems have focused on answering the *who*, *what*, and *when* questions, but organizations often need to know the *where* of data as well. SAS Visual Analytics makes it easy to plot geospatial data, which can add a completely new element to your data visualizations and analysis.

3. Travis Murphy – "Infographics Powered by SAS Visual Analytics and SAS Office Analytics"

   Infographics are a representation of information in a graphic format that is designed to make the data easily understandable at a glance, without having to have a deep knowledge of the data. Because of the amount of data available today, more infographics are being created to communicate the information and insight from all available data, both in the boardroom and on social media. This excerpt shows you how to create information graphics that can be printed, shared, and dynamically explored with objects and data from SAS Visual Analytics.

4. Xiangxiang Meng, Cheryl LeSaint, and Don Chapman – "SAS Visual Statistics 8.1: The New Self-Service Easy Analytics Experience"

   In today's Business Intelligence world, self-service is a prerequisite because enables an everyday knowledge worker to explore data and personalize business reports without being tech-savvy. The new release of SAS Visual Statistics introduces an HTML5-based, easy-to-use user interface that combines statistical modeling, business reporting, and mobile sharing into a one-stop self-service shop. The backbone analytic server of SAS Visual Statistics is also updated, enabling an end user to analyze data of various sizes in the cloud. This excerpt illustrates this new self-service modeling experience in SAS Visual Statistics using telecom churn data, including the steps of identifying distinct user subgroups using decision tree; building and tuning regression models; designing business reports for customer churn; and sharing the final modeling outcome on a mobile device.

5. Sanjay Matange and Jeanette Bottitta – "Create Your First Graph: Visual Data Exploration with SAS ODS Graphics Designer" – *SAS ODS Graphics Designer by Example: A Visual Guide to Creating Graphs Interactively*

   The SAS ODS Graphics Designer is an interactive drag-and-drop feature that you can use to create many graphs, including histograms, box plots, scatter plot matrices, classification panels, and more. You can render your graph in batch with new data and output the results to any open ODS destination, or view the generated Graph Template Language (GTL) code as a leg-up to GTL programming. The book takes you step-by-step through the features of the designer, providing you with examples of graphs that are commonly used for the analysis of data in the health care, life sciences, and finance industries.

6. Sanjay Matange – "Clinical Graphs Using the SAS 9.4 SGPLOT Procedure" – *Clinical Graphs Using SAS*

   Clinical graphs often display the data in one cell along with derived statistics and other details that aid in the decoding of the information in the graph. Most of these single-cell graphs can be created using the SGPLOT procedure. With SAS 9.4, the SGPLOT procedure supports some new and useful features that simplify the creation of such graphs. The goal of this excerpt is to cover in detail the creation of some commonly used clinical graphs using SAS 9.4. The chapter will provide not only code that you can use directly for such graphs, but also will provide ideas on how you can use or combine plot statements to create your own custom graph.

7. Warren Kuhfeld – "Customizing the Kaplan-Meier Survival Plot"

   Heavily used by pharmaceutical companies and other health care researchers, the Kaplan-Meier plot may be the most important plot in SAS/STAT. In this excerpt, I provide a technical customization method for the Kaplan-Meier plot to display patient survival over time for one or more groups of patients.

We hope these selections give you a better picture of the many tools that are available to visualize your data.

Robert Allison
The Graph Guy

Robert Allison has worked at SAS for over 20 years, and is perhaps the foremost expert in using SAS/GRAPH to create custom graphs. Robert has shared his expertise by presenting papers at several conferences such as SUGI, SAS Global Forum, and SESUG. He also wrote the book *SAS/GRAPH: Beyond the Basics*, and he writes blog posts demonstrating how SAS can be used to visually analyze real-world data.

Robert received his PhD and MS degrees from North Carolina State University. During his PhD program, he used SAS software to build a data warehouse and visualization system for textile and apparel-related data.

Big Data and the Stories It Can Tell

I-Sah Hsieh, Alice Louise Kassens

> Excerpt from upcoming title *Exploring Big Data with SAS Visual Analytics: Visualizing Trade Data*, anticipated Fall 2017. Sign up for our new book notice and receive an email when this book is available for purchase.

"Big data" is an amalgamation of information from a variety of sources. Specific skills and powerful technology are required to properly handle and use big data to its fullest potential. This sample chapter from *Exploring Big Data with SAS Visual Analytics: Visualizing Trade Data* shows the characteristics of big data and provides examples of it from a variety of industries. To contextualize the use of big data in these industries, this sample chapter provides specific examples of businesses harnessing the power of big data to find stories from within and to use those stories to cut costs, save lives, and reduce product delivery time are provided. A drawback to many methods of big data analytics is the steep learning curve. An alternative, visual analytics, is introduced. The remainder of the book explores an application of SAS Visual Analytics to divine stories from the millions of rows of data within the UN Comtrade data set—the SAS Visual Analytics to UN Comtrade.

I-Sah Hsieh is an advisor to the international development community where he uses advanced analytics and innovative technologies to improve responses to disasters and other global challenges. Hsieh has also implemented transformational technologies across highly regulated industries like health care, defense, and telecommunications. He holds an engineering degree from Cornell University.

support.sas.com/hsieh

Alice Louise Kassens, PhD, is the John S. Shannon Professor of Economics at Roanoke College in Salem, Virginia, where she teaches econometrics, labor economics, health economics, and principles of macroeconomics. Kassens incorporates the SAS Visual Analytics UN Comtrade platform into her macroeconomics course. Additionally, Kassens started the first undergraduate SAS Joint Certificate Program at Roanoke College and teaches all courses required for the certificate. Dr. Kassens earned a BA in economics and history at the College of William and Mary, a PhD in economics from North Carolina State University. She taught at Washington and Lee University prior to taking her position at Roanoke College. She also serves on the Virginia Governor's Joint Advisory Board of Economists.

support.sas.com/kassens

Chapter 2: Big Data and the Stories It Can Tell

What Is Big Data?

"Big data" is an increasingly popular phrase, but what is it? What does it mean? To what does it refer? The phrase is both obvious in meaning, yet difficult to define. Just as the words suggest, it describes a large amount of information. This is a vague definition, but in some ways no less clear than other more thoughtful ones. The Oxford English Dictionary, which added the term in its 2013 update, defines "big data" as "data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges" ("Big Data," 2008).

Over time, the definition of "big data" changes as data becomes "bigger" and begets new challenges and opportunities. The first documented use of the term "big data" was in a 1997 paper by NASA scientists who were describing the difficulties of using the enormous amount of information created by supercomputers (Cox and Ellsworth, 1997). Incorporating some of the challenges of working with big data, Laney (2001) defines it as "high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization" (Gartner). This definition is an abridged version of a 2001 research note by Laney (Laney, 2001).

In sum, big data is an amalgamation of information from a variety of sources. Handling the data and using it are both an art and a challenge that require specific skills and technology. Both the data and the tools to gather, store, and analyze it are constantly evolving.

The 3Vs

Volume, *velocity*, and *variety* are three terms, collectively referred to as the 3Vs, used to label the challenges and nature of big data. *Volume* refers to the amount of data collected from many sources including social media, consumer purchases, Internet searches, financial transactions, and machine-to-machine interactions. Currently, an estimated 2.5 quintillion bytes of data is created each day: 2,500,000,000,000,000,000 bytes. That is enough bytes to fill 2.5 trillion novels, which stacked on top of one another would measure 33.3 billion feet. That is the distance from the earth to the moon—times 26! Now that is BIG.

Figure 1. 3Vs
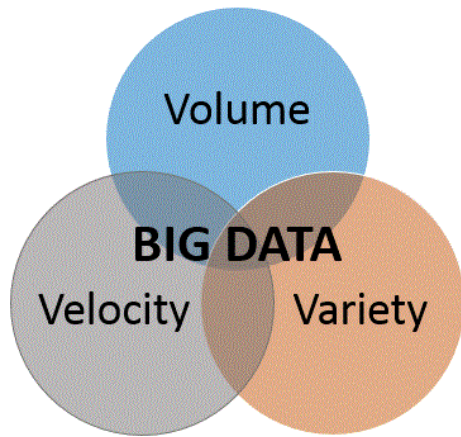
Velocity describes the rate at which data is created, stored, and processed. Data is generated every second. Each social media entry, including video and photos, is data. Every time you use a credit card or loyalty card, information is created, stored, and processed.

Variety references the different forms of data. Data can be structured or unstructured. Most data is unstructured and is not easily searchable. Examples of unstructured data are email, tweets, photographs, videos, and purchase information. Structured data resides in fixed fields and is thus easily searchable and analyzed. Data stored in a spreadsheet is an example of structured data.

Some groups add other dimensions to the big data definition, such as veracity, variability, and complexity (4Vs and a C). Regardless of acronym or number of dimensions, it is clear that big data is a jumble of quickly growing bits and bytes of information.

Examples of Big Data

Google and social media generate a significant amount of data, but let's look at some other examples of big data in more detail.

Big data is generated in many places including the following:
- Retail
- Manufacturing
- Finance
- Health care
- Government
- Education
- Transportation

Retail

Retailers sell goods and services to the public. Walmart Stores, Inc., Kroger Company, Amazon.com Inc., The Home Depot, Inc., and Target Corporation are among the largest retail companies in the world. Consider the data generated, collected, and stored by such establishments: consumer data (from POS systems, website trackers, social media, and call center logs), inventories, prices, and employee information and observations.

Manufacturing

Manufacturers create goods using machinery, typically on a large scale. The top manufacturers in the world include Toyota Motor Corporation, Samsung Electronics Company, Ltd., General Electric, and Apple. Data types that are important to these companies include production quantities, quality-control, and productivity of workers.

Finance

Financial services are those dealing with the management of money including those offered by banks and insurance companies. Berkshire Hathaway, Inc., American Express Company, BNP Paribus, and Banco Santander are some of the largest financial services companies in the world. They consider customer call records, claims records, and inputs for stress tests.

Health Care

The health care industry provides goods and services to treat patients. The industry consists of a variety of markets such as physician, nurse, insurance, hospital, and pharmaceutical. In the private sector, the largest health care companies include McKesson Corporation, UnitedHealth Group, Inc., Express Scripts Holding Company, and Cardinal Health, Inc. Crucial data covers the quality, intensity, quantity, and costs of services and come from both patients and providers and in a variety of forms (structured and unstructured).

Government

Government agencies (local, state, and federal in the United States) purchase and provide a variety of goods and services. In the aggregate, these agencies collect and consider a large amount of data concerning trade, individual income, and public health to name a few.

Education

Public and private educational institutions and firms provide educational services. Data used includes applicant and employee information, financial aid dollars, and operating expenses.

Transportation

Transportation firms provide transportation services for cargo and passengers. United Parcel Service Inc., Delta Air Lines Inc., and Union Pacific Railroad are among the world's largest transportation companies. Essential data for these firms includes passengers, traffic and routes, cargo tonnage, and weather.

Finding the Stories Within

Big data is everywhere. To many, defining big data is not the important issue. Rather, it is how we use it. Structured and unstructured data can tell stories of fraud, need, and want with the right technologies and analytic tools. Let's use a few examples and scenarios to illustrate the storytelling process.

Monitoring the Spread of Disease

Researchers use technology and big data to predict and simulate the spread of disease such as the West Nile Virus and Ebola and assess the impact of public health programs. Telling these stories can save and improve lives and wellbeing. The Centers for Disease Control and Prevention (CDC) is piloting programs in the United States. If successful, officials can make informed decisions to contain and control a potentially catastrophic event.

Wanting to Improve Speed of Delivery

United Parcel Service, Inc. is a leader in the transportation industry, where time is money and losing time is a costly venture. The company has collected data for years with a mind to minimize delivery routes and costs. Recently, UPS took data collection and analysis to a higher level. For example, the company installed telematics sensors in delivery trucks. These sensors record data such as vehicle speed, braking patterns, and productivity. The UPS ORION (On-Road Integrated Optimization and Navigation) initiative aims to improve delivery efficiency. Using mapping and driver data, the system's goal is to re-route drivers in real time. In 2011, it was estimated that the project led to a fuel reduction of 8.4 million gallons by eliminating 85 million miles from daily routes (Davenport and Dyche, 2013). Additional savings is hoped to be realized by restructuring airline routes. Smartly sifting through huge quantities of data is generating big savings for UPS.

Methods of Analysis

In the previous section, we explained the notion of big data and gave examples of both the data itself and the stories that it can tell. How do we get from the data to the storytelling? It is a learned art, much like that so carefully mastered by the likes of Shakespeare and Twain. Telling stories with big data requires special analytic skills and technology and a touch (or more) of creativity. In this section, we will briefly and simply cover popular methods of analysis used today by data scientists like those at UPS. We will assume that you have identified your research question or goal.

Data Preparation

The most time-consuming task of analyzing data is frequently the preparation process. We briefly discuss that process here and then move on to a discussion of analysis methods. First, the data must be gathered. Sometimes the data must be painstakingly collected by the researcher from historical documents, household surveys, or clinical trials. In many cases, the data is collected elsewhere by machines, other humans, and computer programs.

Data comes from a variety of sources, as mentioned earlier, and. Once collected, almost always requires some cleaning prior to analysis. Data cleaning describes the detection and removal of errors and inconsistencies to improve data quality. Problems within a data set might include misspellings, missing observations, entry errors, and other forms of invalid data. To maximize the effectiveness of a particular data set during analysis, steps should be taken to remove these errors. Frequently, this is an arduous task because much of it must be undertaken by the human eye (looking at the data itself or summary statistics) or basic computer programs. The best data-cleaning processes are consistent and treat errors in a uniform way. Data cleaning does not get the glory of data analysis and its results, but is a crucial step toward the accuracy of those results and thus the reputation of the researcher.

Data Integration

Users of big data use data from disparate sources and locations. Examples of sources include email, call logs, social media, and photographs. Examples of locations would be separate businesses. These data sets must be brought together for analysis on a uniform query space. There are several methods of integration ranging from highly manual to virtual and highly automated. For example, a company might have three offices in separate locations, each with their own databases. These databases might contain information pertaining to sales, customer satisfaction, and inventories (databases A, B, and C). In order to more efficiently run their business and increase profitability, the company might decide to bring the three databases together so that anyone within the firm can access and analyze the data as a whole.

Figure 2. Integrating Data into a Warehouse
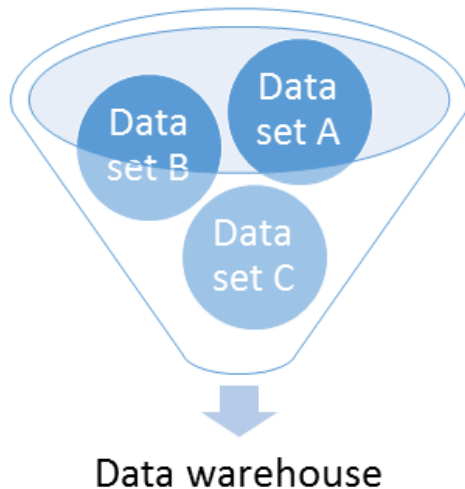
There are several methods of data integration ranging from highly manual with direct access to specific source data to a virtual access point that is separate from the data sources. One popular method is to create a warehouse of the disparate data sources. The warehouse is a unified version of the databases and is managed separately from the originating sources. Users from any of our example firms can access the warehoused data to monitor processes, find trends, test hypotheses, and make predictions.

The following sections briefly discuss these methods of analysis and how we can learn from data. These discussions only scratch the surface of the topics, and interested readers are encouraged to explore these topics in more detail.

Summary Statistics

A simple, but important type of data analysis involves summary statistics of either an entire data set or a randomly drawn subset. Summary statistics provide quick and useful information about the nature and distribution of variables within a data set and can lead to the discovery of data errors. General types of summary statistics are measures of central tendency, dispersion, and relatedness.

Measures of central tendency, such as the mean, median, and mode, indicate the value around which a distribution is centered. The values can be used to describe a typical observation. For example, the sample mean household income provides a measure of the typical income level earned by a group of people. This can be important for a company trying to understand their consumer base.

Measures of dispersion describe the variability of observations from a measure of central tendency. Examples include the variance and standard deviation. It is useful for a firm to know both the earnings of their typical customer and how much observations differ from that measure. If the variability is large in our example, many people earn well below and/or above the mean income value. Targeting the typical household for marketing purposes might not be successful because very few people actually earn that income level.

Frequently, analysts are also interested in relationships between variables. Measures of relatedness, such as the covariance and correlation coefficient, quantify the linear relationship between two variables. For example, a firm might want to know the relationship between sales of a product (for example, boots) and weather. If the relationship is strong, the firm knows to track the weather and adjust inventories accordingly. If there is no relationship, or the variables are independent, there is no need to track weather data (at least as far as sales and inventories are concerned).

Other common summary statistics include minimum and maximum values and measures of range. Not only do summary statistics provide a description of a data set, but they also help us find data errors. For example, if you know that the maximum value of a survey response is five, and you review your summary statistics and note the maximum value reported is a nine, there is a data entry error that should be addressed.

Statistical Modeling

Although summary statistics are simple to generate and useful, they do have drawbacks. One problem is their narrowness. For example, when you look at the relationship between sales and weather, a correlation coefficient only measures the linear relationship between the two variables, and ignores all else. Sales are likely impacted not only by weather, but also by income, consumer sentiment, time of year, and other factors. A more robust measure will tell us the impact of weather on sales while holding everything else constant.

Statistical modeling is a subfield of mathematics in which relationships between inputs are used to predict an outcome. Regression is a common statistical method of analysis that measures the impact of a set of variables or features simultaneously. A dependent variable (for example, each online sale for a particular firm) is regressed on a set of independent variables. Examples of independent variables include income, age, and gender of the buyer, weather at the time of sale, and use of coupon in the transaction. The regression will generate a set of estimated coefficients for each independent variable. These coefficients quantify the additional change in the dependent variable from an additional unit of the independent variable, holding all other independent variables constant. The differences between statistical modeling and other methods (some of which are overviewed in the following sections) are illustrated in Figure 3 (Srivastava, 2015).

Figure 3. Venn Diagram of Data Analysis Methods
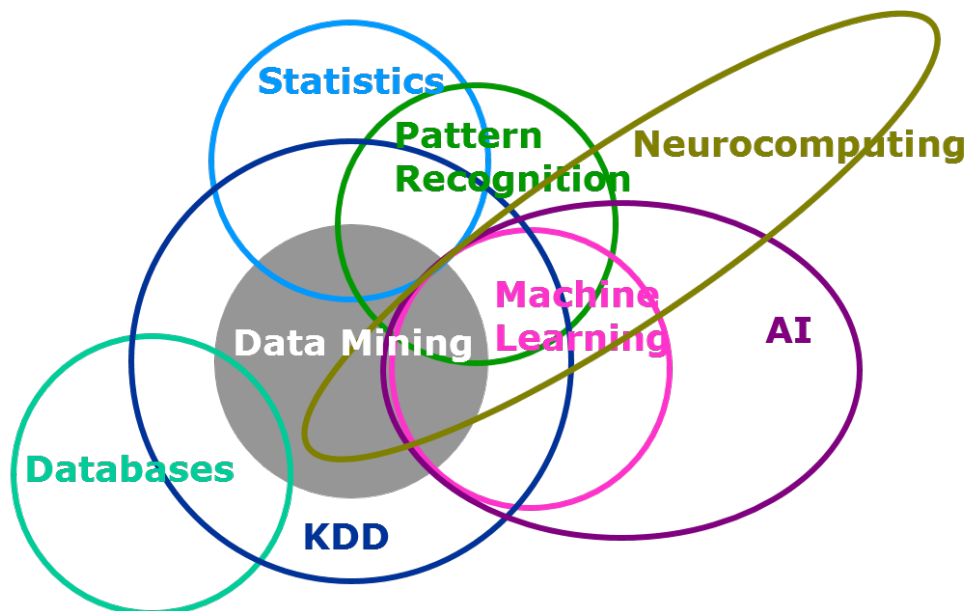
Regression analysis is useful for hypothesis testing (for example, customers buy more boots when it is cold) and for quantifying the relationship between variables while holding others constant (for example, Customers buying 20% more boots when it is cold out than when it is warm out, holding all else constant). In addition to testing hypotheses

and generating estimated values, regression estimates can be used for predictions. For example, given values for all independent variables and estimated coefficients, what is the predicted value of sales? Clearly, statistical models are more advanced and likely more informative than basic summary statistics.

Machine Learning

Machine learning (ML) is a subfield of computer science and artificial intelligence (AI) in which algorithms learn from data. The learned models are then used for predictions. This differs from statistical modeling in which the model is explicitly defined by the researcher. ML became possible with advances in technology and falling prices in that technology. It is widely used with big data. For example, machine learning is telling Pandora, Netflix, and Amazon which song, movie, or product that you might like based on your information, user history, and the choices made by people like you. ML can be supervised (labeled data is used to train the model), unsupervised (data is grouped from unlabeled data), or combinations of the two.

An Alternative: Visual Analytics

Analytic methods such as ML make it possible to divine stories from large amounts of data. A major drawback is the steep learning curve. In order to use these techniques, researchers need a strong background in mathematics and computer programming. Thankfully, this does not mean that while you acquire these skills you cannot glean information from big data. Pictures can show us relationships between variables. "A picture is worth a thousand words" (or numbers) and often requires fewer skills than the techniques discussed in the earlier sections. This is not to say that visual analytics does not require skill, learning, and practice. It does, but is perhaps more attractive to those with an aversion to computer programming and mathematics. Visuals often make more profound statements when presenting your ideas to others, especially with a broad audience.

Visual analytics describes data visualization methods that permit analysts and nontechnical users to visualize data and make decisions; it uses pictures to tell a story. Before developing the visuals, the researcher should be familiar with the data (for example, which variables are numeric); have a specific question or questions to be addressed; and know the audience to whom the results will be presented (SAS, 2014). The simplest visuals to tell the story are the best ones and the best visual can vary from audience to audience.

Visuals range in complexity. Some common, simple graphics include line, bar, and column charts, which are used to show the relationship between two or more variables. Figure 3 shows the percent change in quarterly US Gross Domestic Product (GDP) and the unemployment rate between 2000 and 2016 (Federal Reserve Economic Data, 2016). This chart might be of interest to someone analyzing the relationship between production in the economy and the labor market. It is clear that unemployment is counter-cyclical, meaning it moves counter RGDP (Real Gross Domestic Product) and the business cycle. Added clarity is brought by highlighting periods of recession (in gray). This one picture easily relates the story of recent economic history in the United States. If a researcher presented or analyzed a large table of data, the story would be harder to tell.

Figure 4. US Unemployment Rate and RGDP (Quarterly, SA)
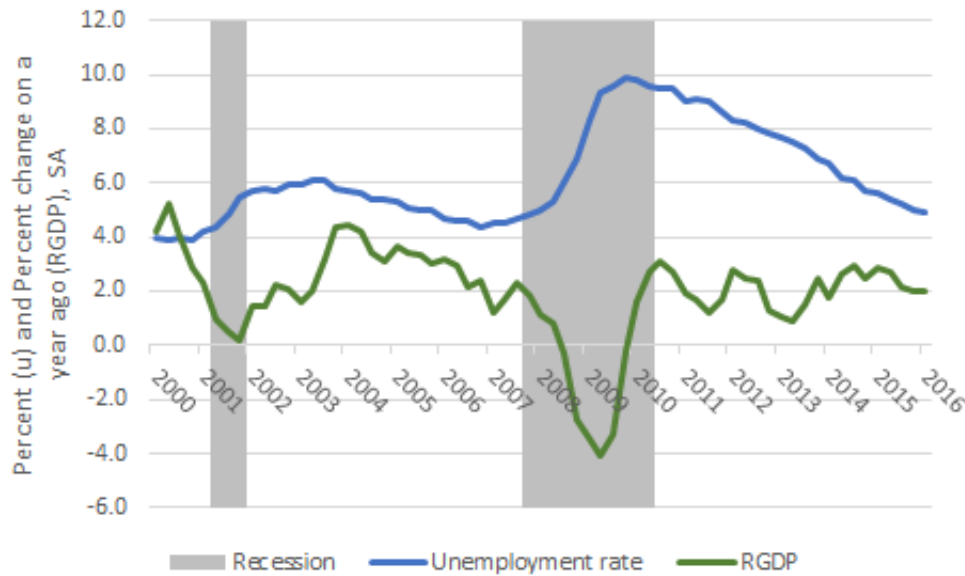
NOTE: Data was downloaded from FRED 6/14/2016. Graphic developed by the author.

UN Comtrade Data and Visual Analytics

The UN Comtrade data set contains over 300 million rows of data and is an example of big data. There are many stories waiting to be told by financial analysts, policy makers, journalists, and more. Finding these stories and relating them is simplified by using visual analytics. In the next chapter, we will detail the data set, including its purpose, uses, and challenges.

References

Big data. 2008. In *OED Online*. Oxford: Oxford University Press. Retrieved June 5, 2016, from http://www.oed.com.

Bureau of Economic Analysis. June 14, 2016. "Real GDP." Federal Reserve Bank of St. Louis.

Cox, M., and D. Ellsworth. 1997. "Application-controlled demand paging for out-of-core visualization." *Proceedings of the 8th Conference on Visualization '97*, pp. 235-ff. Los Alamitos, CA: IEEE Computer Society Press.

Davenport, T. H., and J. Dyche. May 2013. SAS Institute white paper. "Big Data in Big Companies." Retrieved from SAS: http://www.sas.com/content/dam/SAS/en_us/doc/whitepaper2/bigdata-bigcompanies-106461.pdf.

Gartner. No date. *Gartner IT Glossary*. Retrieved from Gartner: http://www.gartner.com/it-glossary/big-data/.

Laney, D. February 6, 2001. "3-D Data Management: Controlling Data Volume, Velocity, and Variety." Retrieved from Gartner Blog Network: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

SAS Institute Inc. 2014. SAS Institute white paper. "Data Visualization Techniques." Retrieved from SAS: http://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/data-visualization-techniques-106006.pdf.

Srivastava, T. July 1, 2015. "Difference between machine learning and statistical learning." Retrieved June 13, 2016, from http://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/.

US Bureau of Labor Statistics. June 14, 2016. "Civilian Unemployment Rate." Federal Reserve Bank of St. Louis. Retrieved June 14, 2016. https://fred.stlouisfed.org/series/A191RL1Q225SBEA

US Federal Reserve Board. July 2014. *2013 Federal Reserve Payments Study*. Retrieved from Federal Reserve Bank Services: https://frbservices.org/files/communications/pdf/general/2013_fed_res_paymt_study_summary_rpt.pdf

The Where of Data

Tricia Aanderud, Rob Collum, and Ryan Kumpfmiller

Excerpt from *An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports and Gain Insight into Your Data*

Maps or geospatial data enable viewers to associate the real concept of land with the abstract concept of data. Exploring geography with your eyes and associating facts with those areas can change a viewer's perspective. It's useful to know where data points occur as well as where they do not. SAS Visual Analytics makes this task easier than ever.

Perhaps you've heard people talk about tornado alley—it's an area down the middle of the United States where tornadoes occur more frequently. Tornadoes are powerful and scary storms that produce wind speeds capable of sending a wood board through a metal car door. These storm events are responsible for massive property damage and loss of lives. In this chapter, the United States National Climatic Data Center storm events (such as tornadoes) since 1950 were plotted. The tornadoes are rated by their intensity with F5 being the most destructive.

What is awesome about SAS Visual Analytics is that users can quickly and easily convert these storm events to a map where they can see tornado touchdowns. With a few more clicks, they can filter the tornados by strength and by year. It's shocking at first to see all the tornados! When you start peeling away the data by filtering for strength, you realize where the events are more likely to occur. Then if you animate the map with time, you realize how rare these events are. Use this chapter to learn about the principles of geospatial data and to add new dimensions to your data.

Tricia Aanderud, Director of the Data Visualization Practice at Zencos Consulting, provides SAS consulting services to organizations that need assistance understanding how to transform their data into meaningful reports and dashboards. She has been a SAS user since 2002, is a frequent speaker at SAS Global Forum and SAS regional user groups, and is the author of two SAS Press books and one self-published book. She has a background in technical communications, process engineering, and customer service. She has a BA in Mass Communications from Eastern Kentucky University.

support.sas.com//aanderud

Rob Collum is a Principal Technical Architect in the Professional Services Division at SAS. For the past twenty years, Rob has enabled the delivery of high-performance solutions that provide substantial value and meaningful impact to SAS customers around the world. He currently works alongside a team of professionals who partner with other divisions to identify, create, and standardize architectural practices for the newest SAS technologies. Rob received a Bachelor's degree in Computer Science from North Carolina State University, is a regular contributor to SAS Global Forum, and has coauthored several SAS certification exams.

support.sas.com/collum

Ryan Kumpfmiller works at Zencos Consulting as a SAS and Data Visualization Consultant where he provides consulting services to companies that need support understanding their data and how to get actionable information from it. He has been a SAS user since 2014 and has presented papers at the SouthEast SAS Users Group (SESUG), as well as at SAS Global Forum conferences. Ryan has a BS in Computer Information Systems and an MS in Competitive Intelligence Systems from Robert Morris University.

support.sas.com/kumpfmiller
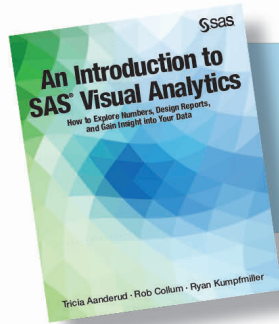
the where of data

Traditional business intelligence systems have focused on answering the *who, what*, and *when* questions, but organizations often need to know the *where* of data as well. Businesses want to plan sales territories based on existing customers. School districts want to understand where students live in relation to the school. Companies want to know whether equipment failures are related to specific locations. All of these issues are related to the *where* of the data.

SAS Visual Analytics makes it easy to plot geospatial data, which can add a completely new element to your data visualizations and analysis. In a tabular report, multiple columns might represent customers, competitors, and demographic information. The tabular report might not reveal anything useful. But if you can geocode the data and overlay it on a map, you quickly see where the better customers are, where they are in relation to competitors, and the regions that provide the most market potential based on underlying demographics.

SAS Visual Analytics geo mapping capabilities are based on integration with two mapping technologies: OpenStreetMap and ESRI ArcGIS. The examples in this chapter use OpenStreetMap with SAS Visual Analytics 7.3 unless otherwise noted. In this chapter, you will learn how to create geographic data items and geospatial objects.

Using geospatial data effectively

SAS Visual Analytics makes plotting geospatial data effortless. This is exciting! It is very easy to just use geospatial data objects for everything. Before traveling that path, though, consider if you really have a location-based data story. You want to use geospatial objects thoughtfully. In this topic, we will discuss an ineffective and an effective use of location analysis.

When location is not part of the data story

In her book, *The Wall Street Guide to Information Graphics*, Dona Wong suggests that there are times when geography is not part of the story, so it doesn't make sense to force it to be. In her example, she shows two sales regions where sales were higher in one. In the following figure, the example was re-created using Australian states.

Figure 6.1 Location is not part of this data story

Because the regions are so disproportionate in size, comparing the sales revenue is not helpful. It does not lead to any conclusion except that Western Australia generated more revenue than Victoria. Your viewer does not have any useful takeaway, because the conclusion might be expected. It doesn't seem relevant to a data story in which the focus was really sales revenue. With only two values to display, would a list table or even a pie chart have been a better data visualization choice? The point of this example is to understand that even if you can show a cool geo object, you should ask yourself if it makes sense for your data story.

When location is the data story

Europe is an international business center and a leading tourist destination. If you want to tell a data story about how popular a location it is, you could start by exploring the airport traffic. The Anna Aero website (http://www.anna.aero) contains data that details trends from most of the world's airports. You can use the airport codes and passenger counts to start a data story about the most and least popular cities.

In the following figure, the location of each airport is shown along with the passenger count and the difference from the previous year. In this example, location enhances the story. Instead of fancy calculations, the viewer can simply use their eyes to search for patterns.

Figure 6.2 Location matters in this story
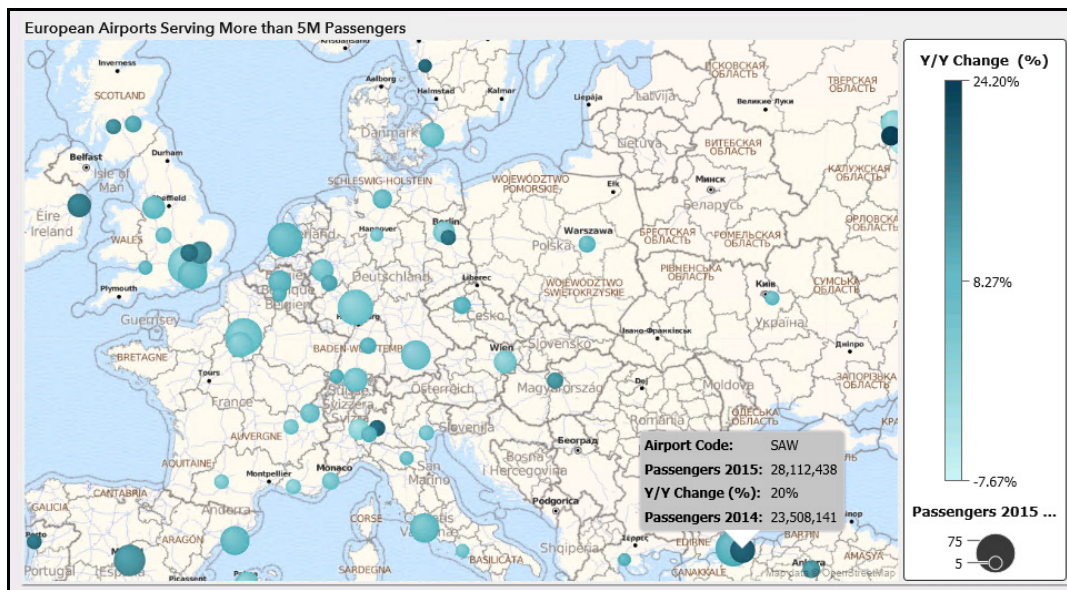
There are multiple observations. You might notice the darker circles that show increased passenger counts. Perhaps you notice where there are multiple airports within a close radius, and you are curious why one has a higher passenger count. When used effectively, geospatial data can reveal previously unknown patterns or assist with confirming suspicions.

Preparing data for geospatial visualizations

Before creating a geospatial visualization, you must have a geographic data item. If a data item contains a location, such as a country or state, then it is considered a geographic data item. Common location examples are customer addresses, store locations, or sales regions. You can use the SAS predefined geographic data elements or create custom geographic data items. This topic describes how to create each type.

Creating a predefined geographic data item

To keep things easy, SAS Visual Analytics has predefined geographic data elements ranging from general values such as country names, to specific values such as ISO country codes. If your geographic data item contains a country name, then it can be matched to an internal table so that the location can be plotted on a map. Starting with SAS Visual Analytics 7.1, geographic data items can be country name, ISO 2-Letter codes, ISO Numeric Codes, or SAS Map ID values. You select the predefined method when you create the geographic data item.

These geographic data elements represent the center of the area. If you are showing France, it can be shown with a country outline or at the center of the country. The following table contains the available predefined geographical data elements provided by default and examples of how the incoming data items are expected to appear. The table shows the data values expected from three countries.

Geographic data element	Examples from Australia	Examples from the Netherlands	Examples from the United States
Country or Region Name	Australia	Netherlands	United States
Country or Region ISO 2-Letter Codes	AU	NL	US
Country or Region ISO Numeric Codes	036	528	840
Country or Region SAS Map ID Values	AU	NL	US
Subdivision (State, Province Names)	Queensland	Noord-Nederland	North Carolina
Subdivision (State, Province) SAS Map ID Values	AU-3	NL-01	US-37
US State Names	—	—	North Carolina
US State Abbreviations	—	—	NC
US ZIP Codes	—	—	27513

On the SAS Support site, there is a Geographic Lookup Values for SAS Visual Analytics (at http://support.sas.com/rnd/datavisualization/vageo/71/VA71LookupValues.html). This page contains a list of these values to help you understand your specific location. The tables at this site list the countries and the associated ISO numeric codes.

SAS Visual Analytics uses internal tables in the MAPSGFK library that is shipped with the product. You can review the tables in this library to ensure that your data matches the expected name by using an application such as SAS Studio. For additional assistance in creating geo data, you can use the GEOCODE procedure that is available with the SAS/GRAPH software.

What is an ISO code?

There is an international standard called ISO 3166 published by the International Organization for Standardization. This standard applies numeric values to countries and regions that everyone can use. There are several advantages to using numeric references, particularly in the data world.

If a programmer is using a non-Latin based language, such as Chinese or Hebrew, the number makes it easy to look up values. Also, when new countries form, a new number can be assigned while maintaining the older number for historical purposes.

Creating a predefined geographic data item

After importing data into SAS Visual Analytics, you must assign the data item to a Geography role before it can be used with any of the geo objects. You can create a geographic data item from an existing character or numeric data item. To create a geographic data item:

1. Right-click the data item that contains the geographic element that matches the predefined role. In this example, the Country data item contains the country names, such as Australia or Brazil.

    Note: Some users prefer to duplicate the data item before assigning it to this role.

2 Select Geography ▶ Country or Region Names. Your data item is moved under the Geography section. You can use it with geographic roles.

3 Choose a geographic role.
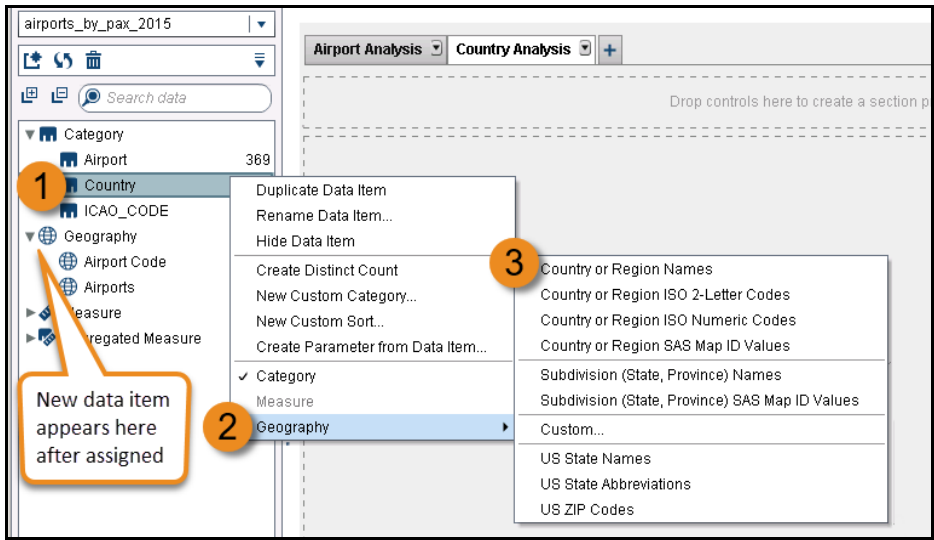
Dealing with location accuracy

If SAS Visual Analytics cannot plot your country data item, you might need to convert a country's common name to its official name. For example, Russia, United States of America, and Great Britain could be in your data set, but SAS Visual Analytics cannot plot them. When you search the country names in the MAPSGFK.WORLD data set, you learn that these countries use a different IDNAME. Often it is easier to convert the values to the ISO numeric code rather than using names.

Figure 6.3 MAPSGFK world data set values

	ID	IDNAME	ISO
1	AE	United Arab Emirates	784
2	GB	United Kingdom	826
3	RU	Russian Federation	643
4	TZ	Tanzania, United Republic of	834
5	US	United States	840

Creating a custom geospatial data item

All geospatial data items represent a location on the planet Earth. A specific point or address has a set of coordinates, which are called latitude and longitude. You might recall from elementary school when you studied the globe and learned how it was divided by imaginary parallel lines that circle the globe from and east to west (called latitude) and from north to south (called longitude).

When you provide a location's latitude and longitude, you are referencing these lines. If you think about the world's airports, it's possible to describe the geospatial location with just latitude and longitude coordinates. In the following figure, the airports are highlighted on the map, and the table on the left shows the airport name with its coordinates. Notice that the latitude numbers are similar because these airports are in a similar eastern European location. There is some variation in the longitude numbers as the airport is further south. Compare the Charles De Gaulle (Paris, France) coordinates to the Dublin (Dublin, Ireland) coordinates to better understand the values.

Figure 6.4 Airports with latitude and longitude
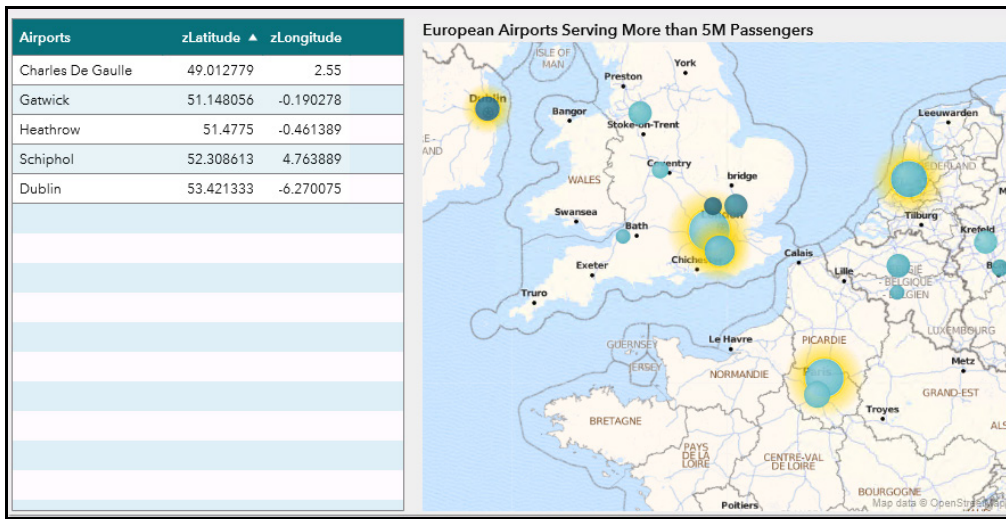
Creating a custom geographic data item

To create a custom geographic item, you must have the latitude and longitude coordinates available in the data set. The coordinates can be based on the World Geodetic System (WGS84), Web Mercator, and the British National Grid (OSGB36). The default is the World Geodetic System (WGS84).

What is a coordinate system?

There are three coordinate systems available for custom data points. These standards were developed for diverse purposes but are now commonly used.

- WGS84 was developed by the United States military for satellite-positioning systems.

- Web Mercator is a web standard. It was first used with Google Maps.

- OSGB36 is a British-developed system that is heavily used in British-based maps.

You should choose the system that works best for your specific location or geo data. In most cases the WGS84 system works.

Let's use the airport coordinates as the basis for the new geographic item. We use the airport name to create this data item, but other data items such as Airport Code would also work.

To create a custom geography data item:

1. Duplicate the Airport data item and name the new item Airport Name.
2. Right-click the new data item, and then select **Geography ▶ Custom**. A Geography window appears.
3. Select your data items for latitude and longitude in the appropriate fields. Your new data item appears in the Geography area.

Figure 6.5 Adding a custom data point
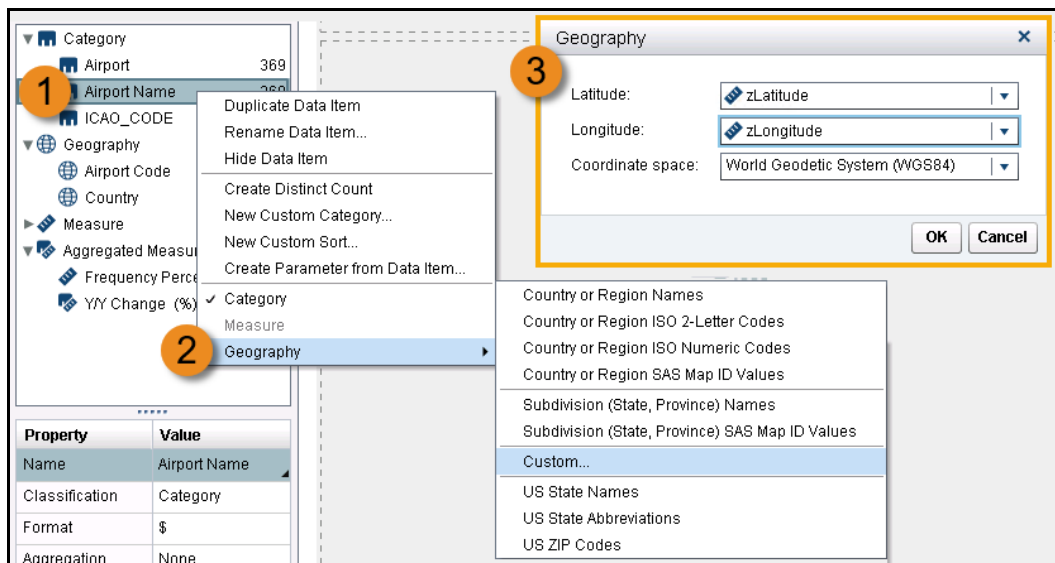
Finding geo coordinates

If your data set does not have the geo coordinates available, you can get them through several sources.

- The SAS MAPSGFK library contains multiple countries and regions.

- There are open-source databases available that you can find with a web search.

- Google has an API that you can query through code. The free service has a daily access limit, but you can subscribe to their service or other commercial services.

Displaying geospatial objects

There are three ways to display geo data in SAS Visual Analytics:

- Coordinate – Pinpoints an exact location on the map using a custom geographical item

- Regional – Outlines a regional area

- Bubble – Combines a bubble plot to show a value at the location

These objects enable you to highlight your geospatial data in different ways and for different stories. Let's explore the different ways that these data objects are used.

For the remaining topics in this chapter, the examples are created using the storm events data set from the United States National Climatic Data Center website. This database contains US storm events (such as tornadoes and thunderstorms) since 1950. The data set contains other facts such as the number of deaths or injuries and the estimated property damage. The tornadoes are rated by their intensity on the Fujita scale from F0 to F5 with F5 being the most destructive. In 2007, the Enhanced Fujita scale was introduced and tornadoes were categorized as EF0-EF5.

Get to the point with geo coordinate data objects

Perhaps you've heard people talk about tornado alley—it's an area down the middle of the United States where tornadoes occur more frequently. Tornadoes are powerful and scary storms that produce wind speeds capable of sending a wood board through a metal car door. These storm events are responsible for massive property damage and loss of lives. Geo coordinate maps are excellent at showing exactly where an event occurred. In the following figure, the teal markers indicate where tornadoes with EF5/F5 strength of 230 mph+ (370 kph) winds arose in the past 50 years.

Figure 6.6 F5/EF5 tornado locations
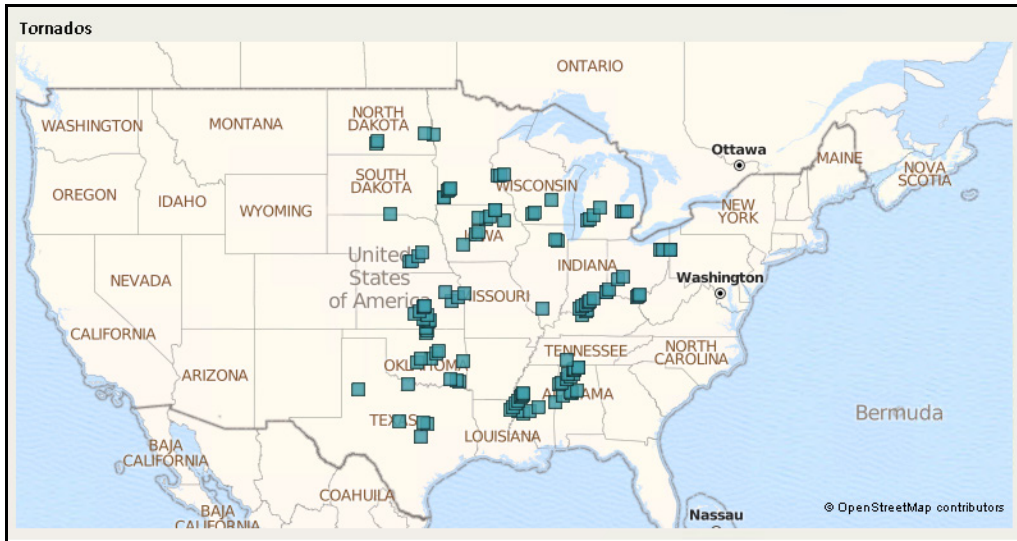
To make this geo coordinate object a little more interesting, let's add the EF4/F4 tornado touchdown points for the same time period. By contrasting the teal and gold markers, the viewer sees that an EF5/F5 tornado is less common.
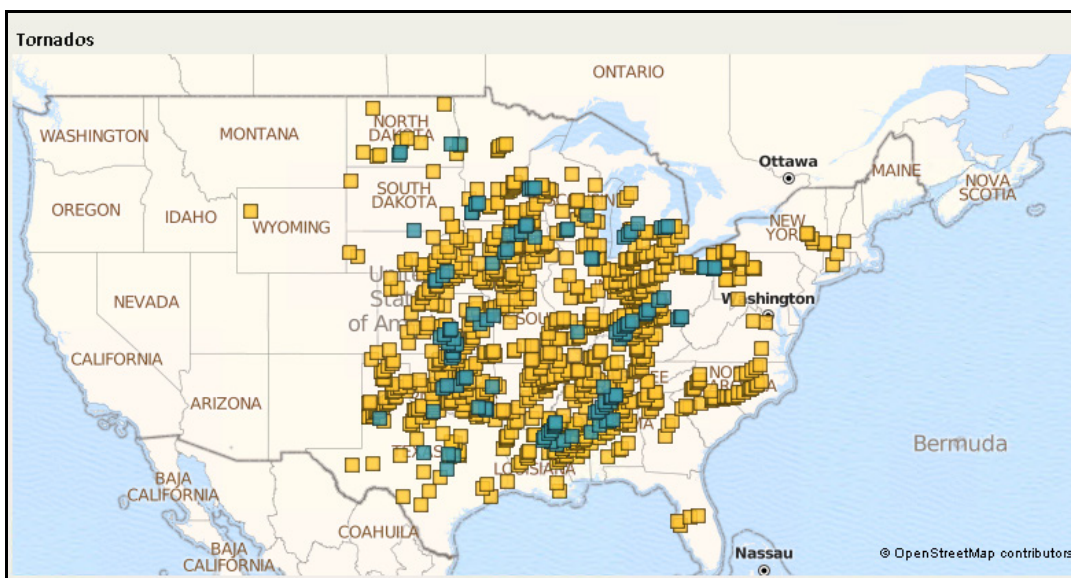
While the data points are chaotic, it's clear where severe tornados occurred. This data would not have had the same impression if we had plotted it as a line chart or even a pie chart. The touchdown points help you realize why those particular states have a higher disaster recovery budget.

Tip 1: Dealing with odd locations

This data object uses a custom geography data item that is based on supplied latitude and longitude values. If the coordinates are incorrect, then the map might show your data in the middle of the ocean. In our sample data, some of the coordinates were entered incorrectly. This resulted in tornados appearing in the Atlantic Ocean. To correct this situation, the latitude and longitude values in the data set would have to be edited or filtered.

Figure 6.7 Tornados in the ocean
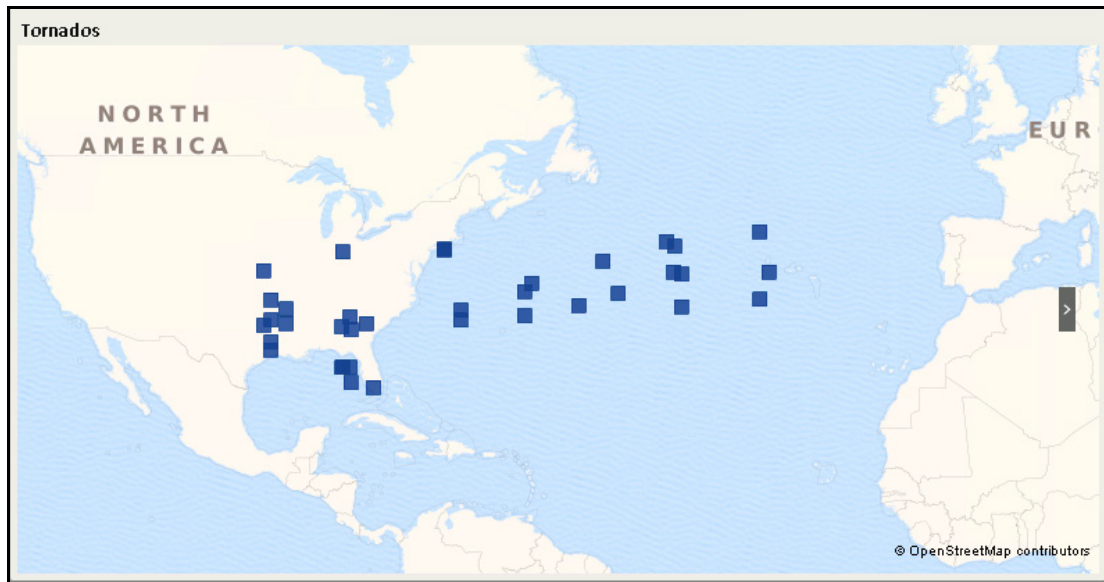
Tip 2: Controlling the data

When you have too much data to display, SAS Visual Analytics issues a yellow icon and warns you to add some filters to your data. With custom geographic data items, it is more likely to happen. The solution is to control how much data appears at once by setting filters.

Figure 6.8 There is too much data at one time!
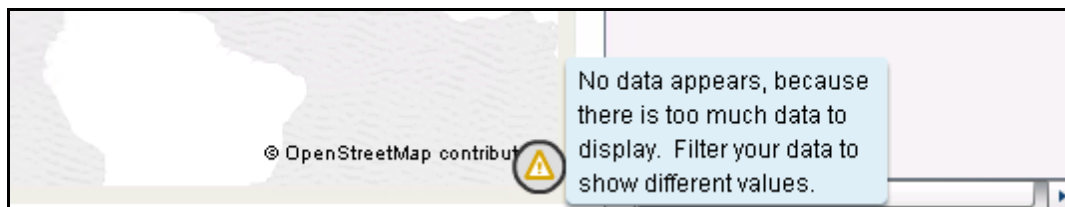
Here are a few suggested filters:

- Add a date range slider to compare events along a time scale. Adding Event Year to the slider enables the user to compare which years might have had more active storm seasons.

- Split the data item categories. Use the display rules to assign the tornado scale to a different color so that each level is clearer. Then add a List filter and assign the Tornado F/EF Scale to the list. Users can select which tornado scale they want to compare.

Figure 6.9 Add filters to keep data visualization manageable
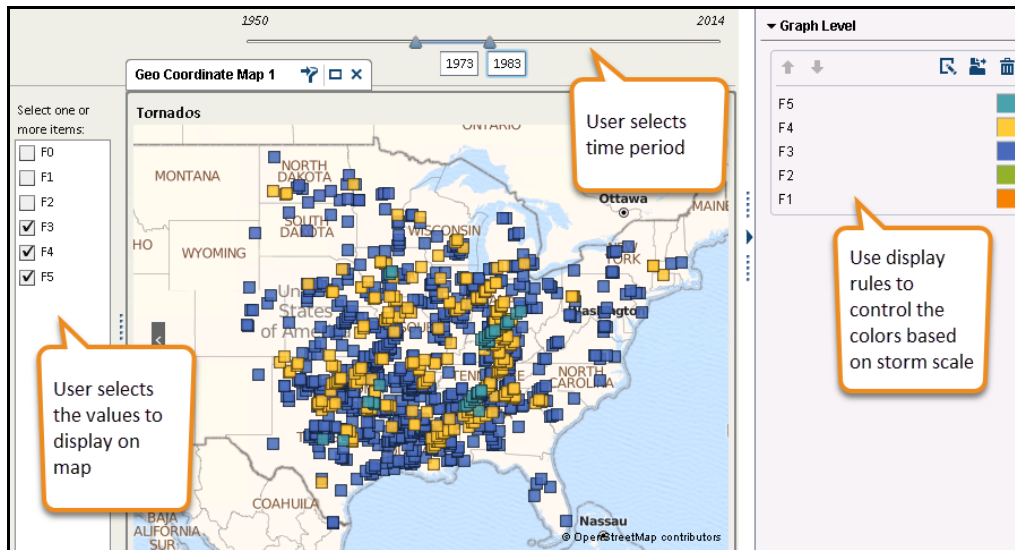
Compare area with geo regional data objects

Use the geo regional data object when you need to introduce a subject about location. This geospatial object helps a viewer understand where to focus their attention or understand how much variation occurs for a value. These objects are also called *choropleth* maps, which is Greek for multitude of areas.

When you start thinking about dangerous storm events, you can imagine that these events cause considerable property damage. States more prone to severe tornados will plan larger disaster recovery budgets. It would be interesting to compare the damage costs by state. Using a geo regional map, you can place a value over an entire region, such as a country or a state. Color is then applied over the regions to indicate the intensity of the value.

Figure 6.10 Understanding regional events
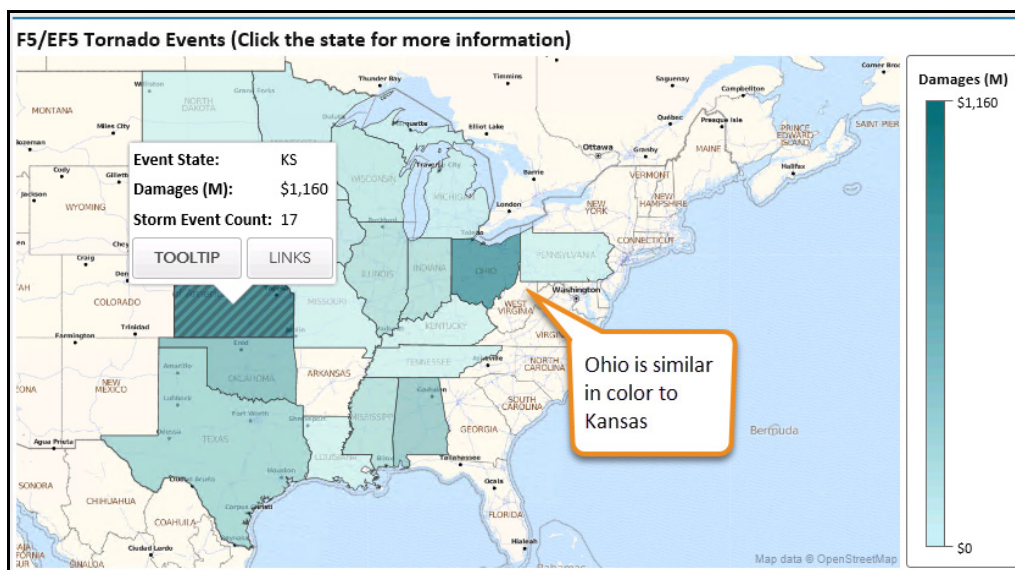
In the preceding figure, you can see the associated property damage cost for the tornadoes across the areas. The darker the color, the costlier the storm damage. Use an average or percentage to make the values comparable or normalized. By exploring the visualization, you can easily see the areas of most damage, but it's harder to understand where there is the least damage. Be sure to use a legend so that the user understands the color range.

When you position your pointer over each state, a data tip appears that contains the assigned data items values. Since Ohio and Kansas are similar in color, viewers might be interested to learn more. Most of Kansas is farm land and rural areas, while Ohio is more densely populated and industrial. Being in tornado alley, Kansas probably experiences more tornados and thus more crop damage. With the larger population, it might be costlier for Ohio when there is an extreme storm event.

Tip 1: Improving your geo regional map

There are a few settings that can make a geo regional data object a nicer user experience.

- Add data tips to provide more information when the user positions the pointer over content.

  You can add as many as you like, but make sure that the data items enhance instead of confuse the viewer. For the preceding example, we added the Storm Event Count as a data tip.

- Adjust the color transparency for the overlay so that the user can see the underlying values.

  If the underlying values are masked, it might cause confusion. For this visualization, the transparency was adjusted to 25%. It was just enough to maintain the color while still allowing the underlying value to peek though.

- Adjust the gradient color to ensure enough contrast.

  The ocean is a light blue, so a contrasting color that does not appear too similar to the landscape features is required. In Figure 6.9, a single color for the Gradient value is used. It is easier to decode a value when the color intensity increases as the value increases.

> ### Choose lighter colors
>
> In *Envisioning Information*, Edward Tufte has a fascinating discussion of color with maps. His suggestion is to use colors that are found in nature. He encourages using a color palette on the lighter side and provides several examples used across several centuries.

Tip 2: Adding rich details for exploration

The geo region data object is excellent for getting the user to focus on specific areas. It leads to more questions about the storm events, so it might be convenient to use an info window to provide more details. This info window shows the storms by duration with estimated damage. A quick storm can result in as much damage as a longer one, although this probably depends on where the tornado touches down.

Figure 6.11 Use a pop-up window to provide more details
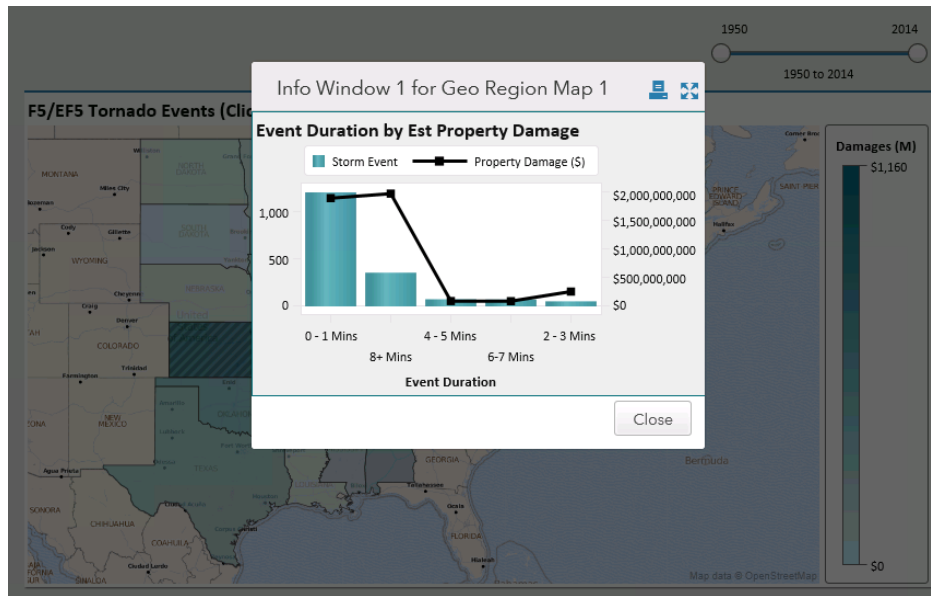
When the user clicks on the state, the info window appears with additional information. This example uses a bar-line chart, but it can be anything you can create in a section. This map is a good way to start a story. It provides an overview and helps the viewer understand where to focus their attention. In this case, it was Kansas and Ohio.

The only pitfall to an info window is that the viewer might not recall the values from the previous pop-up. Use this technique for data discovery or as a way to entice someone into your story. This data story is completely about the location and comparing how the events affected the states.

Adding an info window to your map

Info windows are pages that you can link to from another page. Use the following steps to add an information window to your map.

1. Create a tab. In this example, the geo regional map was created.
2. Create another tab with the data objects of your choice. For this example, a bar-line data object was used to show the event duration and estimated property damage.
3. Select the down arrow next to the title and select **Display as Info Window**.
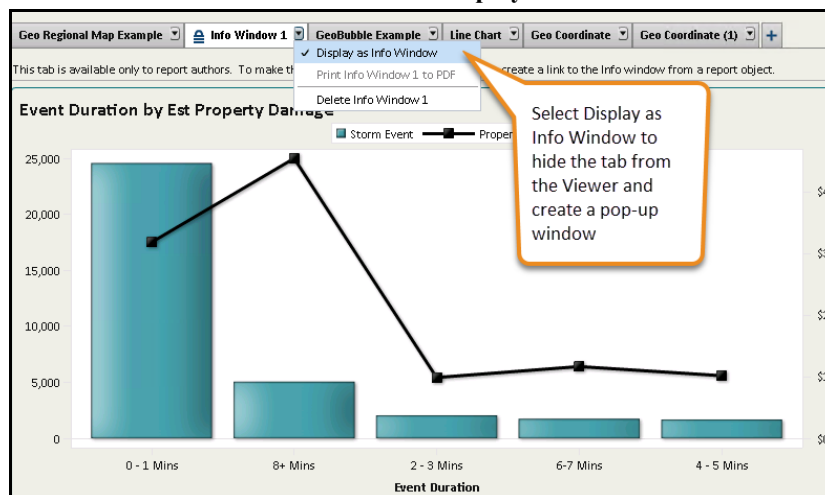
4. Return to the page that you created in step 1. Right-click on the map, and then select Add Link > Info Window Link. In the window that appears, select which info window you want.
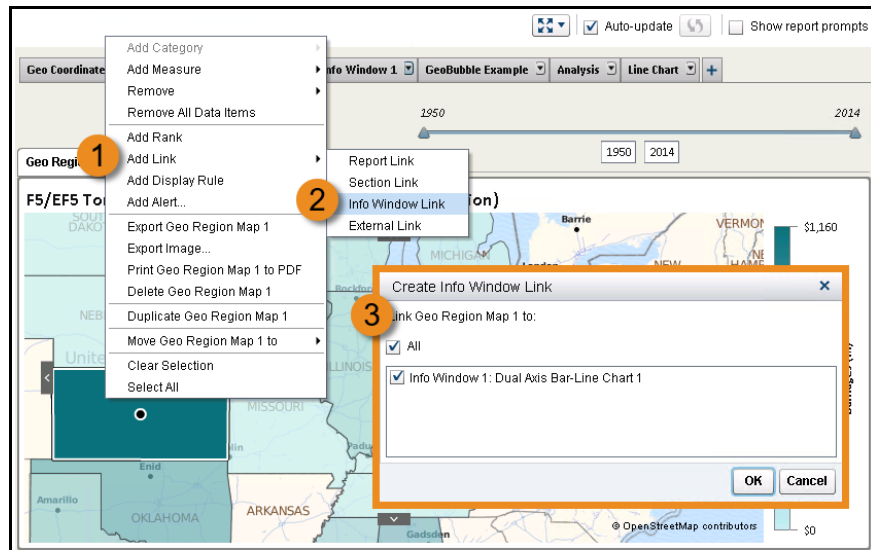
Once you turn the tab into an info window, it does not appear to the viewer. You can use an info window in other situations to provide information about the tab.

Show overall trends with bubble plots data objects

Bubble plots receive a lot criticism for being difficult to understand. These charts can pack a lot of data into a few variables. A layperson might spend more time trying to understand a bubble plot, but this doesn't seem true for the geo bubble maps. Possibly it's because the user sees the map and understands that it is related to location.

In the previous topic, we created a geo regional map to show the average damage cost from F5 tornadoes for each state. One issue with the method was that users had to position the pointer over each state to see how many storm events were associated with each event. If a user wants details, it is a little awkward. A geo bubble map resolves this issue..

A geo bubble plot places a bubble on the geographic location and enables you to control two aspects of the bubble: its size and color. In the following example, the bubble size is the event count (the number of tornadoes) while the color shows the estimated property damages (shown with the scale). Now it is more apparent that Kansas endured a similar number of events as Mississippi, but the price tag was a little larger. However, it also shows that Ohio had a similar cost but fewer events than Kansas.
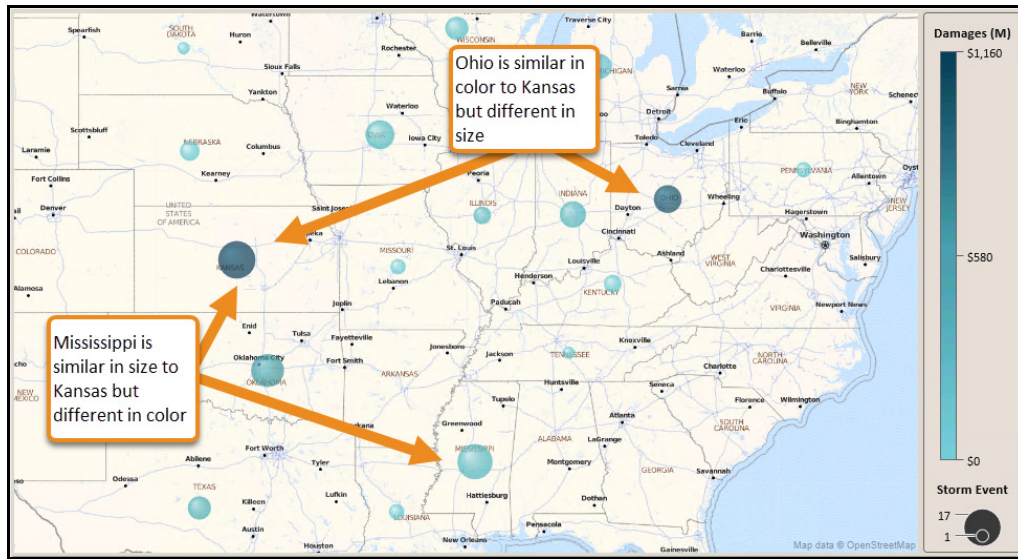
Tip 1: Ensure that the legend is visible

When you use bubbles to encode data, you are asking the user to compare the bubble size and the bubble color. The legend ensures that the user has some visual cues to assist with understanding. You can place the legend anywhere around the object. In the preceding example, the legend is placed on the right.

Tip 2: Watch the default colors

By default, the geo bubble object uses a gradient scale of red to blue. This scale is acceptable when working with performance data and is commonly referred to as trafficlighting. The color mimics the traffic signals where red means stop and green means go. However, we have a logic problem in this instance. The red indicates the least amount of damage and the blue indicates the most. Technically, any property damage is bad. (After all, we are not measuring how well the tornado was at damaging property!)

The gradient scale was changed to teal in our chart. The bubbles are not as close to the ocean and provide enough contrast with the circle. However, notice that the bubble over Tennessee is barely visible. The bubbles were set to 30% transparency to make the state names visible. Perhaps another color would be more suitable? You can experiment with your data object and decide.

Expanding location intelligence

Starting with SAS Visual Analytics 8.1, users have unlimited access to the ESRI base maps from within SAS Visual Analytics. This provides geo search functionality and ad hoc selection of data points on a map. Here's an example of a geo search where the user was looking for how close Los Angeles, CA customers were to the retail chain Sports Authority.
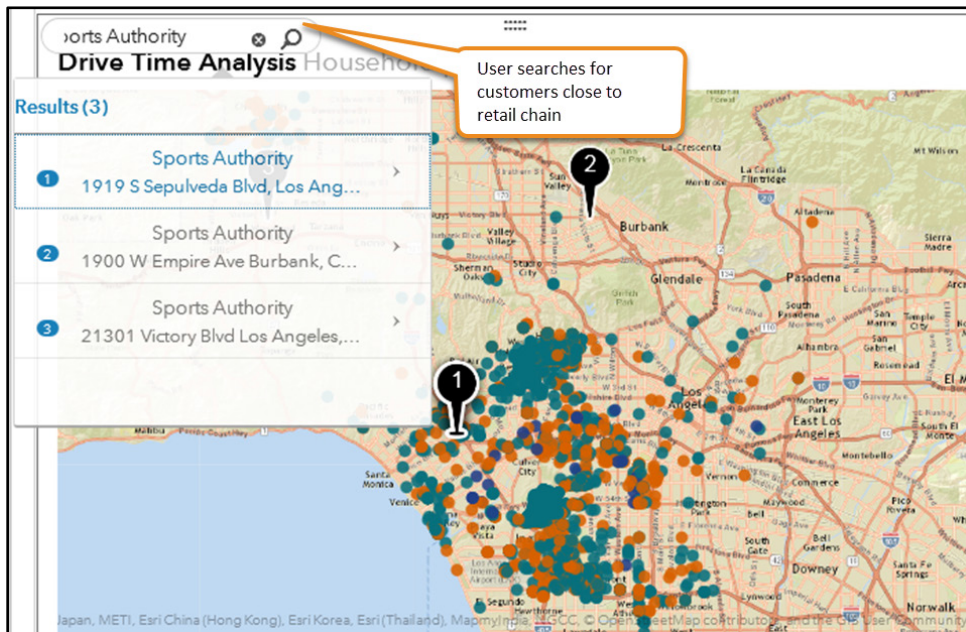
For users who want additional functionality, they can subscribe to the ESRI premium features. The premium service offers drive-time analysis, drive-by-distance analysis, and the ability to create custom shapes. In this example, the user was looking for customers within a 5- to 10-minute driving distance of the store location. The darker inner area is the 5-minute drive, while the lighter inner band in the 10-minute drive.
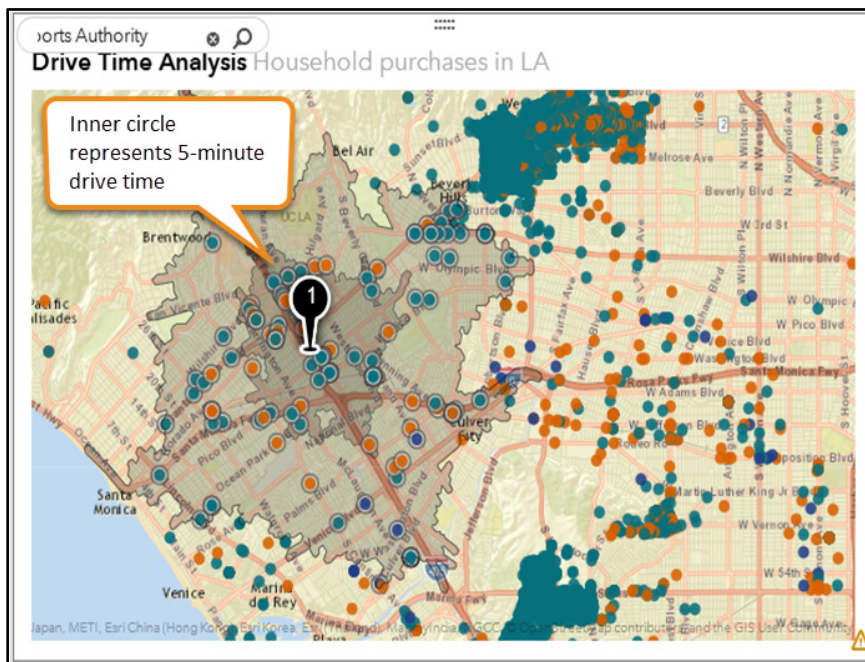
Understanding details about mapping technologies

SAS Visual Analytics geo mapping capabilities are based on integration with two mapping technologies: OpenStreetMap and ESRI ArcGIS. SAS Visual Analytics enables its users to view their enterprise data mapped across the various locations on the map.

OpenStreetMap This is an open-source project, where a worldwide user community maintains the data about roads, boundaries, trails, and much more.

ESRI ArcGIS Maps This advanced mapping platform uses highly interactive and informative geographical maps. The maps are maintained by ESRI, a SAS partner.

The SAS Visual Analytics environment must be configured to point to one of these mapping technologies. An OpenStreetMap server is hosted by SAS and is available as part of the default configuration. Organizations might want to host and maintain their own OpenStreetMap server. Organizations can also use the ESRI server (ArcGIS for Server, version 10.1 or higher) for access to maps. Refer to the *SAS Visual Analytics: Administration Guide* for your release for more configuration details.

Many SAS Visual Analytics users are concerned about what information from their data must be shared in order to retrieve map tiles from OpenStreetMap or ESRI ArcGIS Maps. After all, if the data is confidential to their enterprise, it needs to be kept secure. Fortunately, none of your actual data is leaked outside of the environment. SAS Visual Analytics simply requests the specific map tiles necessary to render the selected geographic area. The highlighted regions, bubble plots, and all are created within the SAS Visual Analytics application.

References

Aanderud, Tricia. 2016. "Where in the World is SAS Visual Analytics?" Available at
https://www.zencos.com/blog/review-geoplot-in-sas-visual-analytics/.

Massengill, Darrell. 2016. "The GEOCODE Procedure and SAS Visual Analytics." *Proceedings of the SAS Global Forum 2016 Conference*. Paper SAS3480-2016. Cary, NC: SAS Institute Inc. Available at
http://support.sas.com/resources/papers/proceedings16/SAS3480-2016.pdf.

Nori, Murali, and Himesh Patel. 2016. "Location, Location, Location—Analytics with SAS Visual Analytics and ESRI." *Proceedings of the SAS Global Forum 2016 Conference*. Paper SAS4060-2016. Cary, NC: SAS Institute Inc. Available at
http://support.sas.com/resources/papers/proceedings16/SAS4060-2016.pdf.

Schulz, Falko, and Anand Chitale. 2014. "More Than a Map: Location Intelligence with SAS Visual Analytics." *Proceedings of the SAS Global Forum 2014 Conference*. Paper SAS021-2014. Cary, NC: SAS Institute Inc. Available at
http://support.sas.com/resources/papers/proceedings14/SAS021-2014.pdf.

Tufte, Edward R. 1990. *Envisioning Information*. Cheshire, CT: Graphics Press.

US NOAA Data. Storm Events Database. Accessed 2016. See https://www.ncdc.noaa.gov/stormevents/.

Wong, Dona M. 2013. *The Wall Street Journal Guide to Information Graphics: the Dos and Don'ts of Presenting Data, Facts, and Figures*. New York, NY: W. W. Norton & Company, Inc.

Infographics Powered by SAS Visual Analytics and SAS Office Analytics

Travis Murphy

> This is an excerpt from a 2016 SAS Global Users Group *Proceeding*. For more SUGI and SAS Global Forum *Proceedings*, visit the online versions of the Proceedings.

A picture is worth a thousand words, but what if there are a billion words?

The old saying is very true today given the rise in data and the appetite for this data in all aspects of life. The picture becomes even more important today, and this is where infographics step in.

Infographics are representations of information in a graphic format designed to make the data easily understandable, at a glance, without your needing a deep knowledge of the data. The amount of data available today is vast, and more infographics are being created to communicate the information and insight from all this available data. This is not just on social media–it is inside the boardroom as well. This paper shows you how to create information graphics that can be printed, shared, and dynamically explored with objects and data from SAS Visual Analytics.

Connect your infographics to the high-performance analytical engine from SAS for repeatability, scale, and performance on big data. In this paper, you will see how to easily leverage elements of your corporate dashboards and self-service analytics while communicating subjective information and adding the context that business teams require, in a highly visual format.

This paper looks at how SAS Office Analytics enables a Microsoft Office user to create infographics for all occasions. You will be presented with a workflow that lets you get the most from your SAS Visual Analytics system without having to code anything. This paper provides the perfect blend of creative freedom and data governance that comes from leveraging the power of SAS Visual Analytics and the familiarity of Microsoft Office.

Travis Murphy is a strong advocate and evangelist around self-service analytics and data visualization. His career has been focused on helping organizations, in both the public and private sectors, to get the most from their investment in business analytics software. Travis has worked on many data warehousing projects, directly with the customer and for great software vendors, like SAS, to assist in delivering on the promise of self-service analytics. He loves technology and how it changes the world around us, and tries to pass his energy to people around the world. Travis has presented at conferences, like SAS Global Forum, and continues to share interesting and reusable use cases for SAS software.

support.sas.com/murphy

Infographics Powered by SAS® Visual Analytics and SAS® Office Analytics

Travis Murphy, SAS Institute Inc.

Excerpt from SAS Global Forum 2016 *Proceedings*

ABSTRACT

A picture is worth a thousand words, but what if there are a billion words? This is where the picture becomes even more important, and this is where infographics step in. Infographics are a representation of information in a graphic format designed to make the data easily understandable, at a glance, without having to have a deep knowledge of the data. Because of the amount of data available today, more infographics are being created to communicate the information and insight from all available data, both in the boardroom and on social media. This session shows you how to create information graphics that can be printed, shared, and dynamically explored with objects and data from SAS® Visual Analytics. Connect your infographics to the high-performance analytical engine from SAS® for repeatability, scale, and performance on big data and for ease of use. You see how to leverage elements of your corporate dashboards and self-service analytics while communicating subjective information and adding the context that business teams require, in a highly visual format. This session looks at how SAS® Office Analytics enables a Microsoft Office user to create infographics for all occasions. You learn a workflow that lets you get the most from your SAS Visual Analytics system without having to code anything. You will leave this session with the perfect blend of creative freedom and data governance that comes from leveraging the power of SAS Visual Analytics and the familiarity of Microsoft Office.

INTENDED AUDIENCE

This paper is aimed at SAS Visual Analytics users who create and design reports and dashboards for their users. Managers can use this paper to determine what the teams can create and design with SAS Visual Analytics and SAS® Office Analytics. This paper is for beginner and intermediate SAS users.

INTRODUCTION

The world is at a point where the attention span of a consumer is only about eight seconds. If something doesn't grab their attention, chances are they will move on. Not only will they move on, but most of the time they won't come back. This has created a shift in the massive growth of using infographics to quickly show data-driven visuals for immediate impact. There are now infographics about how many infographics there are. There is a reason for this growth in infographics.

> *"Now you can circumvent written language to a large extent. A lot of printed words are there to describe things that occur spatially. In many cases a picture is worth a thousand words. Now we can generate these pictures and graphics and we can convey them to other people very easily. I think it's inevitable that visual media are going to become more important in conveying ideas and not just about raging fires."*
>
> **Marcel Just, Center for Cognitive Brain Imaging at Carnegie Mellon University, 2010**

As the quote infers, data visualization is a more effective way of getting a reaction from the audience. Analysts need to adapt to this shift in the audience's attention span and combine visuals with the massive amount of available data. We need to combine infographics with big data to take advantage of the

opportunities that it presents. There are two types of infographics in broad categories: artistic and business. The artistic infographic is one in which graphic designers take information from authors and create a very artistic visual. This visual could be hung on the wall as a poster or put on the front page to support a headline. The next is the business infographic. This is a very structured visual and an extension of the dashboard concept. A dashboard can be an infographic as well. However, generally they are more about data than about being subjective, which often makes them a one-stop shop for self-service business intelligence. Both styles of infographics have their place, and when used correctly, they can have a great impact on the intended audience. The business infographic, which is the focus of this paper, combines artistic elements like clip-art background images, and includes data-driven content from the corporate data warehouse or big data platform. Traditionally, the creator needed to spend a large amount of time with tools such as Microsoft Excel or with programming languages to develop and craft the correct data to support the right visual. This is not so any more.

Analysts do not have to write code to get great visualization from the data. They don't have to spend hours using scripts and spreadsheets to craft the data into a usable format. The analyst can start to be more creative on the visual layer and get better visualization in the infographic. This paper is not focused on the theory of design for infographics. I will leave that to the graphic designers, data journalists, and visual artists of the world. This paper focuses on the SAS analytic engine and associated software to make extremely powerful tools available to the business analyst to better create infographics.

The boardroom, just like the classroom, has a much shorter attention span to consume information and to get to the "AHA" moment as fast as possible. The emergence and growth of many data visualization tools have placed infographics and data visualization at the forefront when considering business intelligence solutions. SAS has been a leader in data visualization for 40 years. Over the years, the tools have improved to be more approachable for more people within an organization. It is proven that data presented visually is more easily processed by the brain than looking at the tabular format or words alone. Also, the number of infographics has increased dramatically over the past five years, and according to Google trends, this number continues to rise. The only caveat is that pictures and visualizations themselves should not impede the message or the clarity of the information being communicated. I know from personal experience, working on many data warehouse projects, that all stakeholders, whether executive or line manager, need to be guided on a path, a repeatable path, to the insight that is being communicated from the data platform.

What has changed is the vast increase in data: the volume, variety, and velocity. The Internet of things and surrounding applications are guaranteeing the continued growth in data assets. The importance of data visualization will be the difference between noticing a pattern or missing a pattern altogether. To add to this, business analysts need to do what they did in multiple pages in a dashboard now in eight seconds to capture the audience. Business analysts must provide insight and create a reason for the audience to click through to more detailed information. This is where the business infographic comes in.

ARTISTIC FREEDOM VERSUS COMMUNICATION

You need to innovate by communicating information from the increasing data in the world. This is often done at the expense of clarity. Creative visuals are great. But, they should not get in the way of context or narrative that needs to exist to provide insight at the fastest possible speed: *insight in an instance*. If the user or audience needs to study the visual and look deep into it for insight, then this delays the action or decision that can be reached from the insight. Actionable insight is key from any data-driven communication.

A balanced approach is key when communicating with data. The famous mathematician John Tukey once said, "The greatest value of a picture is when it forces us to notice what we never expected to see" (Tukey 1977). This quotation has been a constant test that I have applied to my work over the years. This concept is not new, and it is not just about how your graphic looks artistically. In fact, the story or insight

the visualization brings to us is much more important. There is a need to balance the freestyle approach of personal productivity tools like web-based infographic tools, graphic design tools, or office productivity tools. The tipping point for me is empowering a business analyst or citizen data scientist to use the analytics tools that they have available, while being open and approachable and not requiring them to code or learn a totally new product.

To achieve this, you can use the enterprise strength of the SAS Visual Analytics engine with the popular personal productivity tools of Microsoft Office. This leverages the SAS offerings of SAS Office Analytics, which includes SAS® Add-In for Microsoft Office. Check the required software at the end of the paper for more information, including version numbers of the software used in this paper.

WHAT ARE WE TALKING ABOUT?

The best way to set the scene for what is being proposed is to look at the final output first. The following image shows what is being created in this paper:
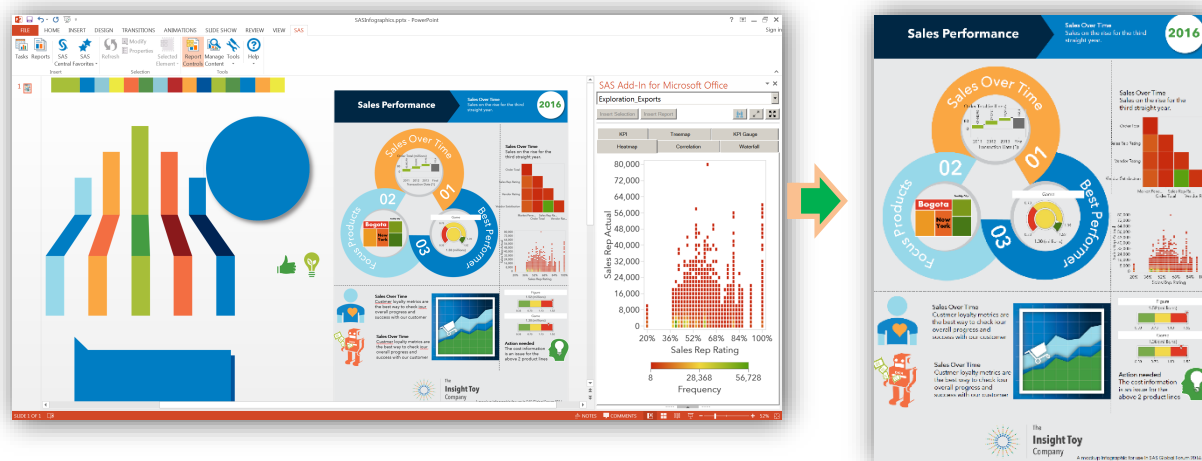
Figure 1. Infographic in PowerPoint and Excel

WHY ARE INFOGRAPHICS SO POWERFUL?

Infographics create the combination of art and science: not unlike a dashboard. The big difference is the subjective data and the narrative or story being told, which enhances the data-driven visual and engages the audience quickly through highlighting the required context.

There is a place for the artistic infographics that we have come to love. Infographics can be used with your day-to-day business analytics information that you create and distribute at work. This is another way to engage your stakeholders.

WHAT ROLE DOES SAS PLAY IN THIS APPROACH?

The infographic in Figure 1 contains visual objects driven from the trusted SAS analytics engine. This provides a more consistent infographic for the boardroom because trusted data sources are used to drive the visual elements. Business analysts can focus on the narrative and the story that they want to tell, rather than spending all their time doing data preparation tasks.

Here are a few key benefits from using this approach:

- **Repeatable**–You can refresh each of these visuals each week or month without any other rework. This is easy to repeat, not having to start from the beginning every time.

- **Scalable**–You need to scale to massive amounts of data, allowing the compute power of the SAS analytic engine to be leveraged underneath what seems to be a simple summary or individual gauge.

- **Approachable**-There is no code required. This approach leverages skills that an everyday user would have already, including using Microsoft Office, the SAS drag-and-drop user interface, and SAS Visual Analytics.

It is important to understand what technology is being used to achieve this unified approach: the SAS® Enterprise Analytics platform, office productivity tools, and your own design creativity.

OVERVIEW OF THE SAS SOLUTIONS

SAS VISUAL ANALYTICS

SAS Visual Analytics is one tool for business intelligence that covers all use cases. Many business intelligence vendors specialize in one of the following mindsets; SAS specializes in both.
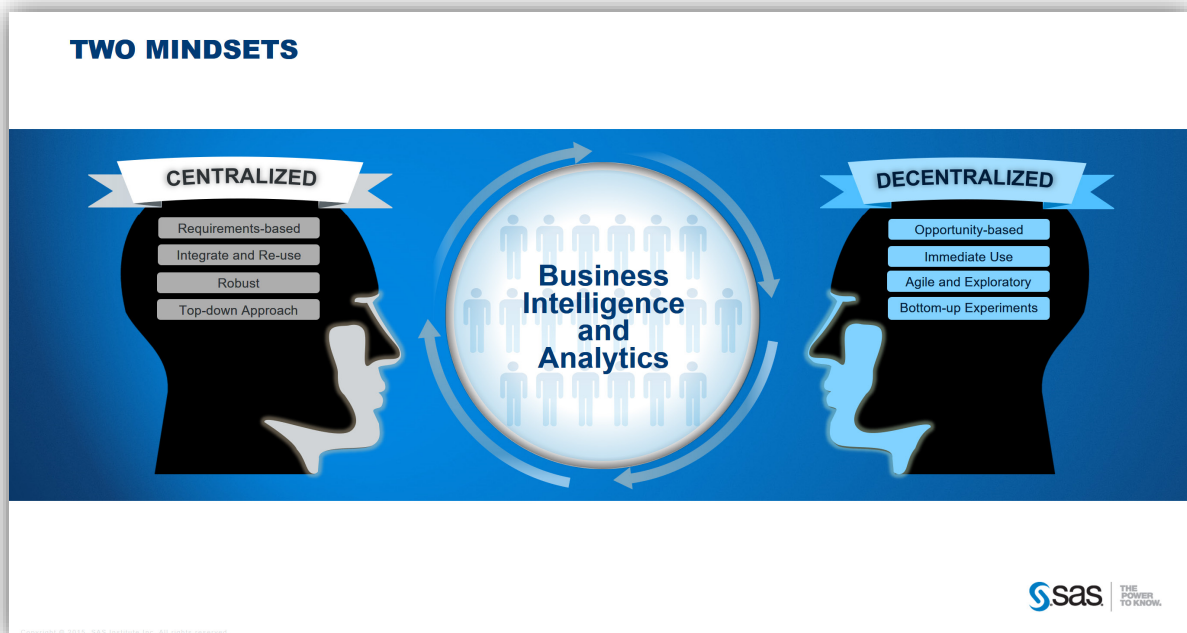
Figure 2. Two Mindsets

There will always be a need for both managed business intelligence and reporting and self-service data discovery and analytics. A single platform that can meet the needs of both IT and business analysts is the best for a modern BI system. IT needs to embrace the fact that business users will always demand a decentralized, self-service approach. However, if IT can monitor and leverage the content created by business users to selectively move through production processes and into their managed BI environments, then it is a win-win for both IT and business teams. For example, BI platforms should allow users to promote BI content and models generated in sandboxes to shared environments to use and collaborate on reports, dashboards, models, and infographics. IT and BI managers can make informed

decisions about what business analyst content is ready for the next level of investment (for example, moving it into production) with all of the security and governance steps associated with such a move.

This is where SAS comes in and where SAS Visual Analytics delivers the value: approachable analytics. SAS Visual Analytics provides one system that includes options for both parts of the business. It is both agile BI and managed BI in an all-in-one approachable toolset. It even has analytics for business analysts like forecasting, binning, clustering, decision trees, correlations, heat maps, box plots, and much more.

SAS ADD-IN FOR MICROSOFT OFFICE
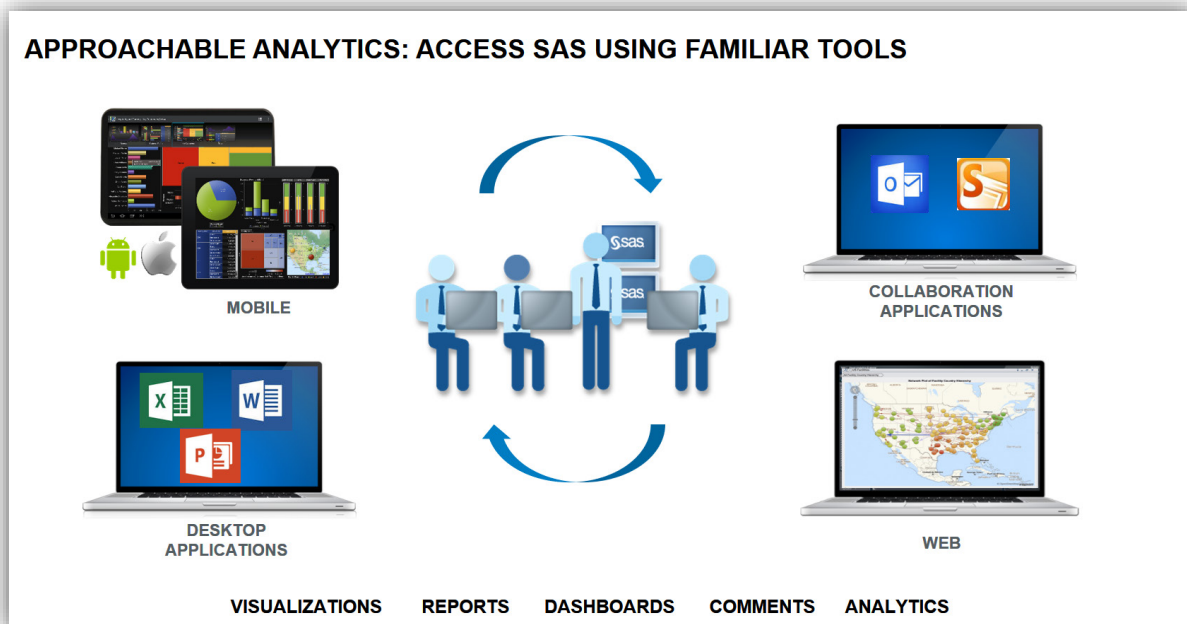
Figure 3. SAS Makes It Easy

As more people have access to data and analytics are applied to the data, it becomes increasingly important that people can easily have conversations about their BI dashboards and reports.

These conversations are critical–critical for understanding and interpretation, critical for alignment, and critical for decision making.

One of the most powerful ways to facilitate conversations around the data is to put the information where people can easily get to it!

Make the information available in the places where you are working most often. Just point and click through your familiar Microsoft Office interface or application of choice. It's that easy to analyze large amounts of data and view results directly in Microsoft Word, Excel, PowerPoint, Outlook, and SharePoint.
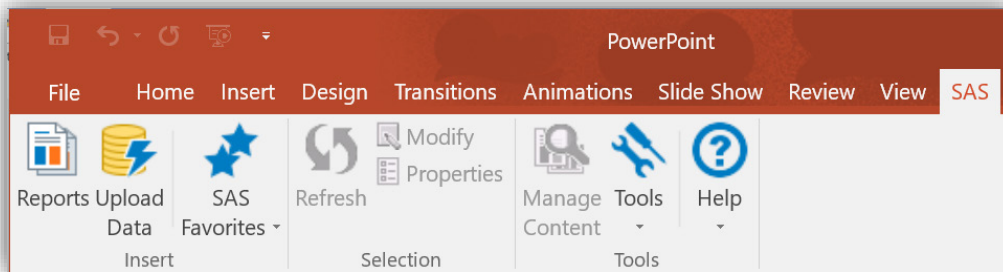
Figure 4. SAS Ribbon in Microsoft Office Showing SAS Visual Analytics Add-In for Office (PowerPoint in Office 2016)
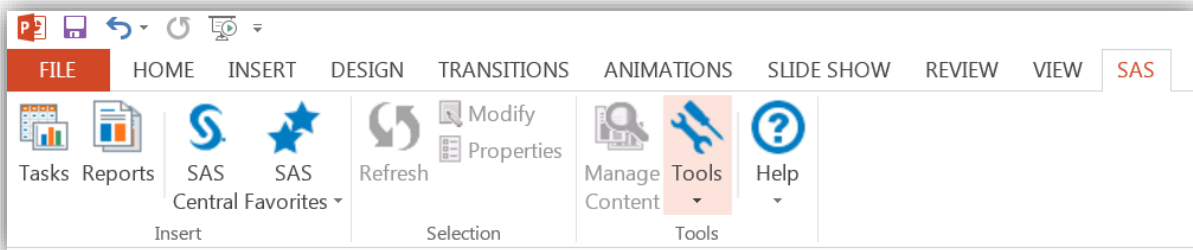
Figure 5. SAS Add-In for Microsoft Office (Office 2013)

The SAS add-in shows the licensed features based on SAS software. Both options provide users with access to SAS Visual Analytics to include in their office documents. The SAS Visual Analytics Add-In for Office is available only for PowerPoint or Excel. If you want Word, Outlook, and SharePoint, then the full SAS Office Analytics solution needs to be purchased. Additional self-service powerful analytical and data preparation tasks are unlocked in the full SAS Office Analytics solution. This latest offering for SAS Visual Analytics provides a great value for a team to grow into full office analytics with SAS.

I am focusing on the integration with PowerPoint and Excel because both are included and are the most likely tools for the business analyst to create data-driven infographics in the workplace for number-crunching or presentations.

Connect your data with SAS Visual Analytics and import whole tables, graphs, or reports directly into Microsoft Office.

WHY IS SAS DIFFERENT?

Big data is extremely relevant. It is not just a buzz word and it is here now. The vast increase in data has changed the landscape for analytics and data visualization. The *volume* of data, the *variety* of data sources, and the *velocity* of data input and the required *velocity* of data output are expected by business stakeholders. The Internet of things and surrounding applications are an opportunity that's never existed. Analytics drives decisions in every aspect of machine-to-machine communications and decisions. Data visualization importance grows to ensure that valuable insight is communicated to stakeholders from all of this data.
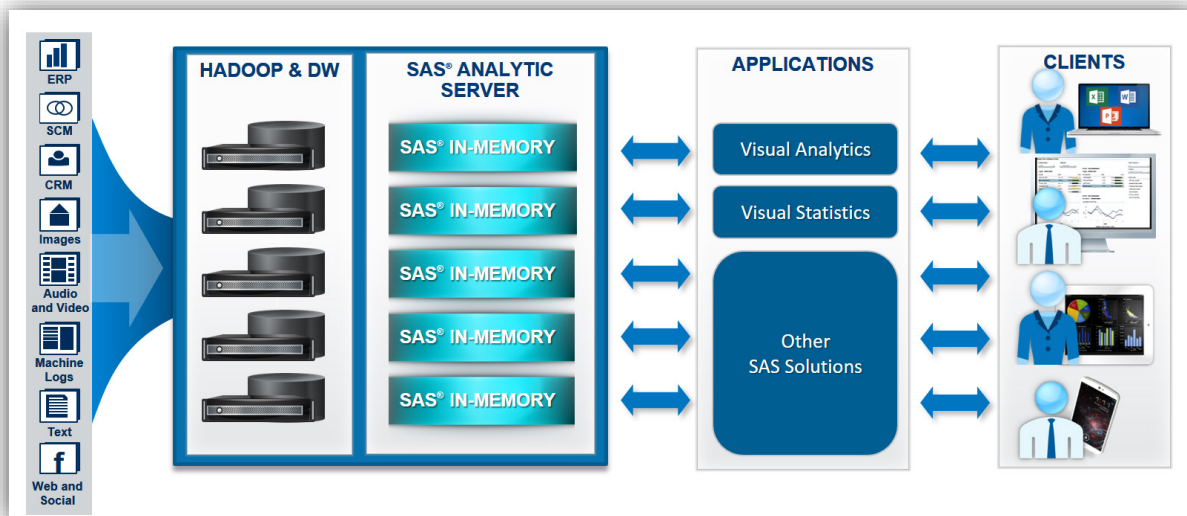
Figure 6. SAS Analytics Can Be Leveraged from Easy-to-Use Applications via Many Accessible Clients

With the rise of and applications for big data, the analytics platform must scale to handle volumes today and the massive increase in volume occurring constantly. It is no longer acceptable to build an analytics platform and data visualization suite that do not have scalability at the forefront. SAS is an analytics platform that provides a very fast, secure, multi-user environment for concurrent access to data that is loaded into memory. It takes advantage of multi-threaded, distributed computing by distributing data and workload among multiple machines and by performing massively parallel processing. It is engineered to address iterative and interactive analytics, and to present results via tools like SAS Visual Analytics. SAS is not an in-memory database. Instead of just selecting rows of data and performing basic calculations (for example, for query, aggregation, and data manipulation), SAS can perform actions and analytical calculations on data. The velocity is handled with real-time capabilities built into the engine. This engine underpins the data visualization and infographics in this paper.

THE APPROACHES

The approach for creating infographics with SAS is to create repeatable dashboard-style business infographics versus the artistic infographics that can look great as posters on the kitchen wall, but can take months to update or create. I have backed a few of these artistic infographic posters on Kickstarter myself. On the flip side, the aim here is creating a business-ready environment to take some of the benefits of the infographics to the enterprise stakeholders or to the C-level executive in the boardroom. We need to continue to innovate how we capture and keep the attention of stakeholders. This is an option for you right now.

Here are three options for incorporating SAS analytics into an infographic.

OPTION ONE–BESPOKE DESIGN

Option one is to do everything manually; for example, build and design as a graphic artist, data artist, or journalist would. This option could also mean doing everything automatically. By automatic, I mean use SAS code (SAS/GRAPH and ODS) to build the entire infographic. This is an accurate and valid way to build the infographic and leverage the power in the SAS analytics platform. It does have one barrier though: it requires you to code in SAS. This is great for experienced SAS programmers. However, this is

not the focus of this paper. I will leave this skill to the other SAS experts who have done a heap of work in this area, like Rob Allison, who is prolific in providing working examples of "the art of the possible" with SAS/GRAPH as a data visualization engine. An example of Rob's work can be seen in Figure 7 (Allison 2013). I find that this option is best used when you have white-boarded your infographic design and have a clear vision of where all of the elements will live. This option can be done in batch to create files like HTML, PDF, etc., or can be packaged and run on-demand using a SAS® Stored Process.
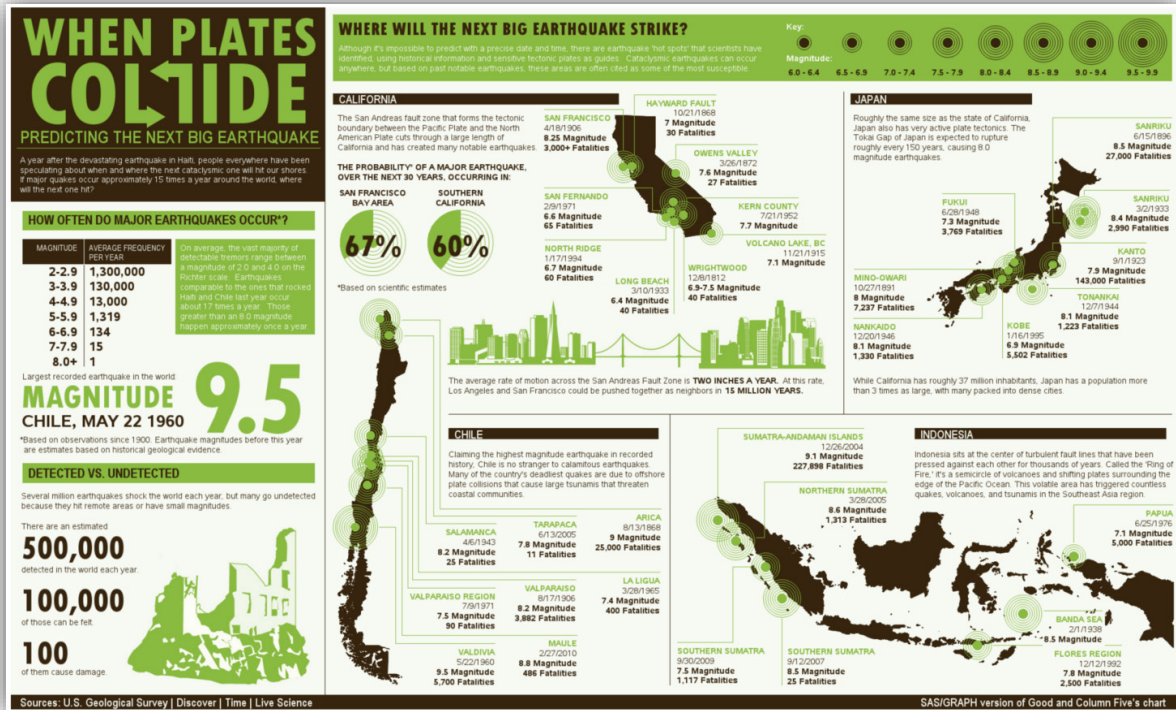
Figure 7. Infographic Created by Rob Allison Using SAS/GRAPH with Built-In Links to Drill Through to Details

OPTION TWO–DRAG-AND-DROP TOOLS– NO CODING REQUIRED

Option two is where you want some of your infographic content to be entered as subject comments and graphics and other content imported easily from SAS Visual Analytics. This option enables you to leverage the large investment in data assets in your enterprise, as well as provide the creative freedom that you need to add the narrative required to communicate the insight in the eight-second window.

This option uses approachable analytics tools, such as SAS Visual Analytics, and your Microsoft Office tools to make it accessible by any business analyst or report user. This option can unlock big data analytics into the infographics created by designers.
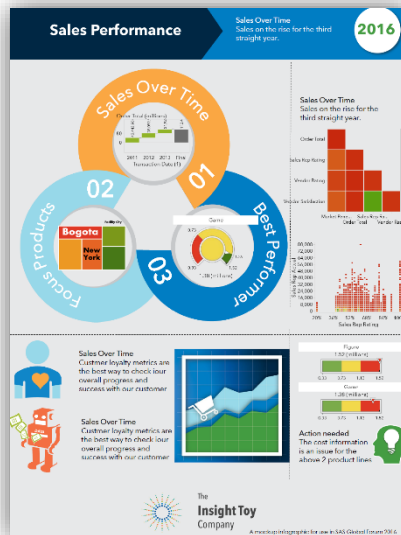
Figure 8. Infographic Using SAS Visual Analytics and SAS Office Analytics

This option focuses on the non-programmer and leverages the drag-and-drop GUI tools from SAS and the familiar Microsoft Office productivity suite.

OPTION THREE–HYBRID–A BIT OF BOTH OPTIONS

Option three is where you leverage bespoke analytics from SAS to embed elements into your infographics. This option leverages custom code or SAS stored processes from your data science team and creates a link to run each month or quarter. This means you are not limited in using reports. You could include custom SAS stored processes to insert an element of your infographic driven from bespoke analysis, right next to objects inserted via option two from the drag-and-drop method.

SAS® Enterprise Guide® options are not covered in this paper. However, it is an important part of the SAS Office Analytics suite. You can refresh any reports that you send from SAS Enterprise Guide to a Microsoft Office document in a flow, and then open the reports with links to the original SAS source when you open them in Microsoft Office. This aids in repeatability and ease of use for your infographics.
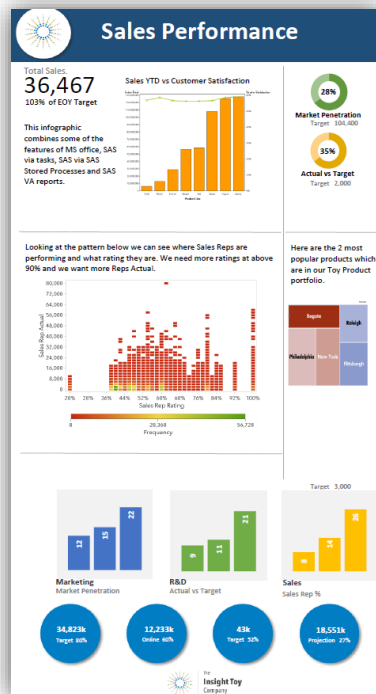
Figure 9. Multiple Graphs

The top graphic (dual axis bar and line graph) is driven by a SAS stored process, mixed with some Microsoft Excel charts and some SAS Visual Analytics charts. The SAS stored process is leveraging a code object written by one member of the data science team.

MY APPROACH

Of the three options, I have chosen to focus on option 2. I will be leveraging graphics created in SAS Visual Analytics and inserting them into my infographic template inside PowerPoint and Excel. This requires no coding skills at all, and it is the most approachable option to get started today.

To do this correctly, I need to do some work inside SAS Visual Analytics first to create the data-driven objects for my infographics. Of course, I could just open my favorite SAS Visual Analytics report and grab any object. This means that I have not optimized my infographic's look and feel, though. I need to use the SAS Visual Analytics Designer to create my optimized report elements for best use in this highly visual design.

You need to choose the subject area and focus on it as you step through the following information. I chose to focus on the sales data for my organization: a fictitious toy company that has sample data shipped with SAS Visual Analytics. This focus enables me to maximize my efforts as I consider the visual elements.
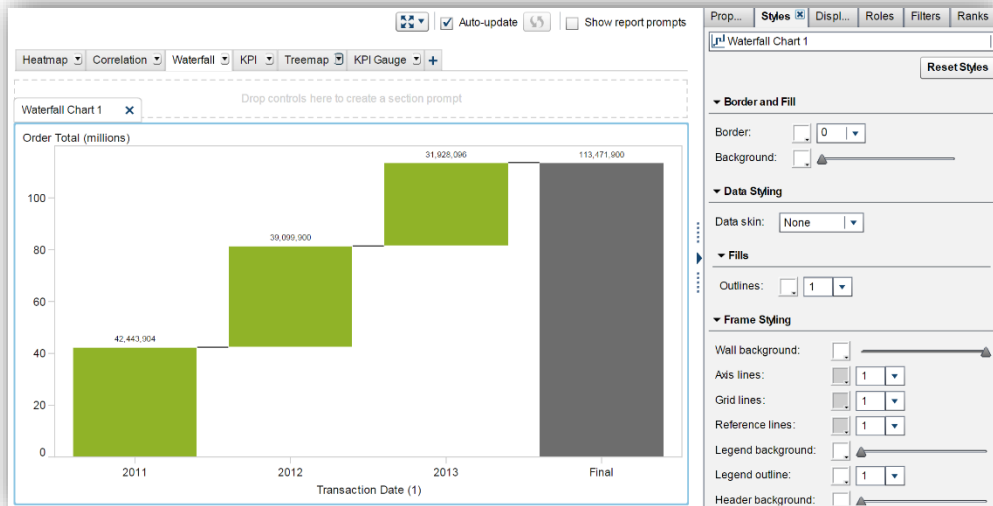
Figure 10. Create Visual Elements

Next, I need to ensure that I have all of my images and creative content for the infographic. This includes images for backgrounds and non-data-driven visuals, like logos and layout graphics (for example, dividers, and so on). At this point, consider fonts that best reflect the look that you are after. All of these touches make the look of the infographic much nicer. These steps will save you a huge amount of time. You could call these assets, or your *infographic toolkit,* the building blocks for your design.
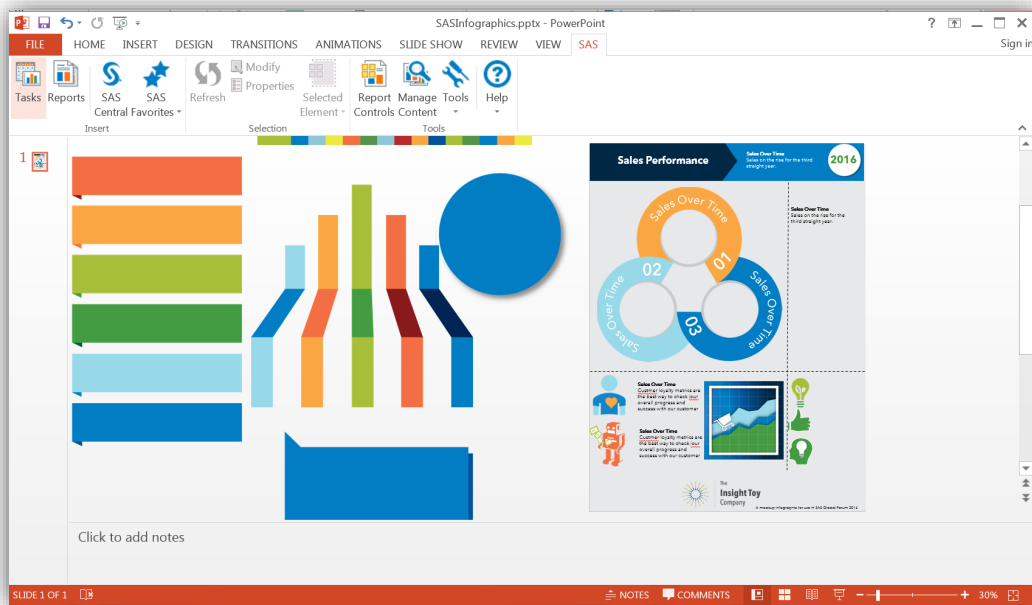
Figure 11. Example of external media to create your infographic

It is now time to unify these visual elements into the infographic itself. If you are anything like me, this can take a long time to get just right. Many iterations of design and trial and error will probably occur before you have the desired layout and information to tell the narrative that you want from your data.

In Figure 12, you can navigate to all of your SAS Visual Analytics reports and analyses and insert desired elements into your infographic. This cannot be any easier, and don't forget that the computation and number-crunching is happening on the SAS platform, not on your laptop.
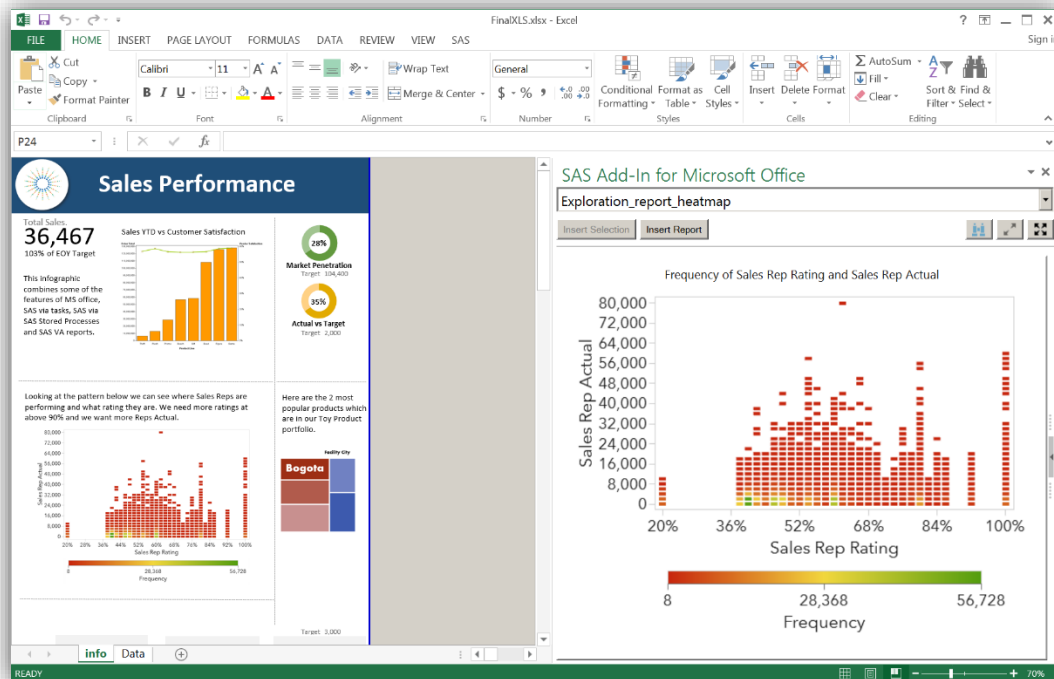
Figure 12. SAS Add-In for Microsoft Office Navigating to SAS Report from Microsoft Excel

Once you have added all of the desired content from SAS, you can add your subjective commentary and narrative to tell the story that you want in your infographic. Now, you are ready to see the end result. The way I have done this is to use the built-in **Save As** option to save to PDF, which is a publication- and print-ready format that you can share without allowing others to change the data. If I do want others to edit the design, I can just email the document and they can edit the design. They cannot update the data if they don't have security permissions in the SAS analytics engine, though. This provides a great balance of creative freedom and data governance.

Figure 13 shows the final product of the infographics created with SAS Visual Analytics and SAS Office Analytics. These infographics show highlights such as the top three sales performers and sales by year for the past three years. The focus of these infographics is sales and company performance data, and they show sales performance at a glance.
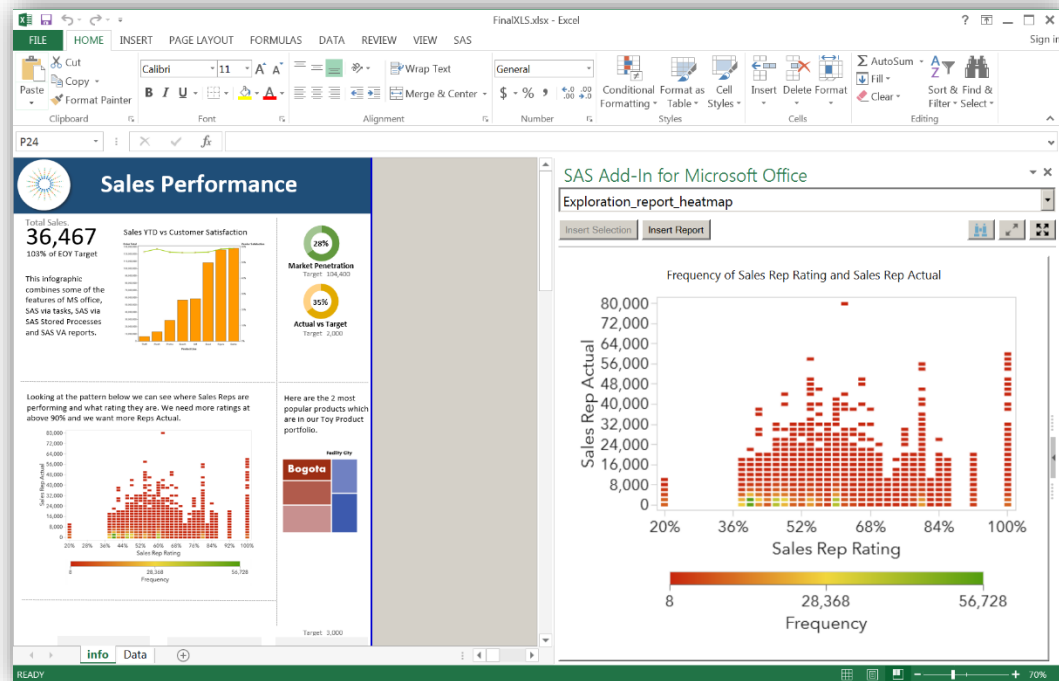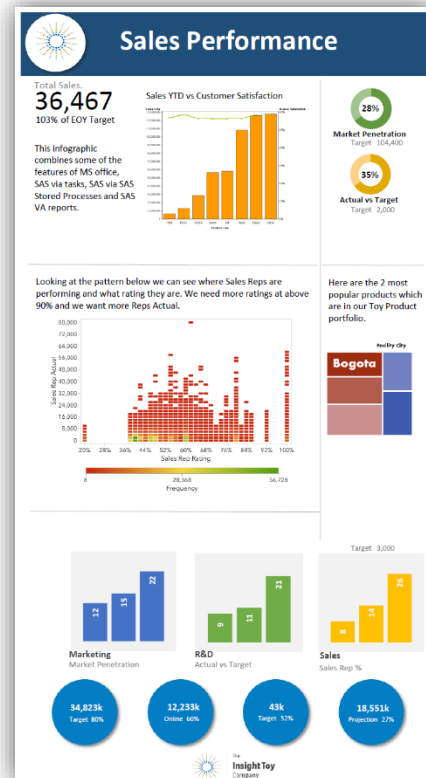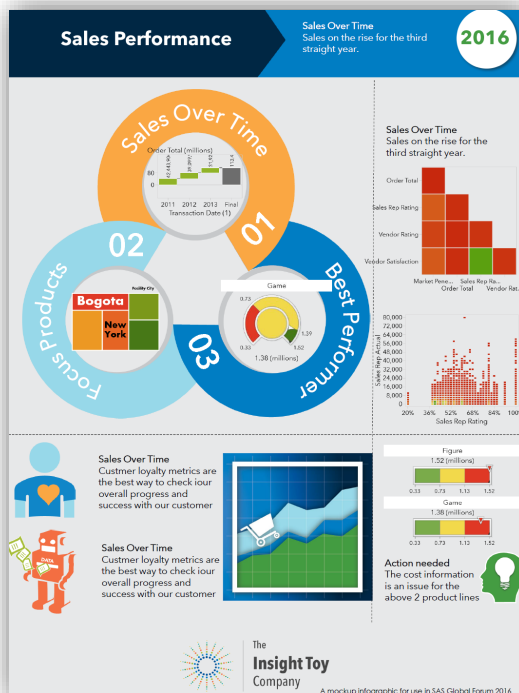
Figure 13. Final Infographics from PowerPoint and Excel

TIPS TO GET YOUR INFOGRAPHIC JUST RIGHT

To achieve this look with your own data, there are a few tips to keep in mind.

- Start with a plan.

  - **Narrative**–What are you trying to say? Is this infographic about the products that you sell, the social media channels, or the performance of the sales team? This frame of reference enables you to create the narrative that will resonate best.
  - **Colors**–Create your color pallet to enable you to insert and match the colors that you are using in both SAS Visual Analytics and Microsoft Office. There are tools to help with color selection, such as Color Picker for Windows.
  - **Set Up the Page**–Using a background image can be a good idea and can frame the entire infographic. It helps to keep the look and feel consistent. Set the page layout for your infographic to override the defaults if you want custom page sizes.
  - **Icons and Artwork**–Have the icons and graphics ready to go. You might have access to some through work. Or, there are many Internet sites with libraries.

- Consider the SAS settings.

  - Change the styles in SAS Visual Analytics as much as possible. This will minimize changes in Microsoft Office and make the infographic more automated for the next data refresh.
  - Change output options to PNG (from default ActiveX) in the SAS Add-In for Microsoft Office.
  - Apply styles when inserting content only. This means that refreshed content should not be affected if you change styles and the layout in your infographic.

- Turn on the background and wall transparencies in the SAS Visual Analytics report. This does not really change the SAS Visual Analytics side. When you insert these as part of the infographic, the background color will be the same as the layout that you have in Microsoft Office.

- Create final touches.

  - If some visual elements cannot be switched off in an object, include some shapes with solid fill to blend the objects with the background color. This will assist with your infographic precision.
  - Consider adding a link to the SAS Visual Analytics server in your design. This is a great way to unite the infographic user with the live dashboard. This can drive your audience to the self-service analytics platform.

CONCLUSION

The aim of this paper is to introduce a new use case for your trusted software from SAS, specifically SAS Visual Analytics and SAS Add-In for Microsoft Office. I hope that this paper has provided some ideas on how you could achieve data-driven visualization in your business. I know that a step-by-step guide to create what is outlined in this paper would be a great asset. As an alternative, I plan to provide a video of creating the infographics, and I will post it to the SAS Communities site following SAS Global Forum 2016. I encourage you to share your creations on the SAS Communities SAS Visual Analytics site.

Remember that infographics and data visualizations need to include narrative or context to meet the needs of the audience, not just show aesthetics. You can have repeatable infographics with governed access to analytics resources within your enterprise. This leverages the power of big data and the SAS analytics engine with your familiar Microsoft Office productivity tools that you use every day.

REQUIRED SOFTWARE

Support Documentation at http://support.sas.com/software/products/addin/index.html

Here are the specific software versions used in this paper:

- Microsoft Office 2016 and 2013

- SAS Visual Analytics 7.3

- SAS Office Analytics, SAS Visual Analytics Add-In for Office (with limited features from SAS Office Analytics), and SAS Add-In 7.12 for Microsoft Office

  - SAS Visual Analytics Add-In for Office is available in Microsoft PowerPoint and Excel and works only with SAS Visual Analytics.

REFERENCES

Tukey, John W. 1977. *Exploratory Data Analysis*. Reading, Massachusetts: Addison-Wesley.

SAS Institute Inc. 2015. *SAS® Visual Analytics: User's Guide*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/onlinedoc/va/index.html.

Time.com. "You Now Have a Shorter Attention Span than a Goldfish." Kevin McSpadden. 2015. Available at http://time.com/3858309/attention-spans-goldfish/.

Nieman Reports. "Watching the Human Brain Process Information." 2010. Available at http://niemanreports.org/articles/watching-the-human-brain-process-information/.

SAS Learning Post–Blog Post. "Creating fancy 'infographics' with SAS." 2013. Robert Allison. Available at http://blogs.sas.com/content/sastraining/2013/04/11/creating-fancy-infographics-with-sas/.

RECOMMENDED READING

Bailey, D., Tim Beese, and Casey Smith. 2015. "Take Your Data Analysis and Reporting to the Next Level by Combining SAS® Office Analytics, SAS® Visual Analytics, and SAS® Studio." *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at https://support.sas.com/resources/papers/proceedings15/SAS1804-2015.pdf.

Bailey, D., Anand Chitale, and I-Kong Fu. 2014. "Share Your SAS® Visual Analytics Reports with SAS® Office Analytics". *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute, Inc. Available at http://support.sas.com/resources/papers/proceedings14/.

Devarajan, R., H. Patel, P. Berryman, and L. Everdyke. 2014. "Create Custom Graphs in SAS® Visual Analytics Using SAS® Visual Analytics Graph Builder." *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings14/SAS346-2014.pdf.

SAS Institute Inc. *SAS® Visual Analytics: Video Library*. Cary, NC: SAS Institute Inc. Available http://support.sas.com/training/tutorial/va73/.

SAS Institute Inc. 2015. *SAS® Visual Analytics: User's Guide*. Cary, NC: SAS Institute Inc. Available http://support.sas.com/documentation/cdl/en/vaug/67500/PDF/default/vaug.pdf.

SAS Institute Inc. *SAS Visual Analytics Community*. SAS Support Communities. Cary, NC: SAS Institute Inc. Available https://communities.sas.com/community/support-communities/sas-visual-analytics.

Allison, Robert. 2015. "Robert Allison's SAS/Graph InfoGraphics!" Available at http://robslink.com/SAS/democd_infographics/aaaindex.htm.

SAS Institute Inc. 2016. *SAS® Office Analytics Fact Sheet*. Cary, NC: SAS Institute Inc. Available at http://www.sas.com/content/dam/SAS/en_us/doc/factsheet/sas-office-analytics-105595.pdf.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Travis Murphy
SAS Institute Australia and New Zealand
300 Burns Bay Road
Lane Cove, Sydney, NSW Australia 2066
Travis.Murphy@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

SAS Visual Statistics 8.1: The New Self-Service Easy Analytics Experience

Xiangxiang Meng, Cheryl LeSaint, Don Chapman, SAS Institute Inc.

When you are analyzing data, it is important that you can easily identify relationships between the variables. By identifying relationships, you can make predictions for variables of interest. A common variable of interest is a binary variable. Should a person be admitted to a program? Is this transaction fraudulent? This paper examines data on whether a customer cancels his or her subscription, which is also known as churn. The churn variable's relationship with other information about the customer's account is examined in a variety of ways throughout this paper. The exploratory data analysis and feature engineering all build up to identifying significant predictors of churn through a logistic regression. The combination of SAS Visual Analytics 8.1 and SAS Visual Statistics 8.1 brings data analysis and reporting to your fingertips.

Xiangxiang Meng, PhD, is a Senior Product Manager at SAS. The current focus of his work is on SAS Visual Statistics, cognitive computing, the Python interface to SAS Cloud Analytic Services, and other new product initiatives. Previously, Xiangxiang worked on SAS LASR Analytic Server, SAS In-Memory Statistics for Hadoop, SAS Recommendation Systems, and SAS Enterpriser Miner. His research interests include decision trees and tree ensemble models, automated and cognitive pipelines for business intelligence and machine learning, and parallelization of machine learning algorithms on distributed data. Xiangxiang received his PhD and MS from the University of Cincinnati. Xiangxiang is also the co-author of *SAS Viya: The Python Perspective*.

support.sas.com/meng

SAS® Visual Statistics 8.1: The New Self-Service Easy Analytics Experience

Xiangxiang Meng, Cheryl LeSaint, Don Chapman, SAS Institute Inc.

Excerpt from SAS Global Forum 2016 *Proceedings*

ABSTRACT

In today's Business Intelligence world, self-service, which allows an everyday knowledge worker to explore data and personalize business reports without being tech-savvy, is a prerequisite. The new release of SAS® Visual Statistics introduces an HTML5-based, easy-to-use user interface that combines statistical modeling, business reporting, and mobile sharing into a one-stop self-service shop. The backbone analytic server of SAS Visual Statistics is also updated, allowing an end user to analyze data of various sizes in the cloud. The paper illustrates this new self-service modeling experience in SAS Visual Statistics using telecom churn data, including the steps of identifying distinct user subgroups using decision tree, building and tuning regression models, designing business reports for customer churn, and sharing the final modeling outcome on a mobile device.

INTRODUCTION

When analyzing data, it is important to be able to easily identify relationships between the variables. By identifying relationships, you are able to make predictions for variables of interest. A common variable of interest is a binary variable. Should a person be admitted to a program? Is this transaction fraudulent? This paper examines data on whether a customer cancels his or her subscription, which is also known as churn. The churn variable's relationship with other information about the customer's account is examined in a variety of ways throughout this paper. The exploratory data analysis and feature engineering all build up to identifying significant predictors of churn through a logistic regression. The combination of SAS® Visual Analytics and SAS® Visual Statistics in the 8.1 release brings data analysis and reporting to your fingertips. The 8.1 release of SAS Visual Analytics and SAS Visual Statistics was pre-production when this paper was authored; therefore, details are subject to change.

SAS VISUAL STATISTICS 8.1

UNIFICATION

The 8.1 release of SAS Visual Analytics is rebuilt from the ground up and combines SAS® Visual Analytics Explorer, SAS® Visual Analytics Designer, and SAS Visual Statistics into a single user interface. Pie charts, histograms, and linear regressions meld together. Exploratory tasks, such as auto-charting and summary tables, sit alongside analytical tasks, such as linear regression and clustering. These tasks are seamlessly blended in classic reporting capabilities, such as a rich layout system and display rules. This unified experience allows SAS Visual Analytics and SAS Visual Statistics users to execute the Business Intelligence and Analytics continuum, as shown in Figure 1.

MODERN DESIGN

The modern user experience supports the modeling needs of statisticians and data scientists as well as the reporting needs of business analysts. The application is designed to run in your browser and is written entirely in HTML5. This opens up options for developing models away from the desktop.

The underlying infrastructure has been rewritten to meet the deployment demands of the future. It supports more versatile deployment options ranging from on-site to in-the-cloud deployments by

leveraging microservices that meet the scaling and reliability needs expected in today's world. You can also take advantage of the next-generation in-memory analytics server from SAS® that unifies all the analytics procedures into a single server.

Data access is built into SAS Visual Statistics. You have full access to enterprise data stored in Hadoop or in the corporate database. You also have the ability to upload your own data stored in plain text files, Excel spreadsheets, or a SAS data set. This level of self-service, along with a rich set of data manipulation capabilities, enables the feature engineering expected by sophisticated users.
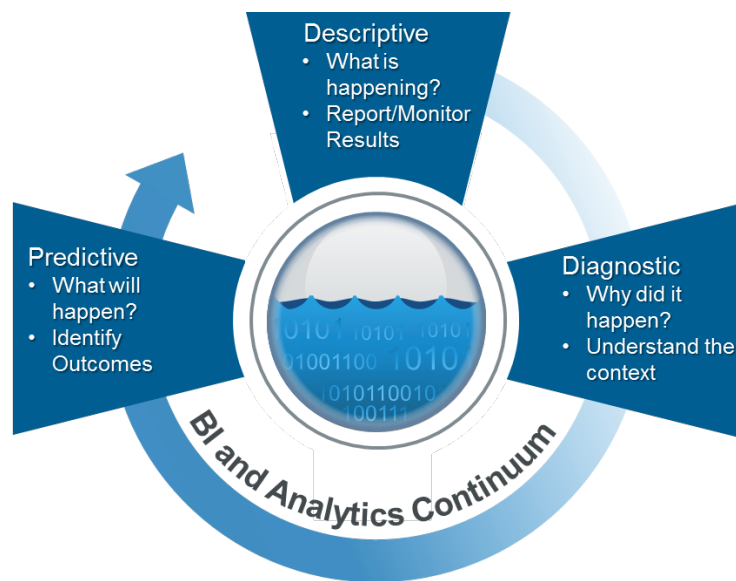
Figure 1. Business Intelligence and Analytics Continuum

USER EXPERIENCE

SAS Visual Analytics and SAS Visual Statistics has been seamlessly integrated, making all modeling work immediately accessible in a report. Documenting and sharing models are easy; you just save your work. This work can be viewed on mobile devices using the SAS Visual Analytics mobile viewers.

Often the generation of a report is secondary to interactively building models and generating score code. SAS Visual Statistics does not require report layout; it is there only if needed. It also provides the tools to compare two models side-by-side, or to evaluate them interactively using the model comparison task. Figure 2 provides a basic introduction to the layout of the user interface of SAS Visual Analytics 8.1. SAS Visual Statistics is an add-on to SAS Visual Analytics, which provides additional statistical modeling tasks in the **Content** menu.

Page Navigation: navigate and manage pages

Banner: help, search, user setting, log out

Toolbar Menu: common application actions such as save and open

Left Pane
Access data
Choose report objects
View report outline

Report Canvas: lay out report objects into a page

Drop a data item or content here

Right Pane: Define and modify a report object (Options, Roles, Actions, Rules, Filters, Ranks)
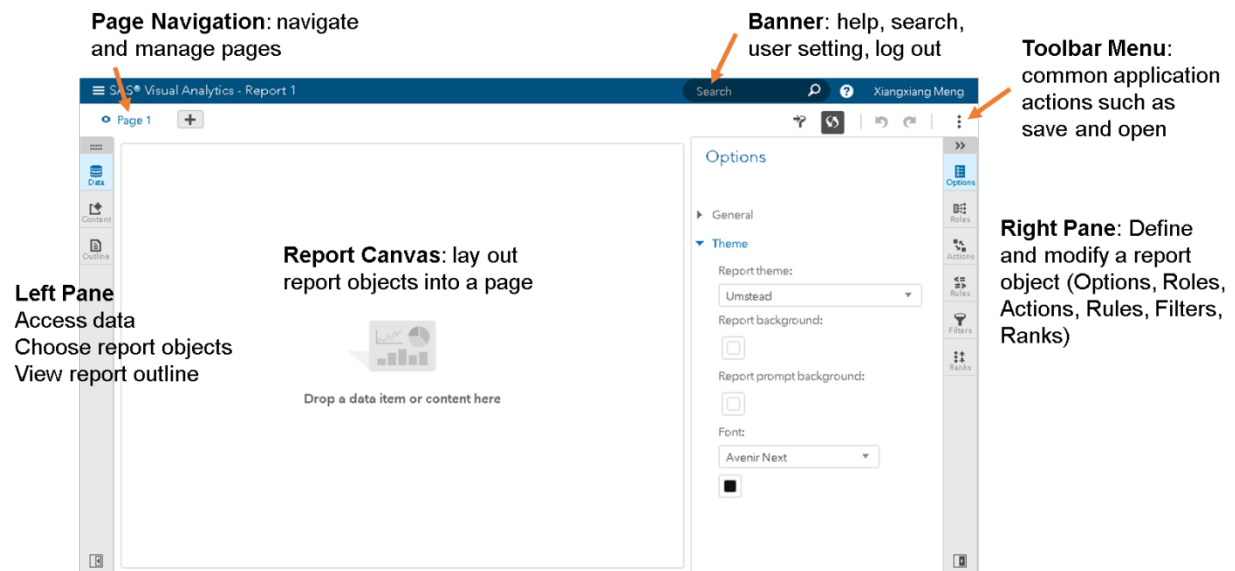
Figure 2. User Interface of SAS Visual Analytics 8.1

CASE STUDY USING TELECOM CHURN DATA

DATA DESCRIPTION

The examples in this paper are derived using telecommunications data. The variable of interest is whether the customers cancel their service or not, also known as churn. The data set is available from the UCI Machine Learning Repository of databases. The data set contains 3,333 observations, where each row contains the information collected for a customer account. Table 1 provides a brief description about the variables in this table.

Variable Name	Description
Churn	Label of churn (Yes/No).
Day_Calls, Day_Charge, Day_Mins	Total day calls, charges, and minutes
Eve_Calls, Eve_Charge, Eve_Mins	Total evening calls, charges, and minutes
Night_Calls, Night_Charge, Night_Mins	Total night calls, charges, and minutes
Intl_Calls, Intl_Charge, Intl_Mins	Total international calls, charges, and minutes
Account_Length	Length of account before churn
CustServ_Calls	Total number of customer service calls
State	State of USA
Intl_Plan	Indicator for international plan
VMail_Message	Number of voice mail messages
VMail_Plan	Indicator for voice message plan

Table 1. Description of the Telecom Churn Data

EXPLORING THE DATA

SAS Visual Analytics 8.1 provides a variety of tools for data visualization and exploration. For the churn data, we are interested in exploring the data and identifying factors that influence churn. Figure 3 shows a collection of several visualizations created in SAS Visual Analytics 8.1.
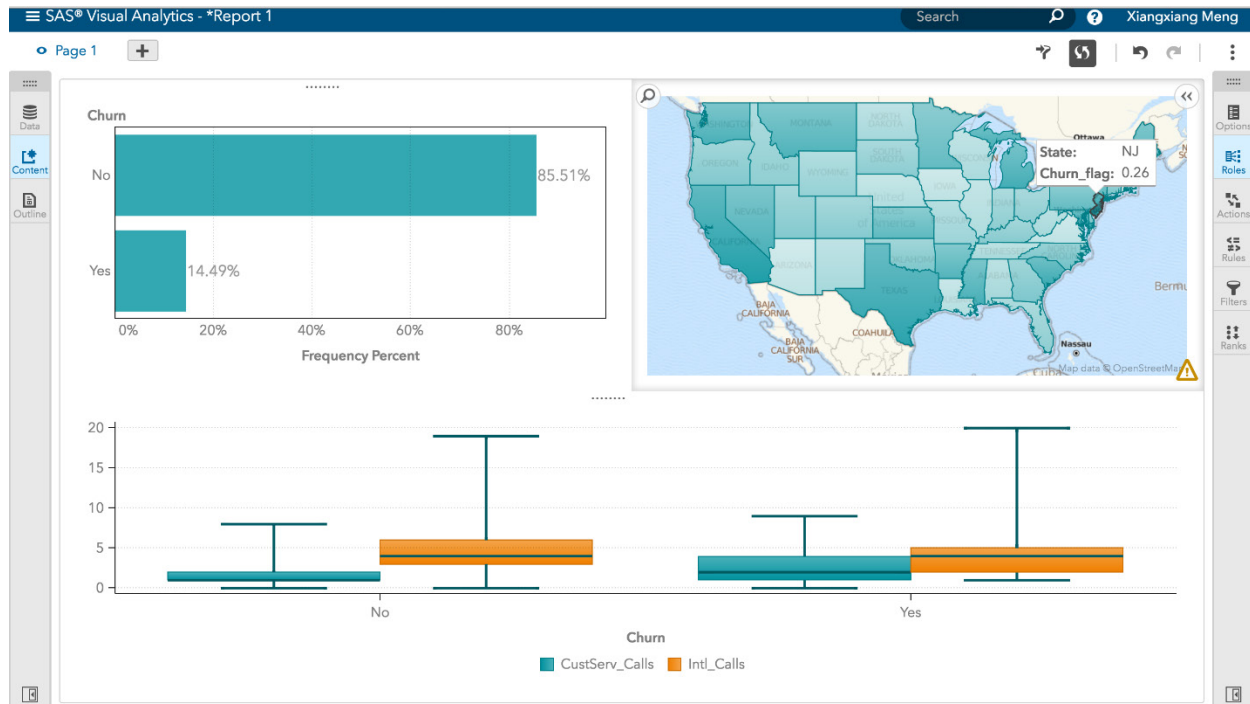
Figure 3. Data Exploration Using SAS Visual Analytics 8.1

The bar chart shows that 14.5 percent of the customer accounts are churners. The geo map shows the percentage of churning across different states. New Jersey has the highest churn rate (26 percent). Note that the color column used in the geo map is a new numeric calculated item CHURN=YES. We explain how to create this calculated item in the next section.

You can also look at the distribution of a data column in SAS Visual Analytics. For example, the box plots in Figure 3 compare the distribution of total numbers of international calls and customer service calls for churners and non-churners. It is interesting to see that churners have more customer service calls but fewer international calls.

FEATURE ENGINEERING

Feature engineering is an important step to improve the performance of a statistical or machine learning model, and it is recognized as the most manual and time-consuming effort in a learning process. SAS Visual Analytics 8.1 provides a few tools to create new features from existing columns. These features are created on-demand and do not require additional disk and memory footprint. Within SAS Visual Analytics 8.1, you can do the following tasks:

1. Determine whether a variable should be used as categorical or measure.

2. Create a new hierarchy using a set of categorical variables.

3. Create a custom category that represents a grouping of the levels of a categorical variable with high cardinality.

4. Create a calculated item using user-specified formulas.

Most of these options are available in the **Add** menu of the Data pane, as shown in Figure 4.
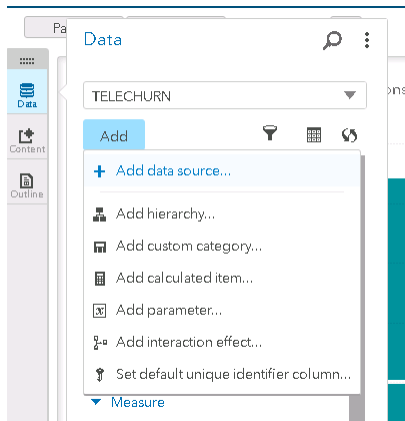
Figure 4. Add Menu for Creating New Columns

The CHURN column contains character values YES and NO, which cannot be accepted by a few visualizations that require numeric aggregators. A workaround is to create dummy indicators using a calculated item for CHURN = Yes and CHURN = No, as shown in Figure 5.
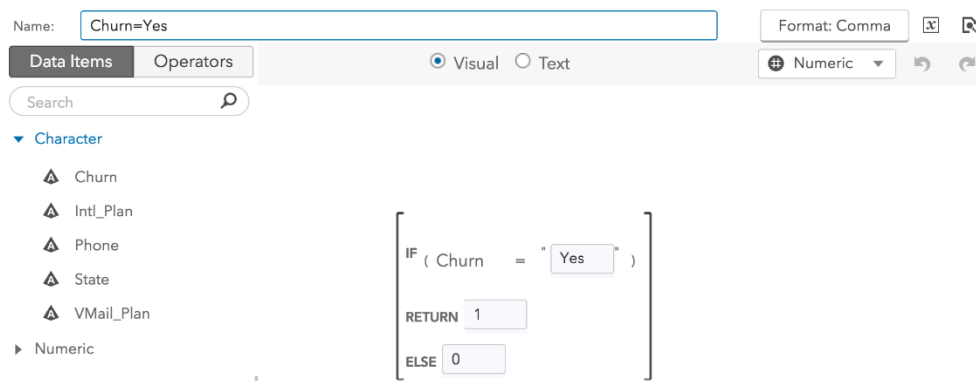
Figure 5. Create Dummy Variables in SAS Visual Analytics

You can drag and drop to create a calculated item using the data items and the operators provided in the Edit Calculated Item window shown in Figure 5, with the output column as either numeric or character. Alternatively, you can also use the text editor to create a new item. For example, the above dummy variable can be created using the following code:

```
IF ( 'Churn'n = 'Yes' )
RETURN 1
ELSE 0
```

Measure data items have a default aggregation type of Sum in SAS Visual Analytics. For the calculated item of Churn=Yes, a more natural aggregation type is Average. This change can be made in the Data pane by editing the properties of Churn=Yes.

You can now use the Churn=Yes column in a treemap to compare average churn rates across the states, as shown in Figure 6. The size of each rectangle represents the number of accounts for that state.
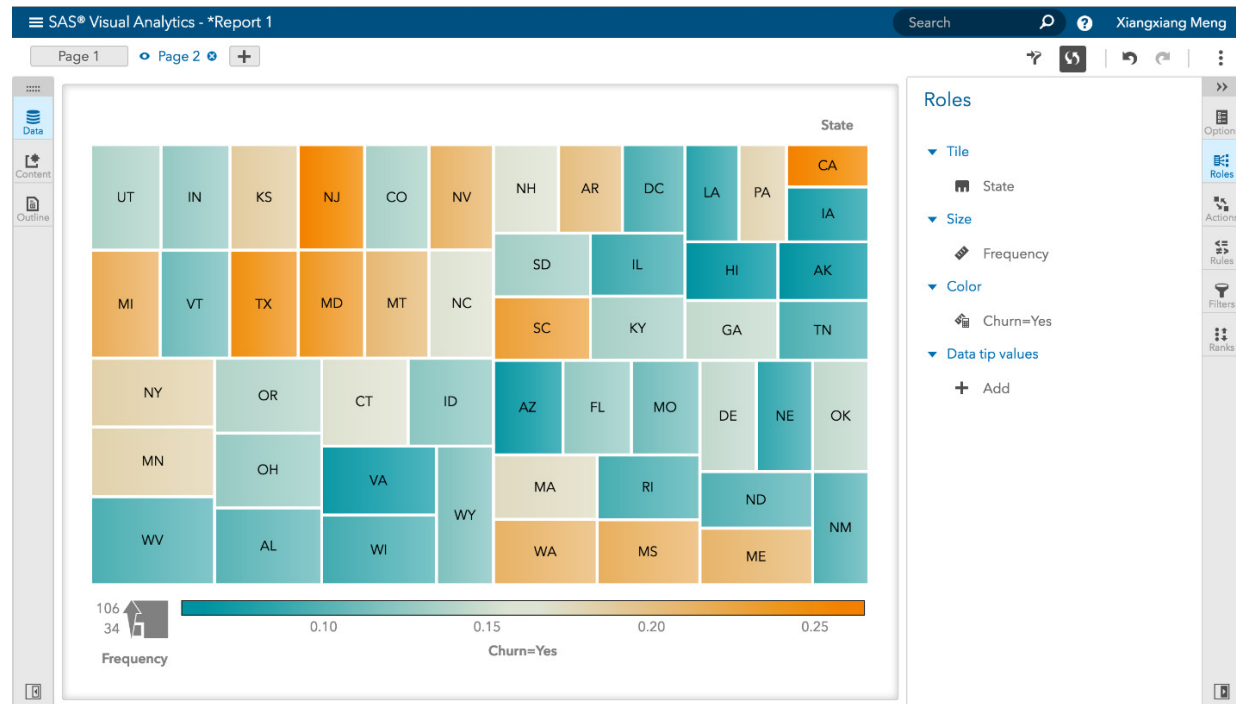
Figure 6. Treemap of Churn Rates across States Using the Calculated Churn=Yes Column

The data contains only total charges for each type of call (Day, Evening, Night, and International). You can easily derive other features such as average charge per call, total numbers of domestic calls, total domestic charge, and so on.:

```
'Day_Charge'n   / 'Day_Calls'n                    /* Day_Avg_Charge */
'Eve_Charge'n   / 'Eve_Calls'n                    /* Eve_Avg_Charge */
'Night_Charge'n / 'Night_Calls'n                  /* Night_Avg_Charge */
'Intl_Charge'n  / 'Intl_Calls'n                   /* Intl_Avg_Charge */
'Day_Calls'n + 'Eve_Calls'n + 'Night_Calls'n  /* Total_Domestic_Call*/
'Day_Charge'n + 'Eve_Charge'n + 'Night_Charge'n /* Total_Domestic_Charge*/
```

Note that deriving multiple calculated items or deriving calculated items multiple times does not require additional disk storage or data passes. The definitions of the calculated items are attached to the in-memory data source and are computed only when they are used by a visualization or model. Figure 7 shows a scatter plot of the two calculated items DAY_AVG_CHARGE and EVE_AVG_CHARGE. You can easily identify data anomalies: one account has an extremely high average evening charge and a few accounts have high average day charges, the majority of which are churners.
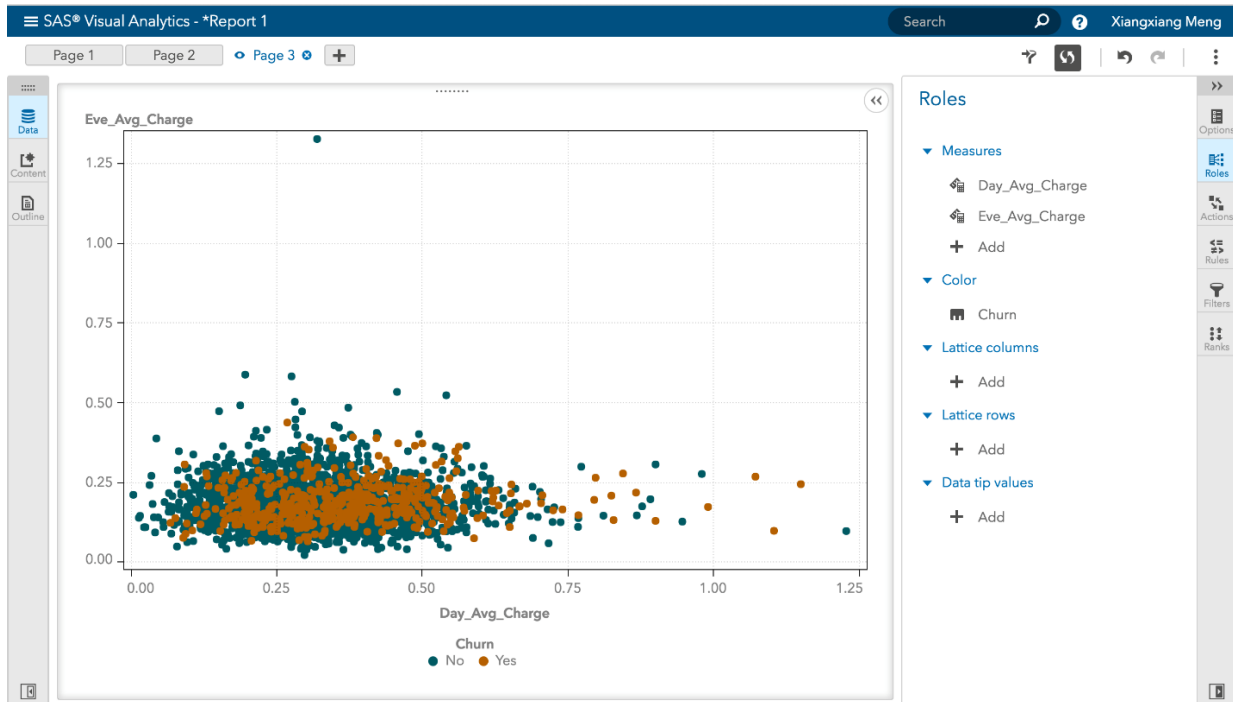
Figure 7. Scatter Plot of the Calculated Average Day and Evening Charges

DATA SEGMENTATION

The geo map in Figure 3 and the treemap in Figure 6 show that churn rate varies across states. This implies STATE is a significant factor for predicting churns. However, the STATE column is a high-cardinality variable with 51 levels and you might not want to use it in a model directly. SAS Visual Statistics provides several methods for dimension reduction and data segmentation. For example, you can use the decision tree model in SAS Visual Statistics to group the levels of a high-cardinality variable into several leaves, based on a target variable, as shown in Figure 8.

In this example, a decision tree model is built with response variable CHURN and only one predictor STATE. Decision trees are widely used in many applications such as predictive modeling, data segmentation, and outlier detection. Each application requires different tree parameter settings. For data segmentation, you often need a smaller tree to ensure each leaf of the tree has enough observations. Figure 8 shows a two-level decision tree with four branches. The tooltip shows that the third node (Node ID = 3) is the data segment with highest churn rate (19.44 percent) and contains the following states: CO, IN, KS, MA, MD, MI, MT, NC, NJ, NV, TX, UT, and WA.

Churn (event=No)   Observations Used **3,333**

Tree

Node ID: 3
Count: 895
State: CO, IN, KS, MA, MD, MI, MT, NC, NJ, NV, TX, UT, WA
No: 721(80.56%)
Yes: 174(19.44%)
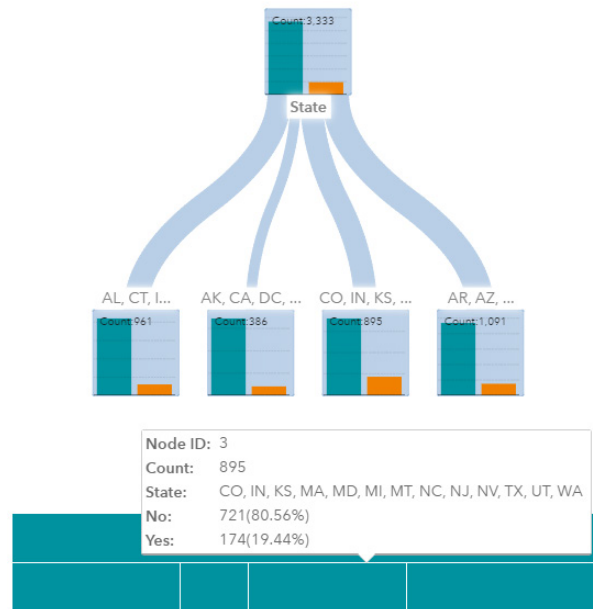
Figure 8. A Two-level Four-branch Decision Tree for Data Segmentation

It is often desirable to use the data segmentation from a decision tree model in other models. With SAS Visual Statistics, you can derive a leaf ID column that represents the leaf assignment of the observations. Figure 9 shows the right-click menu (on a mobile device, hold to pop up this menu) to derive a leaf ID variable. For this use case, the leaf ID contains four values (1, 2, 3, 4) that represent the grouping of 51 states into four segments with different levels of churn rates (CHURN = Yes).

Show percent

Derive a leaf ID variable

Export score code...

Enter interactive mode
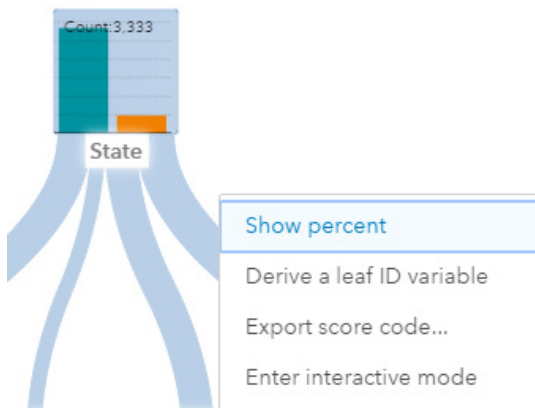
Figure 9. Derive Segmentations (Leaves) Using Decision Tree

You can edit the new derived column as well. Figure 10 shows both the Visual and Text edit windows for the column derived from the decision tree model. You can override the definition of the leaf IDs by a pre-determined business rule.

9
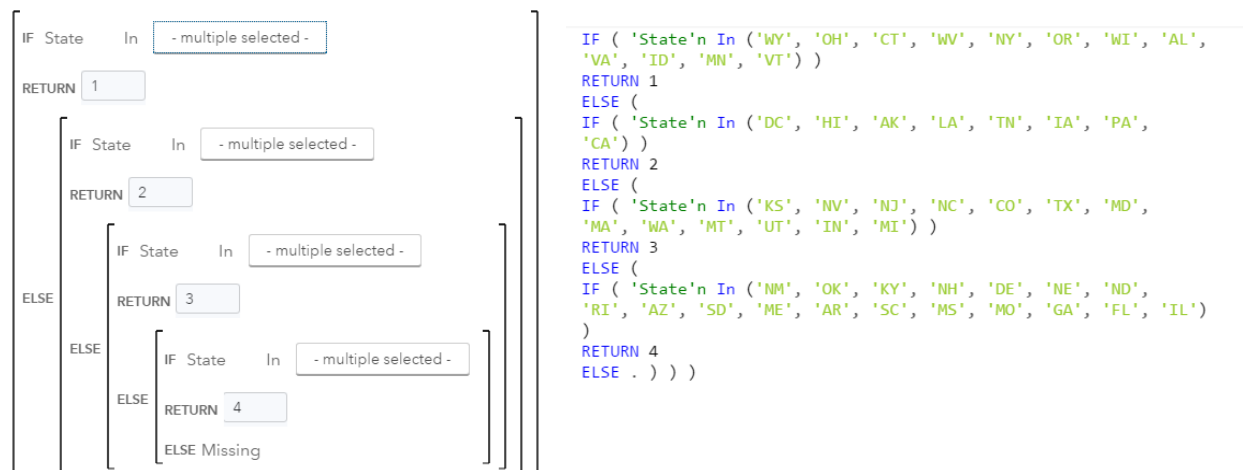
Figure 10. Editable Tree-based Segmentation

Deriving a leaf ID variable is one way of saving the results of a model. All models in SAS Visual Statistics allow you to save the analytical contents for later use. You can do the following tasks:

1. Save the model as part of the report. If you open a saved model and the underlying data source has been updated, the model is automatically retrained.

2. Save a footprint of the model as SAS DATA step code (score code). You can use the score code to score a new data source for either prediction or validation purposes.

3. Derive new columns from the model. These columns are attached to the currently loaded table and can be further used in any other models or report objects.

BUILDING LOGISTIC REGRESSION

Connectivity between different modules is also important for a self-service analytical platform. SAS Visual Statistics 8.1 allows you to derive predicted outcomes from a model and use them to build a report or another model. You can also link a model to a control or a visualization. In this section we demonstrate a group-by logistic regression use case using the data segmentation derived from the decision tree model. First let us build the basic report layout. Figure 11 shows a layout of a report page that contains a button control (top), a bar chart (middle left), and a logistic regression model (middle right). Variables have not been assigned to the components, and therefore each component displays sample output with a red warning icon to indicate that it is still under construction.
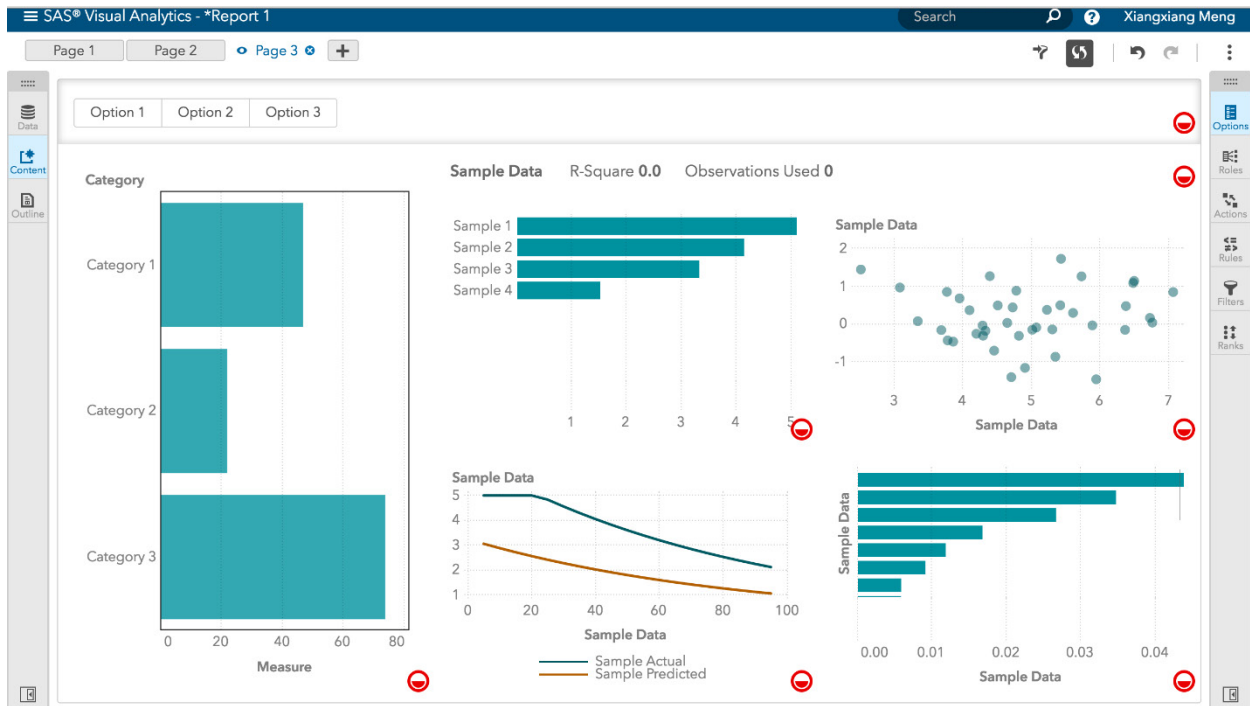
Figure 11. Layout of Various Report Objects and a Logistic Regression without Filling in Actual Data

Second, we link both the bar chart and the logistic regression to the button bar control. This is done by creating a new Filter Action from the button bar, as shown in Figure 12.
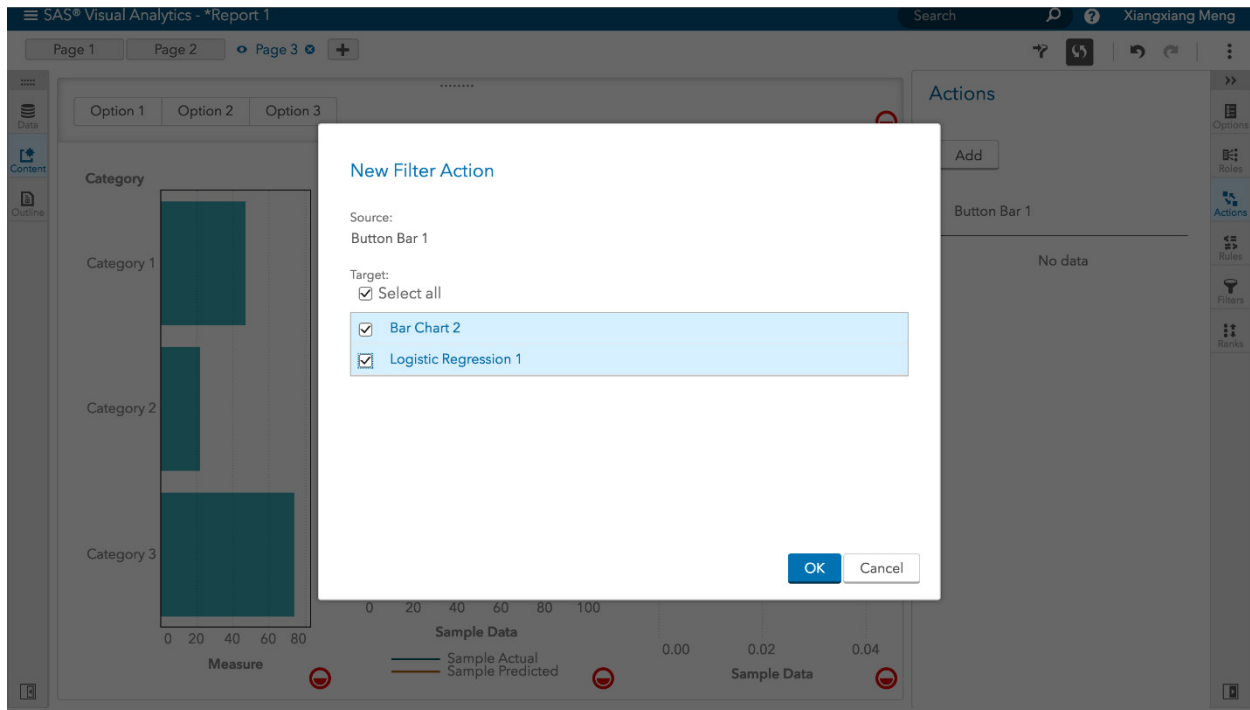
Figure 12. Creating New Filter Action

The last step is to assign data roles to each visualization. For example, you can assign the new LEAF ID to the button bar, assign STATE and CHURN to the bar chart, assign CHURN as the response variable, and assign various explanatory effects to the logistic regression model, as shown in Figure 13.
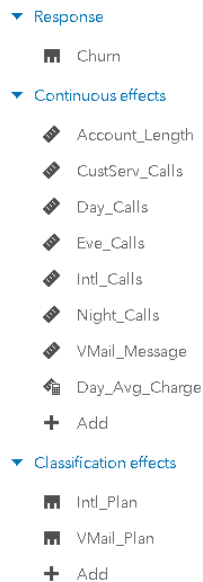
Figure 13. Response and Effects Used in the Logistic Regression

By default, the button bar control is not active and the bar chart and the logistic regression are built based on the entire data (assuming no missing values). If you click a value on the button bar control, the bar chart and the logistic regression model are updated to use only the data with a specific level of the button bar variable. Figure 14 shows a report of the churn distribution across the states and the logistic regression model built for the leaf ID = 3 data segmentation. Note that the model is trained and the visualizations are rendered only the first time you click a value on the button bar. It will be cached afterward and the model won't be refit when you switch the views.

Figure 14. Response Distribution and Logistic Regression Model for the Leaf ID = 3 Data Segment

Looking at the results for the logistic regression, you can see in the Fit Summary plot the variables that are significant at predicting whether a customer will cancel (churn). Here you can see that whether they have an international plan (Intl_Plan), the total number of customer service calls they make

13

(CustServ_Calls), the average charge for their calls during the day (Day_Avg_Charge), the total number of calls during the day (Day_Calls), and whether they have a voice message plan (VMail_Plan) are all significant. This aligns with some of the exploratory data analysis. The box plots showed more separation between churners and non-churners for CustServ_Calls than Intl_Call. The fact that Day_Avg_Charge is significant at predicting churn was seen in the scatter plot, which showed large values of this variable are associated with customers that churn. These significant predictor variables would be good to focus on when attempting to reduce customer churn.

MOBILE VIEWER

Exploratory data analysis, model construction, and report building can all be done through SAS Visual Analytics and SAS Visual Statistics using a web browser from a desktop client or mobile device. Once a final report has been settled on that summarizes the findings of your analysis, the report can be viewed by many. A saved report can be shared simply be opening it in SAS Visual Analytics Viewer. From SAS Visual Analytics Viewer, the user can view and interact with all pages in the report, email the report to others, and print interesting results.

CONCLUSION

In conclusion, SAS Visual Analytics and SAS Visual Statistics 8.1 provide a unified platform for your analytic journey. The paper uses churn data to demonstrate this new self-service experience and provides a working example of exploring, manipulating, modeling, and building business reports of the churn data, all in a single user interface.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Xiangxiang Meng
SAS Institute Inc.
Xiangxiang.Meng@sas.com

Cheryl LeSaint
SAS Institute Inc.
Cheryl.LeSaint@sas.com

Don Chapman
SAS Institute Inc.
Don.Chapman@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Create Your First Graph: Visual Data Exploration with SAS ODS Graphics Designer

Sanjay Matange and Jeanette Bottitta

Excerpt from *SAS ODS Graphics Designer by Example: A Visual Guide to Creating Graphs Interactively*

The previous chapters of this book covered the domain of visual data analytics, with a focus on big data. SAS Visual Analytics provides you with the right tools for this job. Often, however, your data needs fall into the "traditional" category, where you are dealing with clinical or pharmaceutical data on drug safety or clinical trials. Here too, visualizing the relationships between the different categories in your data can lead to quick insights, and indicate the analyses you may want to undertake.

The SAS ODS Graphics Designer application is a visual tool that enables you to visualize your data with zero programming knowledge. If you know your data and the graph you want to make, you can take a point-and-click approach to building your graphs. The application will generate the code that you can later use to create reports.

But often, you get some brand new data, and want to get quick graphical views of your data. SAS ODS Graphics Designer really shines here by enabling you to just pick the variables of interest. The application will then create for you hundreds of possible graphs that can visually show possible correlations in your data. Then, you can just use the graphs created for you, and customize them to your needs.

The SAS ODS Graphics Designer application can simplify your task of finding trends in your data.

Sanjay Matange is Research & Development Director in the Data Visualization Division at SAS, where he is responsible for the development and support of ODS Graphics software. This includes the Graph Template Language (GTL), Statistical Graphics (SG) procedures, SAS ODS Graphics Designer, and other related graphics applications. Sanjay has been with SAS for over 25 years. He is coauthor of two patents and author of four SAS Press books.

support.sas.com/matange

Jeanette Bottitta is a technical writer at SAS Institute, where she specializes in ODS Graphics software. Jeanette has over 12 years of experience writing programming guides, including the *SAS® ODS Graphics Procedures Guide*. She has worked with SAS ODS Graphics Designer since its initial release and enjoys its user-friendly interface.

support.sas.com/bottitta
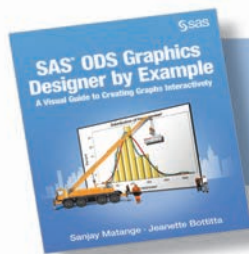
Chapter 3: Create Your First Graph

About This Example

Let's jump right in and create a commonly used graph. This example uses many of the key features of the SAS ODS Graphics Designer (the designer). Some of these features are described in more detail later in the book. For this example, the right amount of description needed to create the graph and gain a basic understanding of the process is provided.

Let's make a distribution plot of city gas mileage for all cars in the Sashelp.Cars data set:

Figure 3.1 Distribution of City Gas Mileage
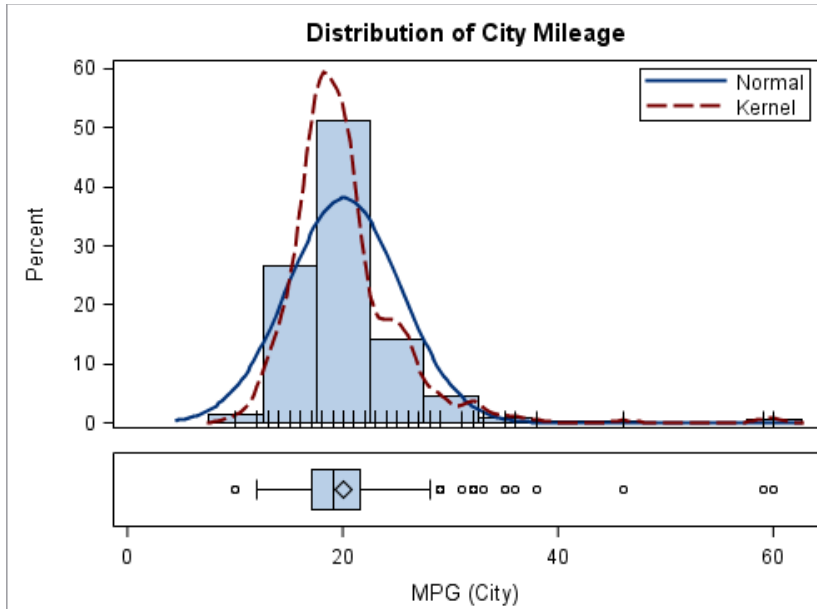
The graph includes the following features:

- a histogram of the MPG_CITY variable from the Sashelp.Cars data set
- overlaid normal density curve and kernel density curve
- an overlaid fringe plot showing individual observations
- an inset legend
- a separate horizontal box plot
- a common external X axis
- a graph title

Create Your Graph

The following sections walk you through the steps of creating your graph.

Create a Histogram from the Graph Gallery

The Graph Gallery is displayed in the work area as shown in Figure 3.2. For graphs that are created from the Graph Gallery, placeholder data is assigned to the graph.

Note: If the Graph Gallery is not displayed, select **View ▶ Graph Gallery**.

To create the histogram:

1. On the Basic tab of the Graph Gallery, select the **Histogram** icon.
2. Click **OK**.

You can also double-click the **Histogram** icon. For future steps, this book uses the double-click action when it is available.

Figure 3.2 Graph Gallery with Highlighted Histogram and Basic Tab
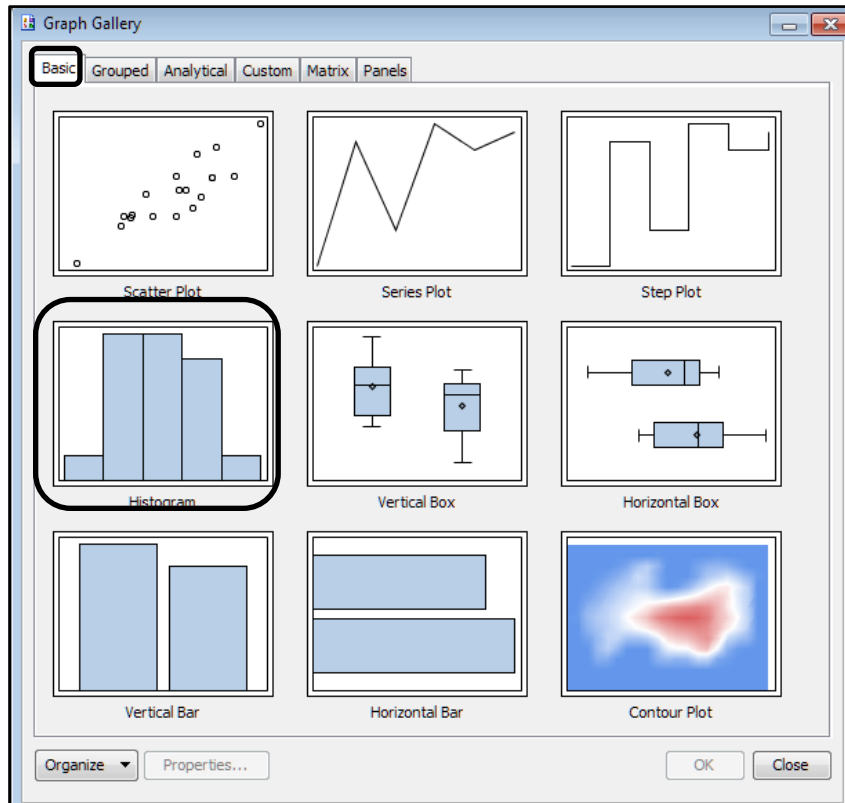
A graph window is displayed that includes a histogram plot. The **Assign Data** dialog box for the histogram appears, showing the placeholder data assignments.

Figure 3.3 Placeholder Data in the Assign Data Dialog Box
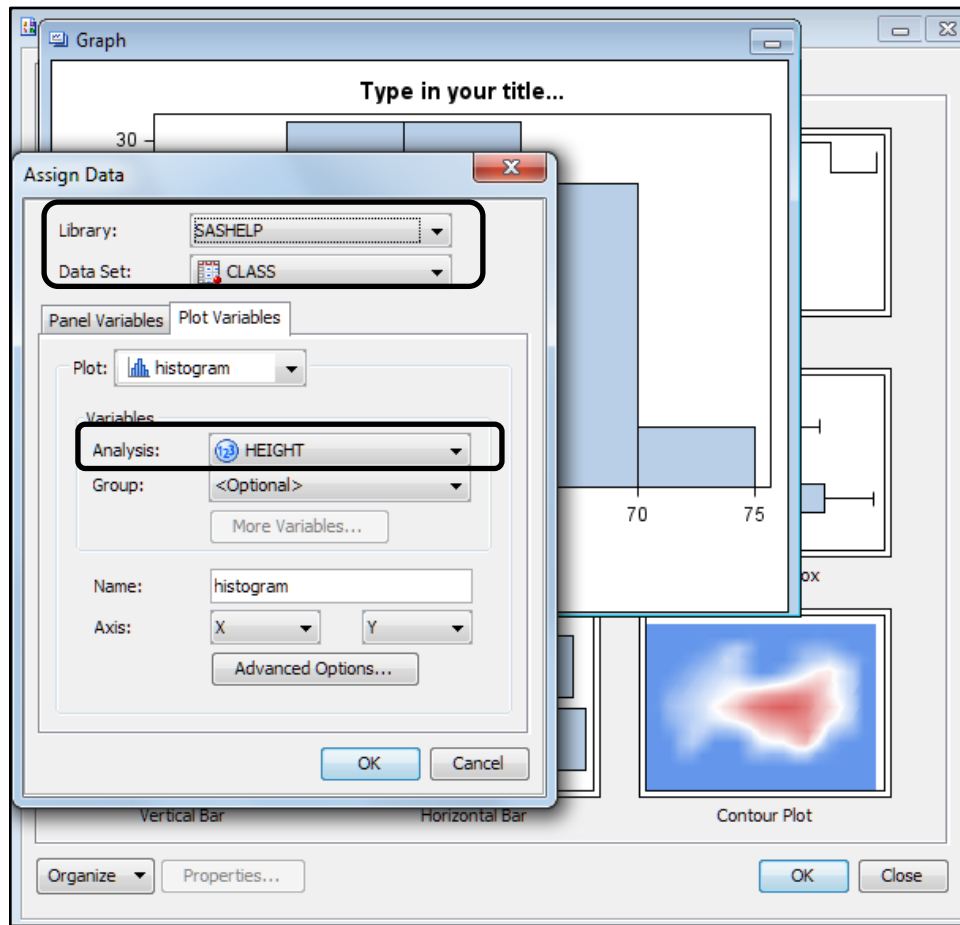
Note: Do not click **OK** in the **Assign Data** dialog box. You will change the data assignments in the next step.

The placeholder data enables the designer to show a visual draft of the type of plot that has been included in the graph. In this case, the designer uses the HEIGHT variable from the Sashelp.Class data set. Using this placeholder data, the designer creates the histogram.

The **Assign Data** dialog box is discussed in more detail in Chapter 4. For now, it's sufficient to know that you use this dialog box to change the data assigned to the plot and to the analysis variable.

Assign Data to the Histogram

Assign the MPG_CITY variable from the Sashelp.Cars data set to the histogram.

1. In the **Assign Data** dialog box, select **CARS** for **Data Set**.

   The previous variable, HEIGHT, is not available in the new Sashelp.Cars data set, so the **Analysis** variable setting has been cleared. Because an analysis variable is required, the settings for the histogram are not complete. As a result, the plot identifier is shown in red and the **OK** button is not available.
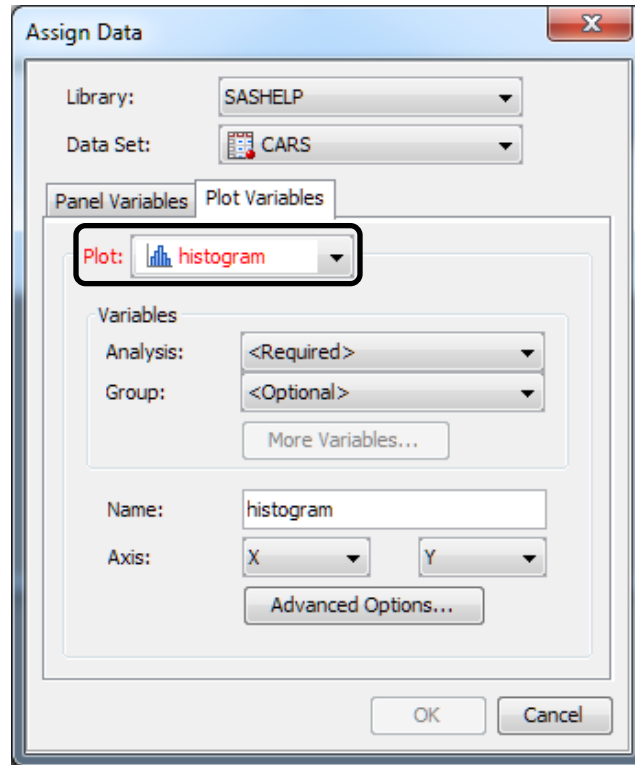
2. Select **MPG_CITY** for **Analysis**.
3. Click **OK**.

The designer creates the graph with a histogram and a placeholder title and footnote.

Figure 3.4 Histogram with Title and Footnote Placeholders
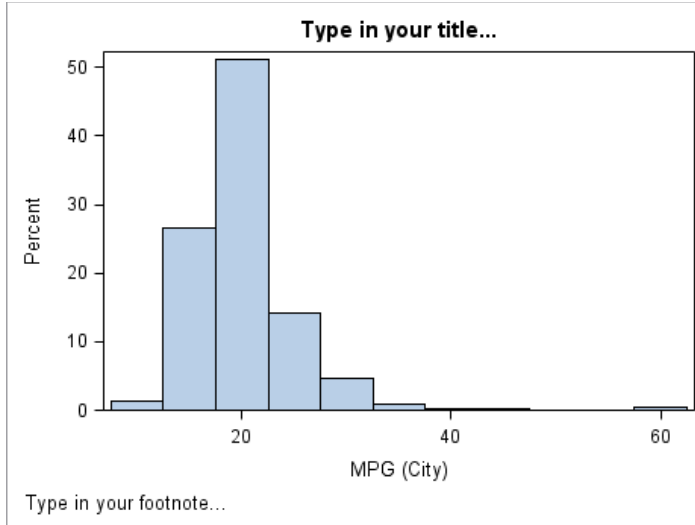
> **Note**: Your graph might be a different size from the one shown here. For information about the graph sizes used in this book, see "Graph Size" in **About This Book**.

Change the Title and Remove the Footnote

Replace the placeholder title and remove the footnote.
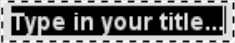
1. Double-click on the placeholder title ("Type in your title"). The placeholder text is highlighted:

2. In the text box, enter `Distribution of City Mileage`.

3. To remove the footnote, right-click on the placeholder footnote ("Type in your footnote"), and select **Remove Footnote**.

    Your changes are reflected in the graph.

Figure 3.5 Graph with New Title
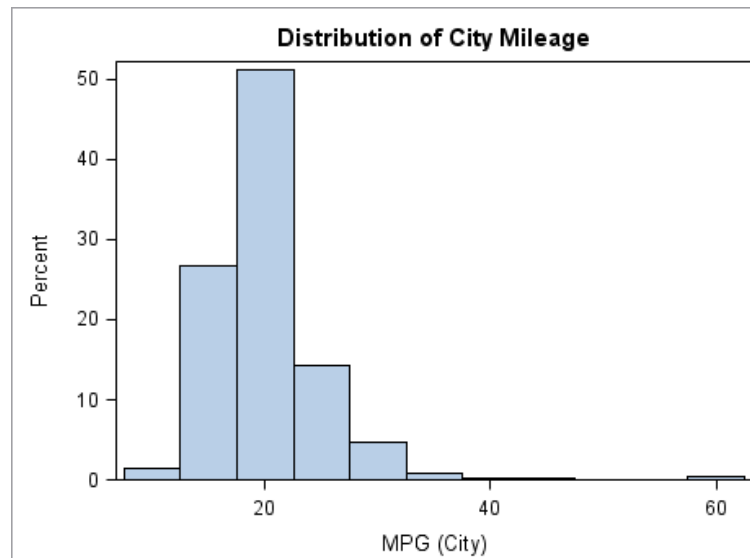
Other actions related to titles and footnotes include the following:

- To set title properties, right-click on the title, and select **Title Properties**. The **Text Properties** dialog box is displayed from which you can customize the visual properties of the title, such as font, color, and so on. You can do the same for the footnote.
- You can insert more titles and footnotes using the **Insert** menu.

Add Plots to the Graph

When a graph is displayed in the work area, the Elements pane is active. You can easily add a plot from the Elements pane to the graph cell, assuming the plot being added is compatible with the existing plot and its axes settings in the graph cell. Plot compatibility is discussed in more detail in Chapter 4.

For this example, let's add a normal density curve, a kernel density curve, and a fringe plot to the graph. These plot types are compatible with the histogram and can be added to this graph.

Figure 3.6 Normal Plot in the Elements Pane
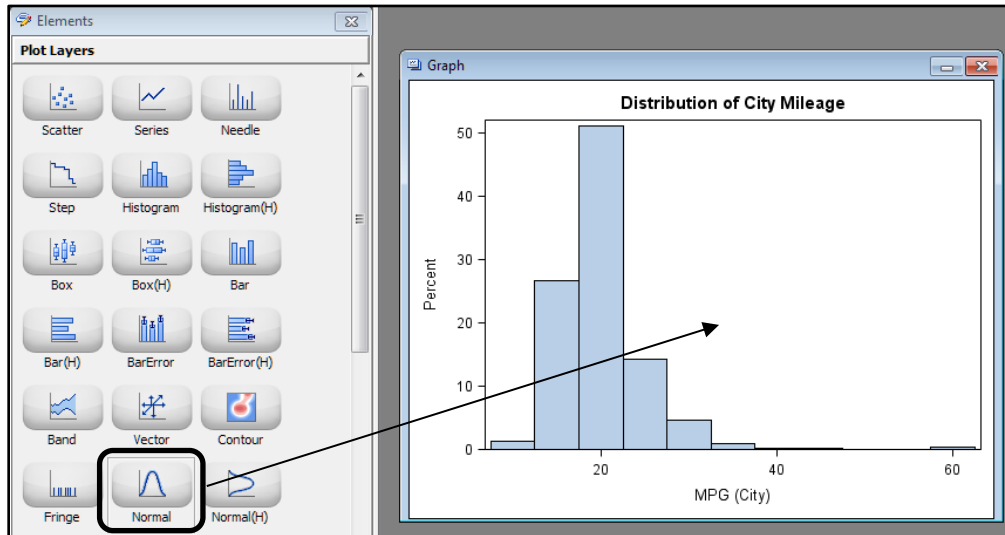
1. In the Elements pane, click the **Normal** icon as shown in Figure 3.6. Drag and drop the **Normal** icon onto the graph. The **Assign Data** dialog box for a normal density curve is displayed.
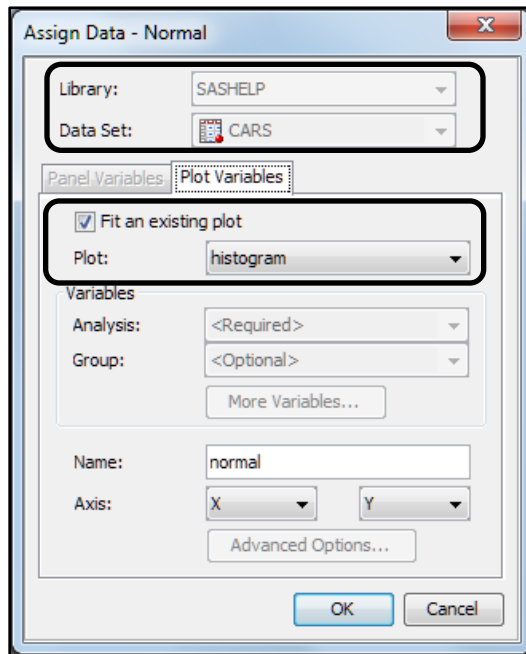
Note the following in the dialog box:

-   **Library** and **Data Set** are not available. All the plots in a single cell of a graph must use data from the same data set.

- ○ **Fit an existing plot** is selected by default. With this option, the normal density curve uses the same data settings as the histogram. There is only one plot currently in the graph—the histogram. If there were more plots, you would have a choice of which plot to use for the fit.

- ○ Because **Fit an existing plot** is selected, **Analysis** is not available. The newly added plot must use the same data as the histogram. In this example, the normal density curve is fitted with the same analysis variable as the histogram.

2. Click **OK**. The normal density curve is added to the graph.
3. In the Elements pane, click the **Kernel** (kernel density curve) icon. Drag and drop the **Kernel** icon onto the graph.
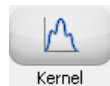
The **Assign Data** dialog box for a kernel density curve is displayed.

This dialog box is similar to the dialog box for the normal density curve. **Fit an existing plot** is selected by default.

4. Keep the default selections and click **OK**. The kernel density curve is added to the graph.
5. In the Elements pane, click the **Fringe** icon. Drag and drop the **Fringe** icon onto the graph.

The **Assign Data** dialog box for a fringe plot is displayed.

6. Select **MPG_CITY** for **X**.
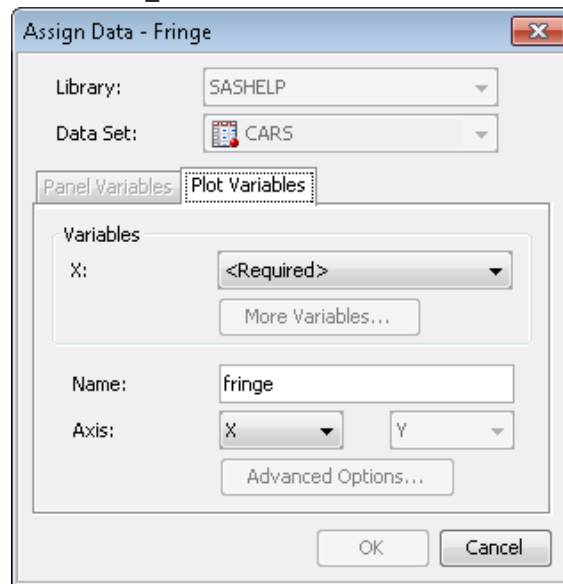
4. Click **OK**.

The graph is updated with the new plots.

Figure 3.7 Graph with Normal Density Curve, Kernel Density Curve, and Fringe Plot
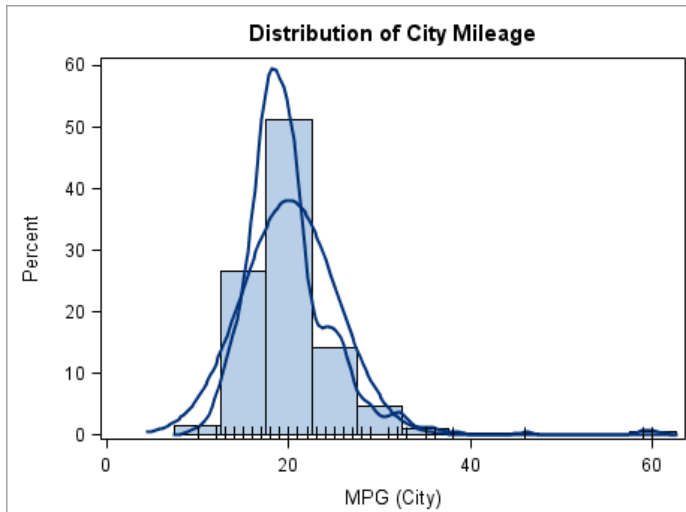
Set Plot Properties

Figure 3.7 shows the distribution of city mileage for all cars in the Sashelp.Cars data set using a histogram, two density curves, and a fringe plot. Because the density curves have the same visual properties, it can be difficult to tell which one is normal and which one is kernel.

To make them clearer, you can modify the visual properties of the kernel density curve. Select the kernel density curve, and then change the properties.

1. In the graph, select the kernel density curve. (The kernel density curve is the taller of the two curves.)
    The kernel density curve is selected and in bold. The other plots in the cell are dimmed.
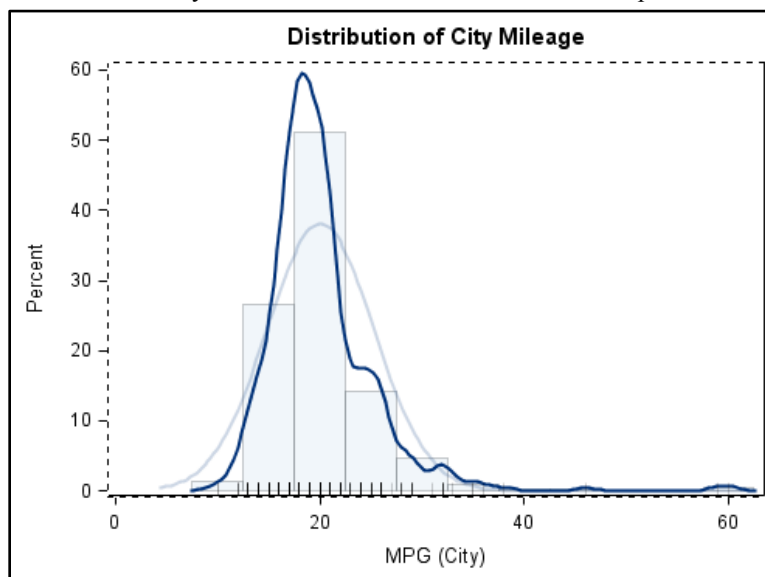
2. Right-click on the curve, and then select **Plot Properties**. The **Cell Properties** dialog box is displayed with the Plots tab selected.

3. Make sure that **kernel** is selected as **Plot**. If not, select **kernel**.

    The style element currently assigned to the kernel density curve is GraphFit, as shown in **Style Element**. (For more information about style elements, see "Visual Properties of a Graph" in Chapter 4.)
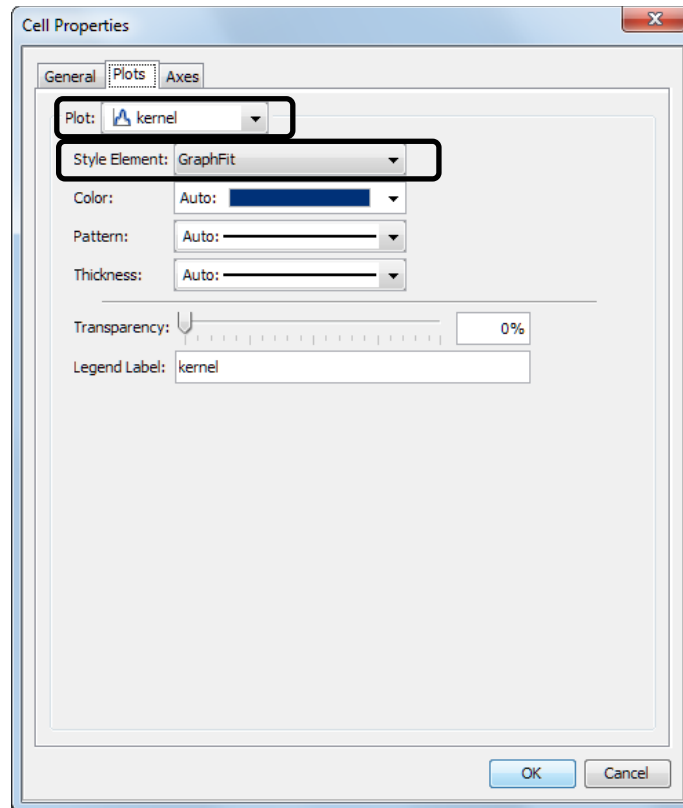
4. For **Style Element**, select **GraphFit2**.

    When you change the style element, the plot's attributes, such as color and pattern, are automatically changed to reflect the new style element.
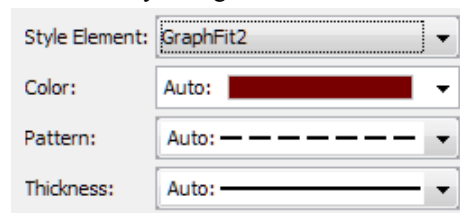
5. Click **OK**.

The graph is updated to reflect your changes.

Figure 3.8 Modified Appearance for the Kernel Density Curve
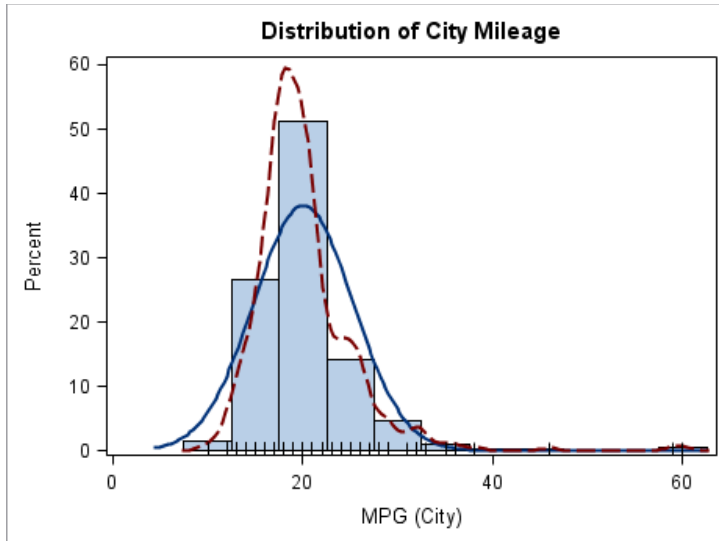
Add and Modify a Legend

In Figure 3.8, the visual properties of the two density curves are now distinct, but it is still difficult to see which one is the normal density curve and which one is the kernel density curve. To be able to identify the curves, add a legend.

The designer provides two types of legends:

Cell legend
is placed inside the data area of a cell. It is available from the Element pane's Insets panel. By default, this legend contains information for all the plots in that cell. It contains entries for only the plots in that cell.

Global legend
is placed outside the cells. It contains entries from all the plots in the entire graph, including multi-cell graphs. A global legend can be added to the graph by selecting **Insert ►Global Legend** or by clicking the global legend 🔲 toolbar button.

For this example, add a cell legend to the cell in the empty space in the upper right corner. Then, modify the legend to contain only the entries for the normal density curve and the kernel density curve.

1. In the Insets panel at the bottom of the Elements pane, click the **Discrete Legend** icon.

2. Drag and drop this icon onto the upper right corner of the cell.

   A legend that contains all the plots in the cell is added to the graph. However, it is unnecessary to show all the plots in the legend because some of that information is obvious. In this example, the histogram and the fringe plot are easily identified, so remove those entries from the legend.
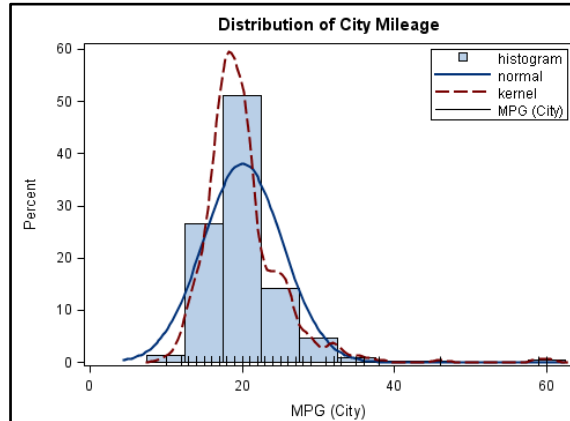
3. Right-click on the legend, and select **Legend Contents**.

   The **Legend Contents** dialog box is displayed. Initially, check boxes for all the plots in the cell are selected.
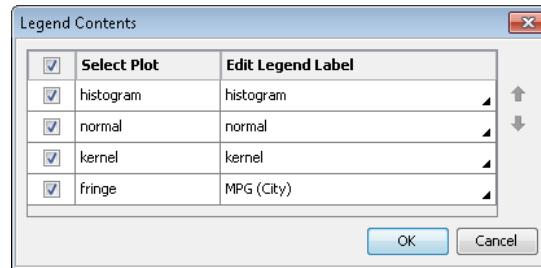
4. Clear the check boxes for **histogram** and **fringe**.

   You can customize the label that is displayed beside the chicklet in the legend. In this example, capitalize the normal density curve and kernel density curve legend labels.

5. In the **Edit Legend Label** column, double-click **normal**. Replace the lowercase **n** with a capital **N**.

6. Double-click **kernel**, and replace the lowercase **k** with a capital **K**. The **Legend Contents** dialog box should now resemble the following:
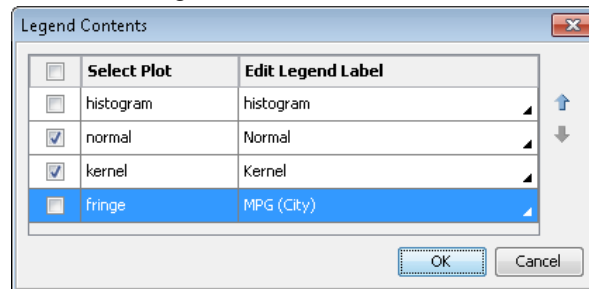
7. Click **OK**.

The graph is updated to reflect your changes.

Figure 3.9 Graph with a Cell Legend in the Upper Right Corner
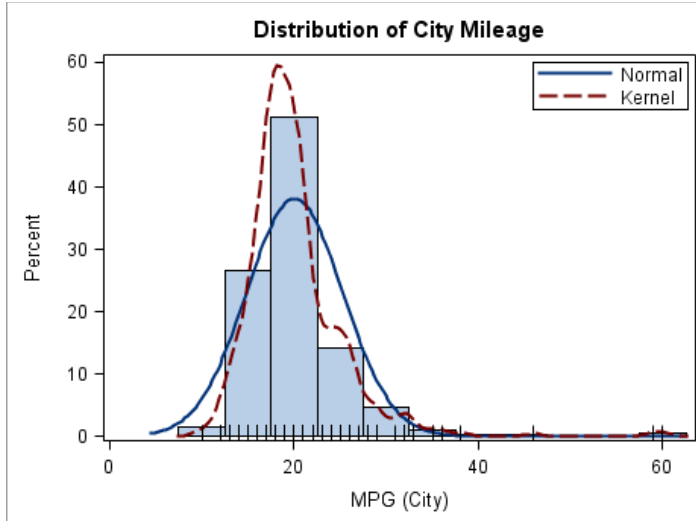
Add a Row to the Graph

Now, let's add a horizontal box plot to the graph in a separate cell below the histogram. To do this, first you have to add a new row to the graph.

To add a new row, right-click anywhere in the plot area, and select **Add Row**.

The graph area is split into two rows of equal height. You now have a graph with two rows; each row has one cell.

> **Tip**: You can add a column by right-clicking and selecting **Add Column**. Cells are always added in full rows or full columns to create a regular grid. After adding the row for this example, if you then add a column, the graph will contain four cells.

The new cell is empty except for the text "drop a plot here." You can now populate this cell with plots and insets.

Figure 3.10 Graph with Two Rows
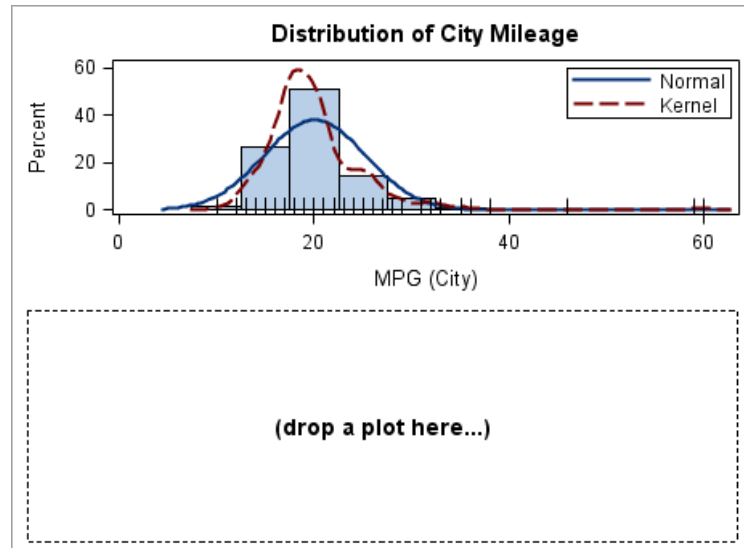
Add a Horizontal Box Plot to the Empty Cell

1. In the Elements pane, click on the **Box(H)** icon, and drag and drop the icon onto the bottom cell of the graph.

    Box(H)
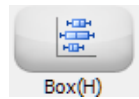
2. The **Assign Data** dialog box for the horizontal box plot is displayed.

    Library and **Data Set** values are based on the previous settings for the graph. You can keep those settings for the new plot. However, because this is a separate cell, you can select a different library and data set. The requirement about using the same data set applies only to plots in the same cell.
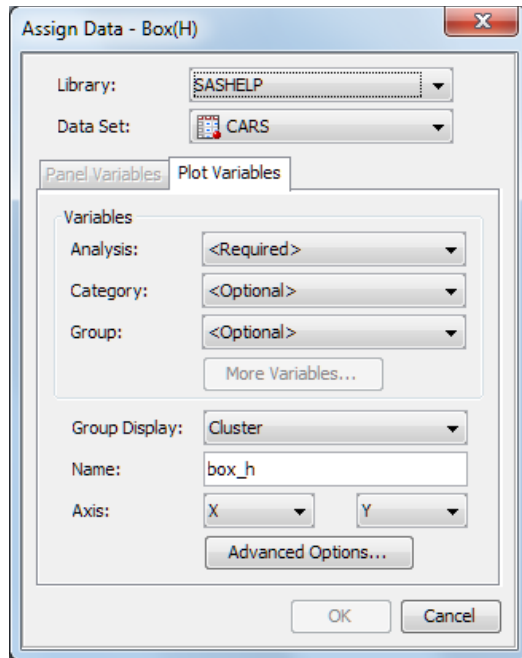
3. For **Analysis**, select **MPG_CITY**.
    For a box plot, the analysis variable is required. **Category** can be set to create box plots by category, but this is not required.
4. Select **OK**.

The box plot is added to the graph.

Figure 3.11 Graph with a Box Plot in the Bottom Row
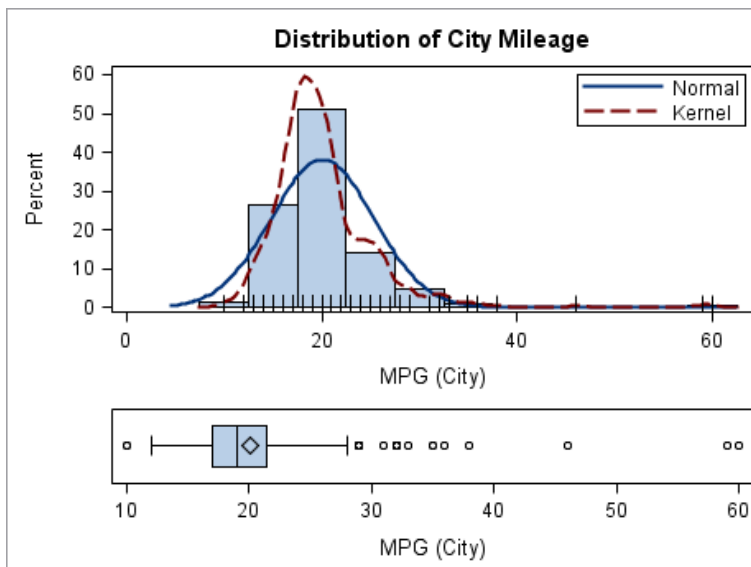
Starting with SAS 9.4, the height of the new cell is automatically reduced to fit the single box plot.

Use a Common X Axis

In Figure 3.11, each cell has its own X axis. The two axes are independent of each other, and their data ranges do not match. In this case, you want the two axes to match. Also, you don't need two separate axes because they take up valuable space in the graph and can be misleading.

To use a common X axis for both rows, right-click on one of the X axis areas, and select **Common Column Axis**.

A common column axis is created for all the cells in the column (two cells in this example). It is displayed at the bottom. The axis range for the common column axis is the union of the ranges for each cell in the column. All plots in each cell in the column are drawn correctly scaled to this new common column axis.

Figure 3.12 Graph with a Common X Axis
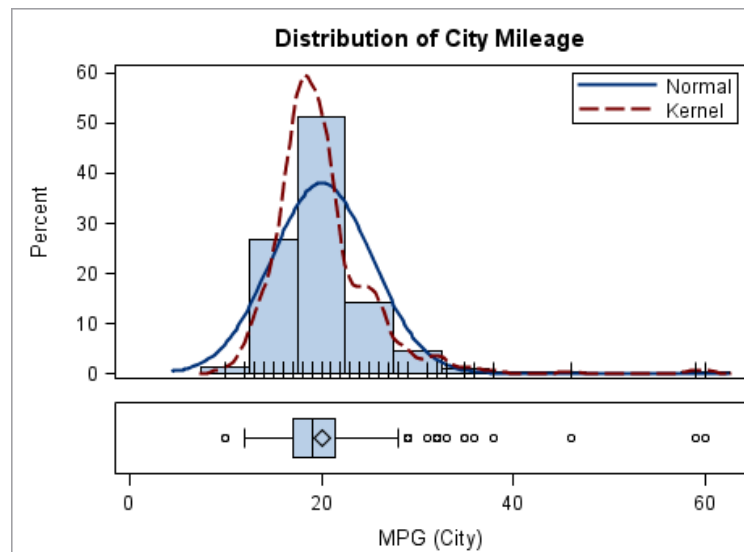
The two rows resize to fit their contents. The bottom row is shorter than the top row.

Note: In early releases of the designer, the rows are not automatically resized. If that is the case for you, you can change the height manually.

Position the cursor between the upper and lower row of the graph. A dashed line appears between the rows, and the cursor changes to a two-headed arrow ↕.

Click and drag the dashed line downward to reduce the height of the bottom row.

View the GTL Code

The designer is an excellent tool for learning the Graph Template Language (GTL). The designer generates GTL code as you create a graph.

To view the generated GTL code, select **View ► Code**.

The code window is displayed in the work area for the active graph.

For simplicity, the following figure shows the code window for the state of the graph when it contained only a histogram, a normal density curve, and a title.

Figure 3.13 GTL Code for the Histogram and Normal Density Curve
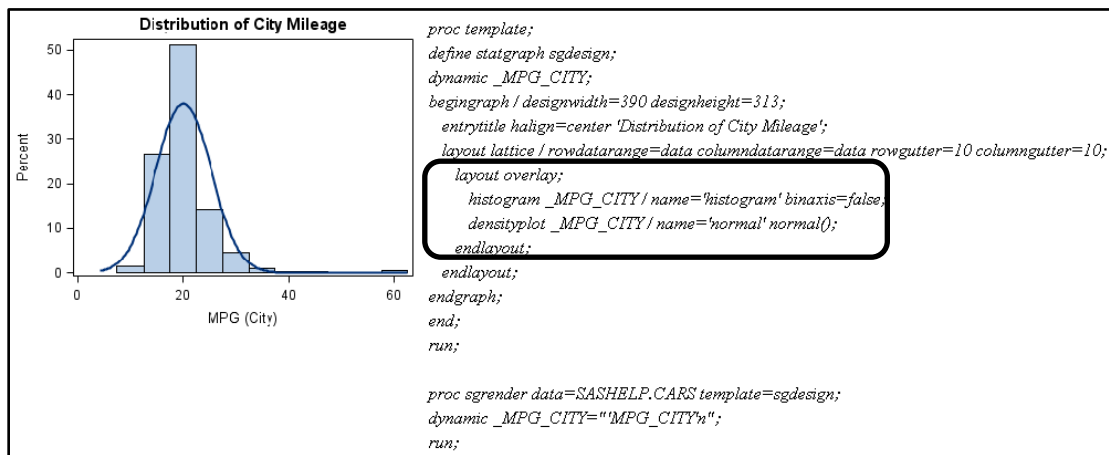
Figure 3.13 shows the GTL code needed to create this graph using the TEMPLATE procedure and the SGRENDER procedure. The boxed section shows the layout overlay block containing the histogram and normal density curve.

You can leave the code window displayed and view the changes to the code as you make changes to the graph.

The code window is Read-only. You can save the code as a SAS file, or you can copy and paste the code into SAS and run the program.

Copy and Paste the Graph to Another Application

At any stage, you can copy an image of the graph to the clipboard. Then, you can immediately paste it into a Word document, an email message, a PowerPoint presentation, or some other application that supports the paste feature.

1. Select **Edit ► Copy**. An image of the active graph is copied to the clipboard.
2. Paste the image into an application by using the application's paste command, such as Ctrl-V.

Save the Graph to a File

Save the Graph as a SAS ODS Graphics Designer File

You can save the graph in the SAS ODS Graphics Designer (SGD) format, which is a metafile format recognized by the designer. These files include the GTL code and other relevant information so that the designer can re-open the graph later for further modification.

1. Select **File ► Save As**.
2. In the **Save** dialog box, select the **SGD** file type.
3. Provide a name for the file, and click **Save**.

You can open a previously saved SGD file by selecting **File ► Open**, and then selecting the file that you want to open. Any SGD files in the selected folder are represented using an icon for the graph.

> **Note**: You use this SGD file later in the book, so be sure to complete the previous step.

Save the Graph as an Image File

You can save a copy of the graph to the file system as an image file. The designer supports multiple industry-standard image formats such as BMP, EMF, GIF, JPG, PDF, PNG, PS, SVG, and TIF. For the PNG format, you can specify a DPI to get higher resolution files.

1. Select **File ► Save As**.
2. In the **Save** dialog box, you can do the following:
   - specify the filename and folder into which to save the file
   - select a file type
   - select image resolution in DPI for PNG files

Save the Graph in the Graph Gallery

After you have designed a graph, you can add it to the Graph Gallery. You can then open and reuse the graph like you would any graph in the gallery.

Although graphs cannot be saved on the first six tabs of the gallery, you can add new tabs to the gallery.

To save the graph to the gallery:

1. Select **File ▶ Save in Graph Gallery**. The Save in Graph Gallery dialog box is displayed.
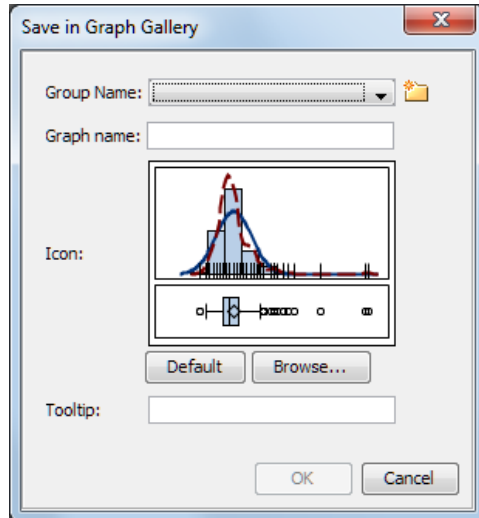
2. For **Group Name**, select the name of the group into which you want to add the graph. Each group corresponds to a tab in the gallery.

   Group Name contains the names of groups that have been created at your site. It does not contain the names of the default groups. If no groups have been created at your site, or if you want to create your own group, do the following:

   a. Select the New ⬚ icon.

   b. In the Create New Group dialog box, specify a name for the group. For this example, specify **MyGraphs**, and click **OK**.

      The new group appears in **Group Name** in the Save in Graph Gallery dialog box.

3. For **Graph name**, enter **Histogram**.

4. (Optional) The designer creates a default graph icon for the generated graph. If you want to replace the default icon with an icon from your file system, click **Browse**, and select a different icon.

5. (Optional) For **Tooltip**, you can provide a description that is displayed as a tooltip for your new graph.

6. Click **OK**.

The graph is saved in the Graph Gallery for future use. Your gallery might look something like this:

Figure 3.14 Graph Gallery with the MyGraphs Tab
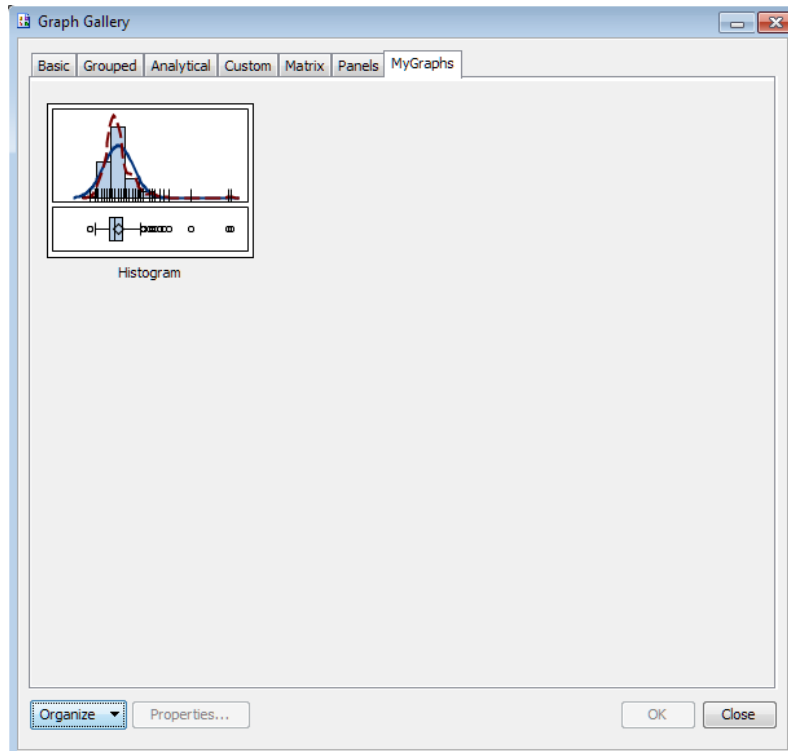
Run the Graph in Batch Mode

Graphs saved as SGD files can be run in the SAS windowing environment or in a batch session by using the SGDESIGN procedure.

```
proc sgdesign sgd='c:\histogram.sgd';
run;
```

In the code, replace *c:\histogram.sgd* with the name and location of the SGD file that you saved.

The graph is run using the definition in the SGD file and the original data set used to create the graph. The libref and data set should be available in the SAS session.

SGD graphs can be run with different data as long as all the required variables exist in the new data set. You can specify different data by using the DATA= option in PROC SGDESIGN. This topic is covered in more detail in Chapter 7.

Clinical Graphs Using the SAS 9.4 SGPLOT Procedure

Sanjay Matange

Excerpt from *Clinical Graphs Using SAS*

The previous chapters of this book covered the domain of visual data analytics, with a focus on big data. We also looked at creating graphs using SAS ODS Graphics Designer with no coding needed. But often after the visual exploration and analysis of data, you need to convey the results of your analyses to your clients or to regulatory agencies, such as FDA, that have stringent requirements.

One such area is in the reporting of the analysis of safety data for clinical trials or for drug safety. The pharmaceutical industry spends billions of dollars bringing life-saving drugs to the communities. The process requires extensive safety and efficacy testing of the drugs before they are approved. Results have to be shared with other researchers and the regulators using sophisticated graphical techniques.

The tools in the SAS ODS Graphics Designer enable you to deliver your analytic results to your consumers. These tools include Survival Plots, Forest Plots, and Adverse Event Plots. This chapter will introduce you to the procedures that you can use to create such graphs. A moderate level of programming knowledge is required.

Sanjay Matange is Research & Development Director in the SAS Data Visualization Division, where he is responsible for the development and support of SAS ODS Graphics software. This includes the Graph Template Language (GTL), Statistical Graphics (SG) procedures, SAS ODS Graphics Designer, and other related graphics applications. Sanjay has been with SAS for over 25 years. He is coauthor of two patents and author of four SAS Press books.

support.sas.com/matange

Chapter 4: Clinical Graphs Using the SAS 9.4 SGPLOT Procedure

Clinical graphs often display the data in one cell along with derived statistics and other details that aid in the decoding of the information in the graph. Most of these single-cell graphs can be created using the SGPLOT procedure.

With SAS 9.4, the SGPLOT procedure supports some new and useful features that simplify the creation of such graphs. These include the following new statements and features:

- **XAXISTABLE and YAXISTABLE.** These two statements support axis tables along the x- and y-axes. These statements can be used to draw "At-Risk" tables along the X-axis, or study names and statistic values along the Y-axis. Rows and columns of textual data can be displayed inside the data area or outside.
- **TEXT plot.** This statement displays a text string from a column at the specified location. It replaces the need for using a SCATTER plot statement with the MARKERCHAR option. Because a text plot draws only text strings, other features are available for this function, including control of offsets that might be driven by the text values.
- **POLYGON plot.** This statement displays polygons in the graph based on the columns in the data set. This is useful in drawing ranges in the graph for various levels, including complex regions in graphs for device evaluation like the Clark Error Grid.

The goal in this chapter is to cover in detail the creation of some commonly used clinical graphs using SAS 9.4. The chapter will provide not only code that you can use directly for such graphs, but will also provide ideas on how you can use or combine plot statements to create your own custom graph.

The SG Annotate facility features are also available for you to use in cases where the result cannot be achieved using plot layers. SG Annotate was used extensively in Chapter 3 to create the clinical graphs. See Section 2.9 for an introduction to this feature.

4.1 Box Plot of QTc Change from Baseline

This graph displays the distribution of QTc change from baseline by week and treatment for all subjects in a study. The x-axis has a linear scale. The "Subjects At-Risk" values are displayed by treatment at the right location along the time axis.

4.1.1 Box Plot of QTc Change from Baseline

For the graph in Figure 4.1.1, we will use a box plot to display the distribution of QTc change from baseline by week and treatment on a linear x-axis. The "Subjects At-Risk" table is shown in the traditional arrangement at the bottom of the graph using the XAXISTABLE statement.

Figure 4.1.1 – Graph of QTc Change from Baseline with the Subjects Table at the Bottom
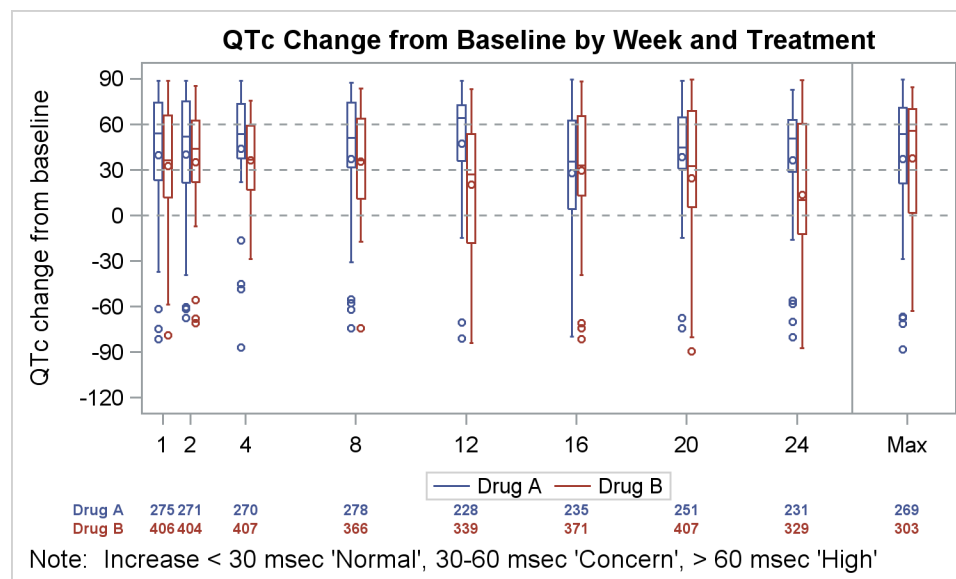

```
title 'QTc Change from Baseline by Week and Treatment';
footnote j=l "Note:  Increase < 30 msec 'Normal', "
            "30-60 msec 'Concern', > 60 msec 'High' ";
proc sgplot data=QTcData;
  format week qtcweek.;
  vbox qtc / category=week group=drug groupdisplay=cluster nofill;
  xaxistable risk / class=drug colorgroup=drug;
  refline 26 / axis=x;
  refline 0 30 60 / axis=y lineattrs=(pattern=shortdash);
  xaxis type=linear values=(1 2 4 8 12 16 20 24 28) max=29
        display=(nolabel);
  yaxis label='QTc change from baseline' values=(-120 to 90 by 30);
  keylegend / title='' linelength=20;
run;
```

Normally, a box plot treats the category variable as discrete, which would have placed all the tick values on the x-axis at equally spaced intervals. However, in this case the values on the x-axis represent days from start of study, and we want to place the data at the correctly scaled distance along the x-axis. This can be done be explicitly setting TYPE=LINEAR on the x-axis. Now, each box is placed at the scaled location along the x-axis.

The box plot is classified by treatment, which has two values "Drug A" and "Drug B". The boxes are sized by the smallest interval along the x-axis. In this case, it is one day at the start of the study. Hence, the effective midpoint spacing is set by that interval, and all boxes are drawn to fit in this space.

The box plot uses the GROUPDISPLAY=CLUSTER option to place the groups side by side. We have used the NOFILL option to draw empty boxes.

The table of the subjects at risk is displayed using the XAXISTABLE statement, showing risk values by week and drug. The optional X role is not specified, so the table uses the X role that is active; in this case, it is from the VBOX statement. The option LOCATION=OUTSIDE is used to display the risk values outside the data area at the default bottom position.

The XAXISTABLE is classified by treatment by setting CLASS=DRUG. Now, the values for risk are displayed in separate rows by drug. The value of the classifier DRUG is shown in the row label on the left of the data. The option COLORGROUP=DRUG is used to color the risk values by drug for easier association. Display attributes such as font size and font weight are set for both the risk values and labels using the appropriate options:

```
xaxistable risk / class=drug colorgroup=drug
                  valueattrs=(size=6 weight=bold)
                  labelattrs=(size=6 weight=bold);
```

Reference lines are placed on the y-axis at y= 0, 30, and 60 to represent the levels of concern. A reference line is also placed on the x-axis at x=26 to separate MAX value.

The axis tick value "Max" has a value of x=28, and a format is used to display the tick value. The tick values displayed on the x-axis are determined by the VALUES option on the XAXIS statement, and the option MAX is set to 29 to allow an even display of the tick values.

The y-axis places the tick values from -120 to 90 by 30, and also sets the displayed axis range.

A legend is automatically added by the procedure because the box plot has a GROUP role. We have used the KEYLEGEND statement to customize some aspects of the legend. The legend title is removed, and the lengths of the line segments representing each classification value are shortened using the LINELENGTH option.

Normally, the procedure draws longer lines for each class value in the legend in order to represent the full line pattern. In this case, however, we are using the HTMLBlue style, which uses the attribute priority of color. So, most line styles used are solid, and a long line segment is not required.

Relevant details are shown in the code snippet above. For full details, see Program 4_1, available from the author's page at http://support.sas.com/matange.

4.1.2 Box Plot of QTc Change from Baseline with Inner Risk Table and Bands

The traditional graph commonly in use in the industry, as shown in Figure 4.1.1, shows the "At-Risk" table at the bottom of the graph, just above the footnote with other items in between. Such a layout places the risk data relatively far away from the graph. Even though the values are aligned with the data along the x-axis, the distance and intervening items like the legend and the axis items create a distraction.

Figure 4.1.2 – Graph of QTc Change from Baseline with Subjects Table inside
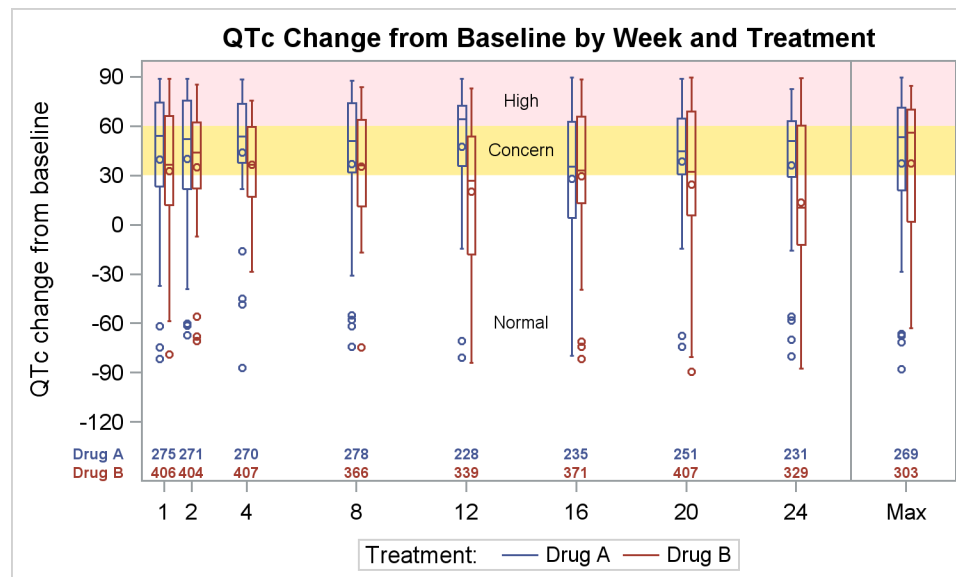
Graphs are easier to decode when relevant information is placed as close as possible, thus reducing the amount of eye movement needed to decode the graph. Following this principle, it would be more effective to place the risk information inside the graph area, closer to the graphical information. This arrangement is shown in the Figure 4.1.2. It was achieved by placing the XAXISTABLE with LOCATION=INSIDE, as shown in the code snippet below.

Another improvement would be to represent the levels of concern as colored bands with direct labels. This reduces the eye movement that is required to decode the information in the data. We can do this by using the BAND plot statements. A text plot is used to label each band with the level

of concern "Normal", "Concern", and "High". The columns needed are included in the data. The code snippet for inclusion of bands, band labels, and the inner risk table is shown below.

```
title 'QTc Change from Baseline by Week and Treatment';
proc sgplot data=QTcBand;
  format week qtcweek.;
  band x=wk lower=L0  upper=L30 / fill legendlabel='Normal' name='a'
       fillattrs=(color=white transparency=0.6) ;
  band x=wk lower=L30 upper=L60 / fill legendlabel='Concern' name='b'
       fillattrs=(color=gold transparency=0.6) ;
  band x=wk lower=L60 upper=L90 / fill legendlabel='High' name='c'
       fillattrs=(color=pink transparency=0.6);
  vbox qtc / category=week group=drug groupdisplay=cluster
       nofill name='d';
  text x=wk y=ylabel text=label / contributeoffsets=none;
  xaxistable risk / class=drug colorgroup=drug location=inside;
  refline 26 / axis=x;
  xaxis type=linear values=(1 2 4 8 12 16 20 24 28) valueshint
       min=1 max=29 display=(nolabel)
        colorbands=odd colorbandsattrs=(transparency=1);
  yaxis label='QTc change from baseline' values=(-120 to 90 by 30);
  keylegend 'a' / title='Treatment:' linelength=20;
run;
```

For graphs that are consumed in a color medium, this graph provides all the information in a compact form that is free of clutter. The levels of concern are color coded with direct labels, and risk values are moved closer to the rest of the data. For full details, see Program 4_1.

4.1.3 Box Plot of QTc Change from Baseline in Grayscale

The graph in Figure 4.1.3 is created for a grayscale medium using the Journal style, along with a few other customizations.

Figure 4.1.3 – Box Plot of QTc Change from Baseline in Grayscale
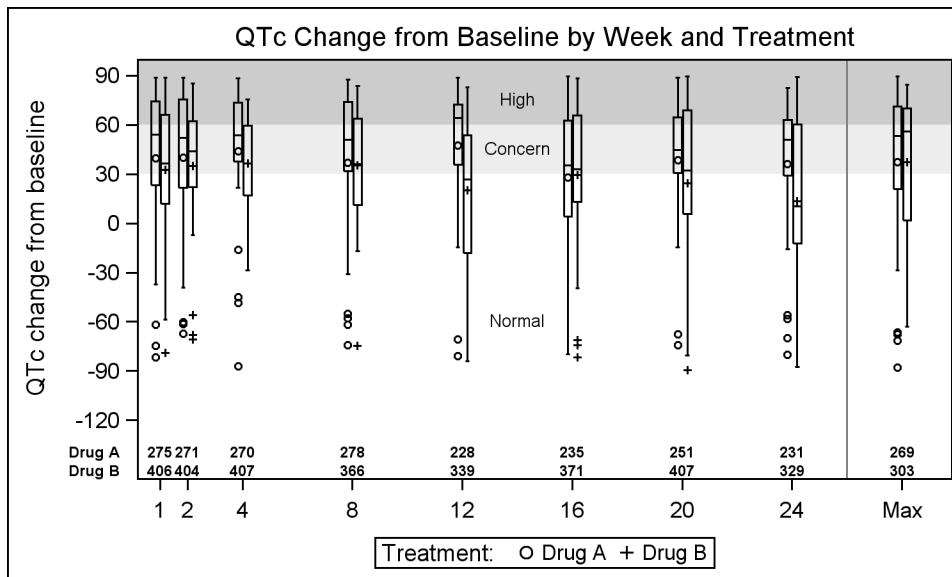

```
title 'QTc Change from Baseline by Week and Treatment';
proc sgplot data=QTcBand;
  format week qtcweek.;
  styleattrs datalinepatterns=(solid);
  band x=wk lower=L0  upper=L30 / fill legendlabel='Normal'
       fillattrs=(color=white transparency=0.6);
  band x=wk lower=L30 upper=L60 / fill  legendlabel='Concern'
       fillattrs=(color=lightgray transparency=0.6);
  band x=wk lower=L60 upper=L90 / fill legendlabel='High'
       fillattrs=(color=gray transparency=0.6) ;
  vbox qtc / category=week group=drug groupdisplay=cluster nofill;
  scatter x=wk y=QTc / group=drug name='a' nomissinggroup;
  text x=wk y=ylabel text=label / contributeoffsets=none;
  xaxistable risk / class=drug colorgroup=drug location=inside;
  refline 26 / axis=x;
  xaxis type=linear values=(1 2 4 8 12 16 20 24 28) valueshint
       min=1 max=29 display=(nolabel)
       colorbands=odd colorbandsattrs=(transparency=1);
  yaxis label='QTc change from baseline' values=(-120 to 90 by 30);
  keylegend 'a' / title='Treatment:' linelength=20;
run;
```

Box plots are represented in the legend using the display characteristics of the box. In this case, the boxes are not filled. Normally, when using grayscale, the line style for the second group is a

dashed line. To avoid drawing boxes with patterned lines, we have specified only one solid pattern for all groups in the STYLEATTRS statement. So, using lines in the legend will not be effective.

To distinguish the two groups, we would like to display markers in the legend. To do this, we added a scatter plot that plots QTc by Week and Drug, except that all these QTc values are missing. So, no markers are actually drawn in the plot, but the legend that is derived from the scatter plot displays the marker symbols. Relevant details are shown in the code snippet above. For full details, see Program 4_1.

4.2 Mean Change in QTc by Visit and Treatment

In this section we discuss creating the graph of the mean change in QTc by visit and treatment.

4.2.1 Mean Change in QTc by Visit and Treatment

The graph above displays the mean value of QTc change from baseline by week and treatment for all subjects in a study. The x-axis displays weeks on a numeric scale, with week 28 formatted to "LOCF". The mean values are clustered side by side.

Figure 4.2.1 – Mean Change in QTc by Visit and Treatment
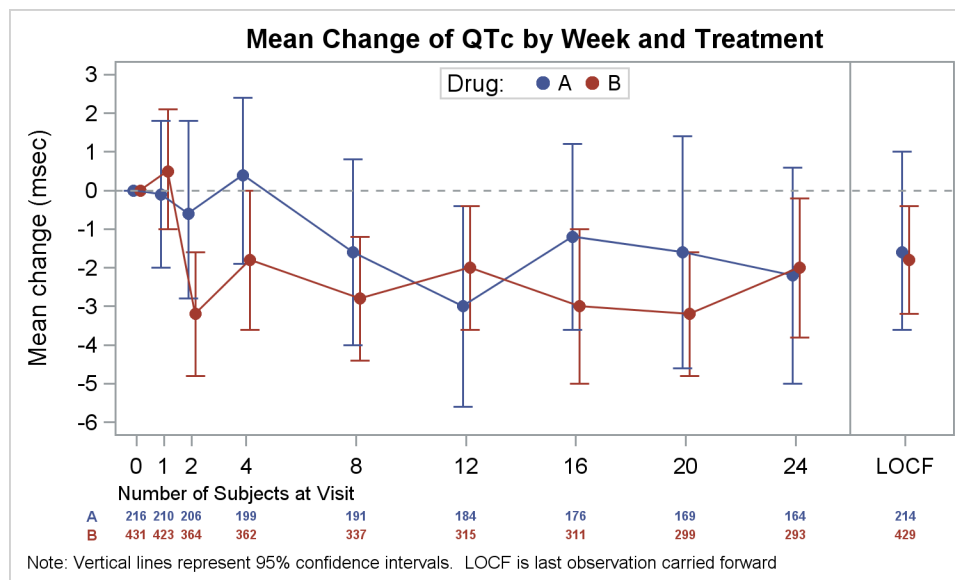

```
title 'Mean Change of QTc by Week and Treatment';
proc sgplot data=QTc_Mean_Group;
  format week qtcmean.;
  format n 3.0;
  scatter x=week y=mean / yerrorupper=high yerrorlower=low
          group=drug groupdisplay=cluster clusterwidth=0.5
          markerattrs=(size=7 symbol=circlefilled) name='a';
```

```
    series x=week y=mean2 / group=drug groupdisplay=cluster
           clusterwidth=0.5 lineattrs=(pattern=solid);
    xaxistable n / class=drug colorgroup=drug location=outside
           title='Number of Subjects at Visit' titleattrs=(size=8);
    refline 26 / axis=x;
    refline 0  / axis=y lineattrs=(pattern=shortdash);
    xaxis type=linear values=(0 1 2 4 8 12 16 20 24 28)
          max=29 valueshint;
    yaxis label='Mean change (msec)' values=(-6 to 3 by 1);
    keylegend 'a' / title='Drug: ' location=inside position=top;
run;
```

The overlaid series plot could have also used the same response variable "Mean", but then the last value on the x-axis, "LOCF", would have been joined to the previous one.

To avoid this, we have copied the values from the variable "Mean" to the variable "Mean2", with a missing value for the x=28. The series plot uses "Mean2" as the response variable, which excludes the last value to avoid connecting to the "LOCF" value.

Note, both the SCATTER and SERIES statements use GROUPDISPLAY=CLUSTER. This option spreads the position of each group value on the x-axis. CLUSTERWIDTH=0.5 is set to keep the clusters tight. This means that all the class values will be spread within 50% of the midpoint spacing. Since both statements use same setting for group display and cluster width, the lines and markers match for each group value.

An XAXISTABLE is used to display the "Number of Subjects" values at the bottom of the graph. The display variable is "N", and the x-axis variable is the same as the x variable for the primary plot – "Week". So, the optional X role does not need to be specified in the statement.

```
  xaxistable n / class=drug colorgroup=drug location=outside
         valueattrs=(size=5 weight=bold) labelattrs=(size=6 weight=bold)
         title='Number of Subjects at Visit' titleattrs=(size=8);
```

The graph is classified by "Drug". We have specified "Drug" for the CLASS role for the axis table. This causes the values for the two values of "Drug" to be displayed in separate rows. COLORGROUP is also set to "Drug", so the values are colored by "Drug".

The table of subjects is displayed at the bottom of the graph by setting LOCATION=OUTSIDE, which is also the default setting. The table has a title, which was set using the TITLE option. Text attributes for the values, labels, and title are specified using the appropriate options.

The Y reference line is drawn at y=0, and the X reference line is drawn at X=26, which acts like a separator for the "LOCF" value. A user-defined format is used to display "LOCF" for x=28.

The legend is generated by default by the procedure because group is in effect. But to prevent multiple items in the legend, we specify the KEYLEGEND statement with the name of only one statement. The legend is placed at the top center of the wall.

Relevant details are shown in the code snippet above. For full details, see Program 4_2.

4.2.2 Mean Change in QTc by Visit and Treatment with Inner Table of Subjects

In this graph, the table of subjects at visit is displayed above the x-axis. This makes it easier to understand the numbers because they are closer to the rest of the graph.

Figure 4.2.2 – Mean Change in QTc by Visit and Treatment
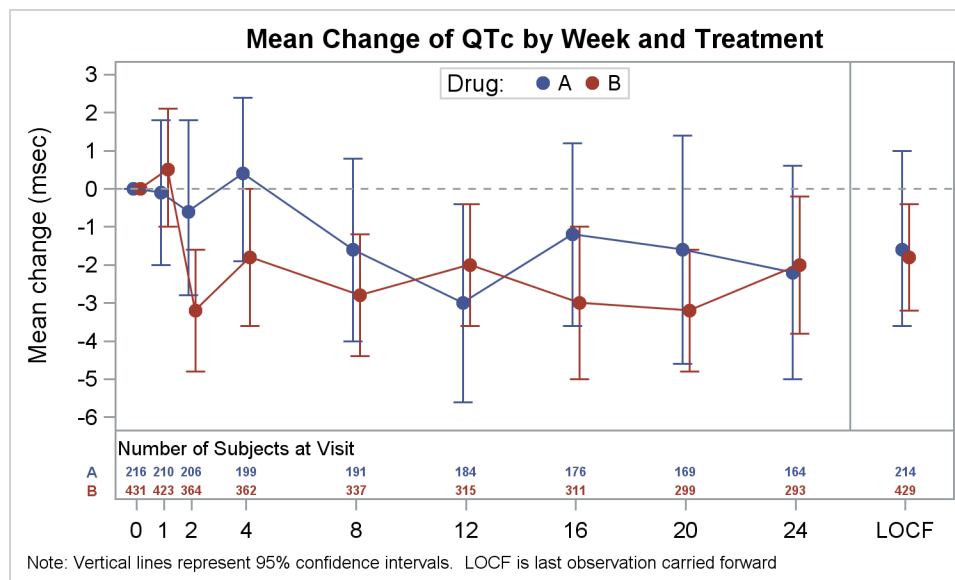

```
proc sgplot data=QTc_Mean_Group;
  format week qtcmean.;
  format n 3.0;
  scatter x=week y=mean / yerrorupper=high yerrorlower=low
          group=drug groupdisplay=cluster clusterwidth=0.5
          markerattrs=(size=7 symbol=circlefilled) name='a';
  series x=week y=mean2 / group=drug
         groupdisplay=cluster clusterwidth=0.5;
  xaxistable n / class=drug colorgroup=drug location=inside
             valueattrs=(size=5 weight=bold)
             labelattrs=(size=6 weight=bold) separator
             title='Number of Subjects at Visit' titleattrs=(size=8);
  refline 26 / axis=x;
  refline 0  / axis=y lineattrs=(pattern=shortdash);
  xaxis type=linear values=(0 1 2 4 8 12 16 20 24 28) max=29
        valueshint display=(nolabel);
  yaxis label='Mean change (msec)' values=(-6 to 3 by 1);
  keylegend 'a' / title='Drug: ' location=inside position=top;
run;
```

The graph above is mostly similar to 4.2.1, with the key difference of placing the "Subjects at Visit" inside the data area instead of at the bottom of the graph. This improves the readability of the graph.

The key difference in the code is the use of LOCATION=Inside for the XAXISTABLE. We also use the SEPARATOR option to draw the line above the table. A reference line is used to separate the "LOCF" value. Relevant details are shown in the code snippet above. For full details, see Program 4_2.

4.2.3 Mean Change in QTc by Visit and Treatment in Grayscale

The graph shown in Figure 4.2.3 shows the Mean Change in QTc graph in grayscale.

Figure 4.2.3 – Mean Change in QTc by Visit and Treatment in Grayscale
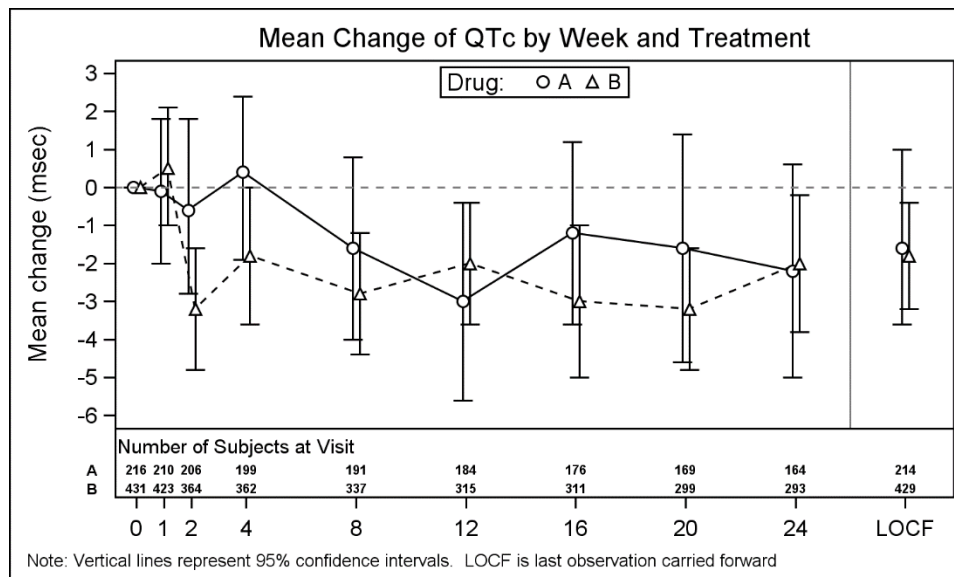

```
ods listing style=journal;
proc sgplot data=QTc_Mean_Group;
  styleattrs datasymbols=(circlefilled trianglefilled)
             datalinepatterns=(solid shortdash);
  format week qtcmean.;
  format n 3.0;
  series x=week y=mean2 / group=drug groupdisplay=cluster c
         clusterwidth=0.5;
  scatter x=week y=mean / yerrorupper=high yerrorlower=low
          group=drug name='a' groupdisplay=cluster
          clusterwidth=0.5 markerattrs=(size=7)
          filledoutlinedmarkers markerfillattrs=graphwalls;
  xaxistable n / class=drug colorgroup=drug location=inside
          title='Number of Subjects at Visit' separator;
  refline 26 / axis=x;
```

```
    refline 0  / axis=y lineattrs=(pattern=shortdash);
    xaxis type=linear values=(0 1 2 4 8 12 16 20 24 28) max=29 valueshint;
    yaxis label='Mean change (msec)' values=(-6 to 3 by 1);
    keylegend 'a' / title='Drug: ' location=inside position=top;
run;
```

To create an effective graph in grayscale, we have run the same graph as in 4.2.2 with ODS Style=JOURNAL. Also, we have used STYLEATTRS option to customize the group attributes.

Note the use of FILLEDOUTLINEDMARKERS. When we are using the filled markers that are specified here, the markers are drawn with fill and outline.
MARKERFILLATTRS=GRAPHWALLS is used. Relevant details are shown in the code snippet above. For full details, see Program 4_2.

4.3 Distribution of ASAT by Time and Treatment

The graphs below consist of three sections. The main body of the graph contains the display of ASAT by Week and Treatment in the middle. A table of subjects in the study by treatment is at the bottom, and the number of subjects with value > 2 by treatment is at the top of the graph.

4.3.1 Distribution of ASAT by Time and Treatment

The values of ASAT by week and treatment are displayed using a box plot. The x-axis type is linear.

Figure 4.3.1 – Distribution of ASAT by Time and Treatment
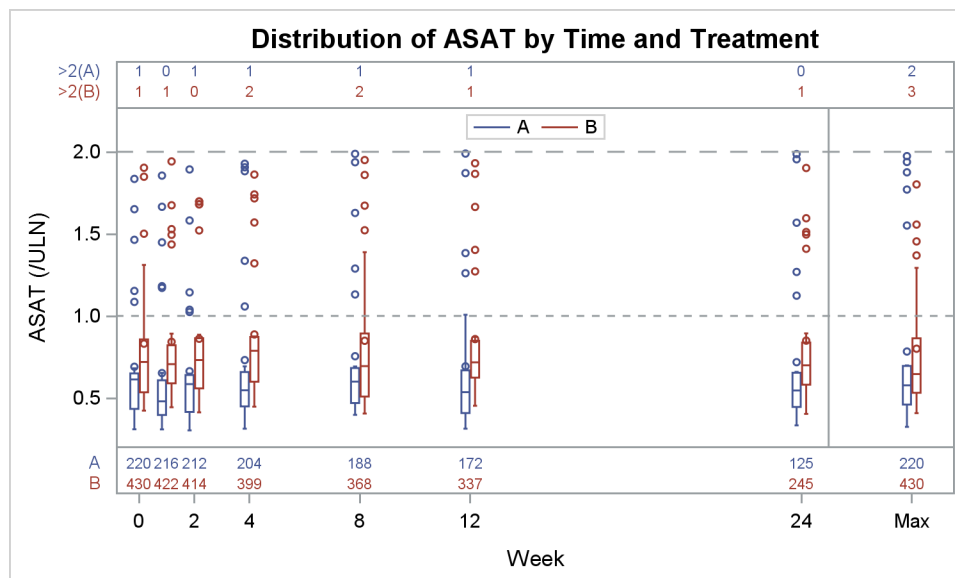
This graph is likely one of the most complex displays that can be created using the SGPLOT procedure. This graph displays the distribution of ASAT by treatment over time using a grouped

box plot on a linear x-axis. The visit values are scaled correctly on the time axis.   The smallest
interval between the visits determines the "effective" midpoint spacing used for adjacent placement
of the treatment values.

```
title 'Distribution of ASAT by Time and Treatment';
proc sgplot data=asat;
  vbox asat / category=week group=drug name='box' nofill;
  xaxistable gt2 / class=drugGT colorgroup=drugGT position=top
         location=inside separator valueattrs=(size=6)
         labelattrs=(size=7);
  xaxistable count / class=drug colorgroup=drug position=bottom
         location=inside separator valueattrs=(size=6)
         labelattrs=(size=7);
  refline 1 / lineattrs=(pattern=shortdash);
  refline 2 / lineattrs=(pattern=dash);
  refline 25 / axis=x;
  xaxis type=linear values=(0 2 4 8 12 24 28) offsetmax=0.05
       valueattrs=(size=7) labelattrs=(size=8);
  yaxis offsetmax=0.1 valueattrs=(size=7) labelattrs=(size=8);
  keylegend 'box' / location=inside position=top linelength=20;
run;
```

An XAXISTABLE statement is used to display the "Number of Subjects" values at the bottom of
the graph. A second XAXISTABLE at the top is used to display the count of values above 2.0 by
treatment.

4.3.2 Distribution of ASAT by Time and Treatment in Grayscale

The graph in Figure 4.3.2 is the same as above in grayscale. Markers are used in the legend for treatment.

Figure 4.3.2 – Distribution of ASAT by Time and Treatment in Grayscale
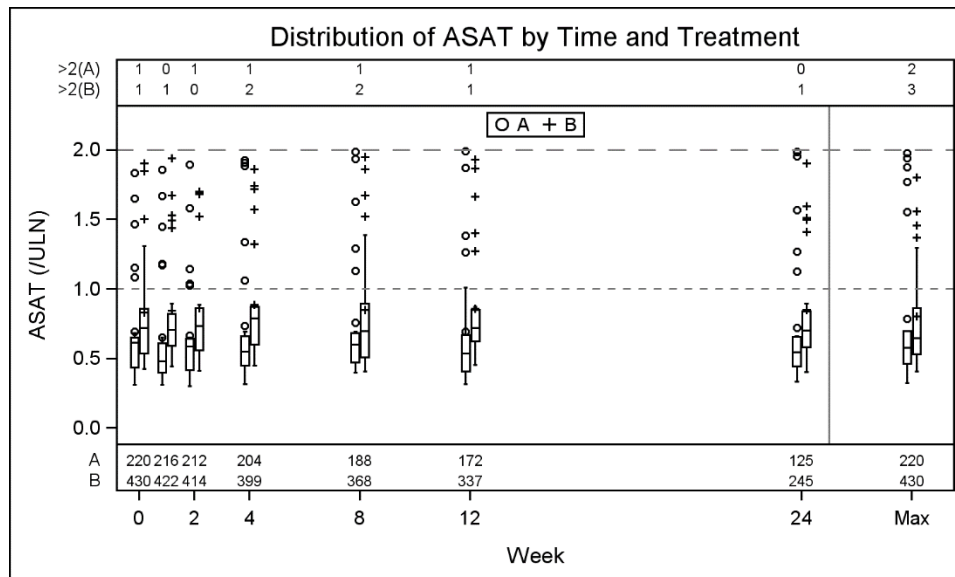
Drawing this graph using the Journal style poses a few challenges, mainly in the drawing of the boxes and their representation in the legend. Using the Journal style, the boxes for Drug "B" will get drawn using dashed lines. Because those look odd, I set the STYLEATTRS option to use only solid lines.

```
ods listing style=journal;
title 'Distribution of ASAT by Time and Treatment';
proc sgplot data=asat2;
  styleattrs datalinepatterns=(solid);
  vbox asat / category=week group=drug nofill;
  scatter x=week y=asat2 / group=drug name='s';
  xaxistable gt2 / class=drugGT colorgroup=drugGT position=top
       location=inside;
  xaxistable count / class=drug colorgroup=drug position=bottom
       location=inside;
  refline 1 / lineattrs=(pattern=shortdash);
  refline 2 / lineattrs=(pattern=dash);
  refline 25 / axis=x;
  xaxis type=linear values=(0 2 4 8 12 24 28) offsetmax=0.05;
  yaxis offsetmax=0.1 valueattrs=(size=8) labelattrs=(size=9);
  keylegend 's' / location=inside position=top linelength=20;
run;
```

Although this improves the rendering of the boxes, it will put two solid lines in the legend for "A" and "B". It would be better to show the mean markers in the legend instead. To do this, I have to add a scatter plot of asat2 by Week and Drug and include that in the legend. Because values in "asat2" are all missing, no markers are displayed in the graph itself, but the group markers are displayed in the legend. Relevant details are shown in the code snippet above. For full details, see Program 4_3.

4.4 Median of Lipid Profile by Visit and Treatment

This graph displays the median of the lipid values by visit and treatment. The visits are at regular intervals and represented as discrete data.

4.4.1 Median of Lipid Profile by Visit and Treatment on Discrete Axis

The values for each treatment are displayed along with the 95% confidence limits as adjacent groups using GROUPDISPLAY option of "Cluster" and the option CLUSTERWIDTH=0.5. The HTMLBlue style is used.

Figure 4.4.1 – Median of Lipid Profile by Visit and Treatment
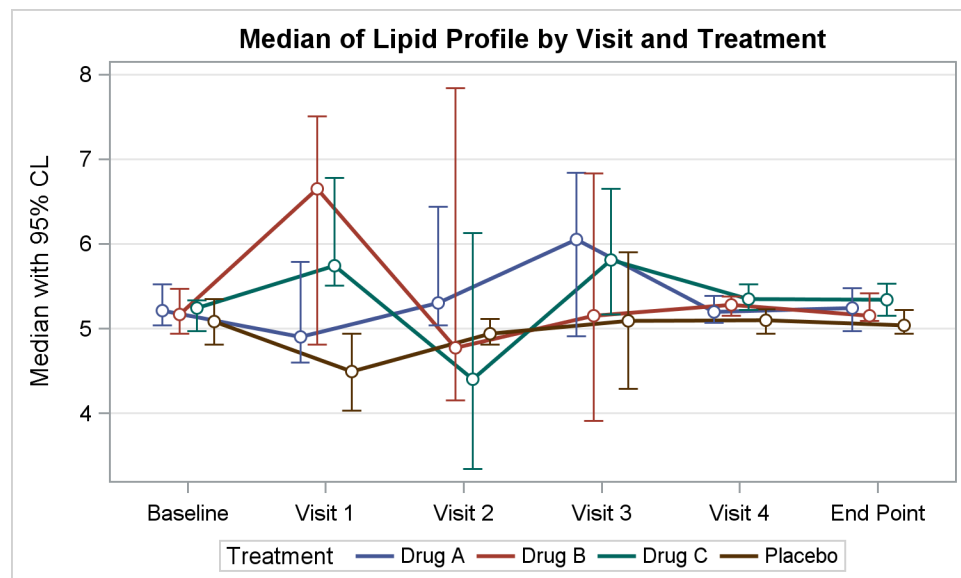

```
title 'Median of Lipid Profile by Visit and Treatment';
proc sgplot data=lipid_grp;
  series  x=day y=median / lineattrs=(pattern=solid) group=trt name='s'
          groupdisplay=cluster clusterwidth=0.5 lineattrs=(thickness=2);
  scatter x=day y=median / yerrorlower=lcl yerrorupper=ucl group=trt
          groupdisplay=cluster clusterwidth=0.5
          errorbarattrs=(thickness=1) filledoutlinedmarkers
          markerattrs=(symbol=circlefilled)
```

```
        markerfillattrs=(color=white);
  keylegend 's' / title='Treatment' linelength=20;
  yaxis label='Median with 95% CL' grid;
  xaxis display=(nolabel);
run;
```

This graph displays the median of the lipid data by visit and treatment. The visits are at regular intervals and represented as discrete data. However, they could also be on a time axis with unequal intervals. The values for each treatment are displayed along with the 95% confidence limits as adjacent groups using GROUPDISPLAY=Cluster and CLUSTERWIDTH=0.5.

The values across visits are joined using a series plot. Note, the series plot also uses cluster groups with the same cluster width. The lengths of the line segments in the legends are reduced using the LINELENGTH option. Markers with fill and outlines are used with specific fill attributes.

Relevant details are shown in the code snippet above. For full details, see Program 4_4.

4.4.2 Median of Lipid Profile by Visit and Treatment on Linear Axis in Grayscale

This graph displays the median of the lipid data by treatment in grayscale on a linear x-axis.

Figure 4.4.2 – Median of Lipid Profile by Visit and Treatment on Linear Axis
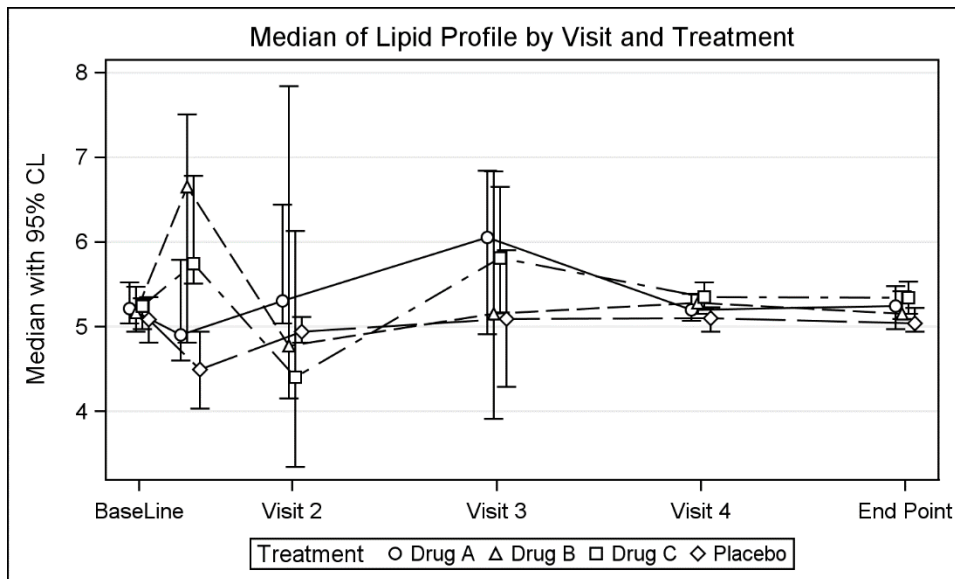

```
title 'Median of Lipid Profile by Visit and Treatment';
proc sgplot data=lipid_Liner_grp;
  styleattrs datasymbols=(circlefilled trianglefilled
            squarefilled diamondfilled);
  series  x=n y=median /  group=trt groupdisplay=cluster
```

```
          clusterwidth=0.5;
  scatter x=n y=median / yerrorlower=lcl yerrorupper=ucl group=trt
          groupdisplay=cluster clusterwidth=0.5
          errorbarattrs=(thickness=1) filledoutlinedmarkers
          markerattrs=(size=7) name='s'
          markerfillattrs=(color=white);
  keylegend 's' / title='Treatment' linelength=20;
  yaxis label='Median with 95% CL' grid;
  xaxis display=(nolabel) values=(1 4 8 12 16);
run;
```

The visits are not at regular intervals and are displayed at the correct scaled location along the x-axis. The visits are at week 1, 2, 4, 8, 12, and 16. These values are formatted to the strings shown on the axis. "Visit 1" collides with "Baseline", causing alternate tick values to be dropped, so I removed "1" from the tick value list.

As you can see, the group values are displayed as clusters, and the "effective midpoint spacing" is the shortest distance between the values. The markers are reduced in size to show the clustering. This can be adjusted by setting marker SIZE=7. Four filled markers are assigned to the list of markers.

Relevant details are shown in the code snippet above. For full details, see Program 4_4.

4.5 Survival Plot

The survival plot is one of the most popular graphs that users want to customize to their own needs. Here I have run the LIFETEST procedure to generate the data for this graph. The output is saved into the "SurvivalPlotData" data set. For more information about the LIFETEST procedure, see the SAS/STAT documentation.

4.5.1 Survival Plot with External "Subjects At-Risk" Table

The survival plot shown below in Figure 4.5.1 has the traditional arrangement where the table of Subjects At-Risk is displayed at the bottom of the graph, below the x-axis.

Figure 4.5.1 – Survival Plot with External "Subjects At-Risk" Table
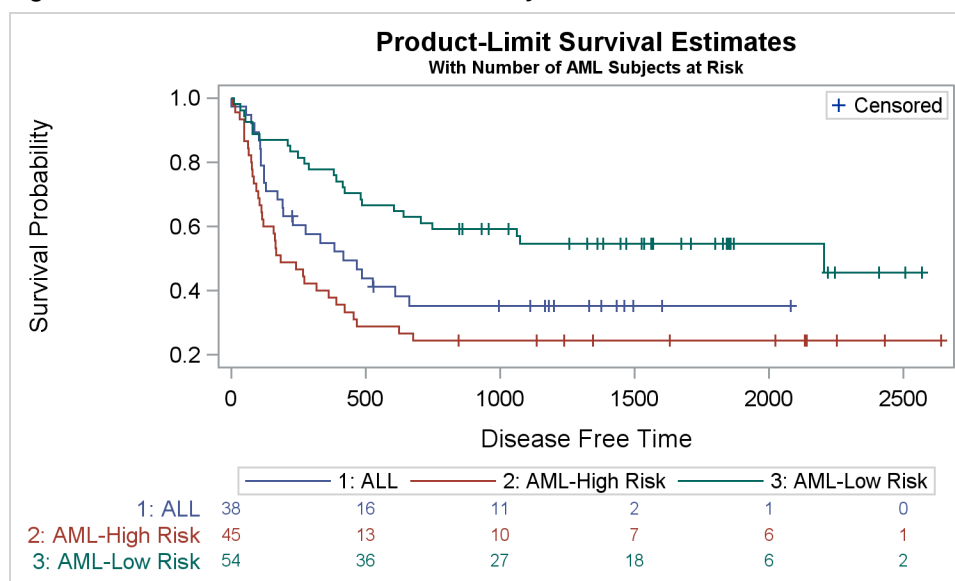

```
ods output Survivalplot=SurvivalPlotData;
proc lifetest data=sashelp.BMT plots=survival(atrisk=0 to 2500 by 500);
   time T * Status(0);
   strata Group / test=logrank adjust=sidak;
   run;
```

A step plot of survival by time by strata displays the curves. A scatter overlay is used to draw the censored values, and an XAXISTABLE statement is used to display the at-risk values at the bottom of the graph. Relevant details are shown in the code snippet above. For full details, see Program 4_5.

```
title 'Product-Limit Survival Estimates';
title2  h=0.8 'With Number of AML Subjects at Risk';
proc sgplot data=SurvivalPlotData;
  step x=time y=survival / group=stratum
```

```
        lineattrs=(pattern=solid) name='s';
  scatter x=time y=censored / markerattrs=(symbol=plus) name='c';
  scatter x=time y=censored / markerattrs=(symbol=plus) GROUP=stratum;
  xaxistable atrisk/x=tatrisk location=outside class=stratum
        colorgroup=stratum;
  keylegend 'c' / location=inside position=topright;
  keylegend 's';
run;
```

4.5.2 Survival Plot with Internal "Subjects At-Risk" Table

The graph shown here is mostly similar to the graph in Section 4.5.1, with the difference that the "Subjects At-Risk" table is moved above the x-axis, close to the rest of the data. Bringing all the data closer makes it easy to align the values with the data, and that improves the effectiveness of the graph.

Figure 4.5.2 – Survival Plot with Internal "Subjects At-Risk" Table
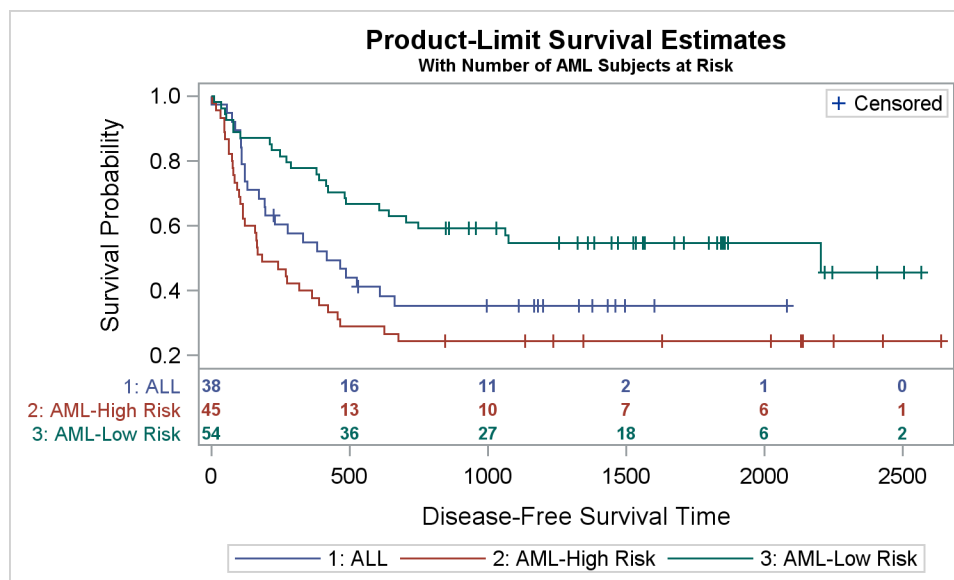

```
title 'Product-Limit Survival Estimates';
title2  h=0.8 'With Number of AML Subjects at Risk';
proc sgplot data=SurvivalPlotData;
  step x=time y=survival / group=stratum lineattrs=(pattern=solid)
        name='s';
  scatter x=time y=censored / markerattrs=(symbol=plus) name='c';
  scatter x=time y=censored / markerattrs=(symbol=plus) GROUP=stratum;
  xaxistable atrisk / x=tatrisk location=inside class=stratum
        colorgroup=stratum separator valueattrs=(size=7 weight=bold)
        labelattrs=(size=8);
```

```
    keylegend 'c' / location=inside position=topright;
    keylegend 's';
run;
```

All this graph needs is to simply specify LOCATION=Inside for the XAXISTABLE statement. In addition to that, we have switched on the separator that draws the horizontal line between the table and the curves.

Relevant details are shown in the code snippet above. For full details, see Program 4_5.

4.5.3 Survival Plot with Internal "Subjects At-Risk" Table in Grayscale

Displaying the survival plot in a grayscale medium presents some challenges.

Here we cannot use colors to identify the strata. Normally, the Journal style uses line patterns to identify the groups. Although line patterns work well for curves, they are not so effective with step plots because of the frequent breaks. So, it is preferable to use solid lines for all the levels of the step plot and to use markers to identify the strata.

Figure 4.5.3 – Survival Plot with Internal "Subjects At-Risk" Table in Grayscale
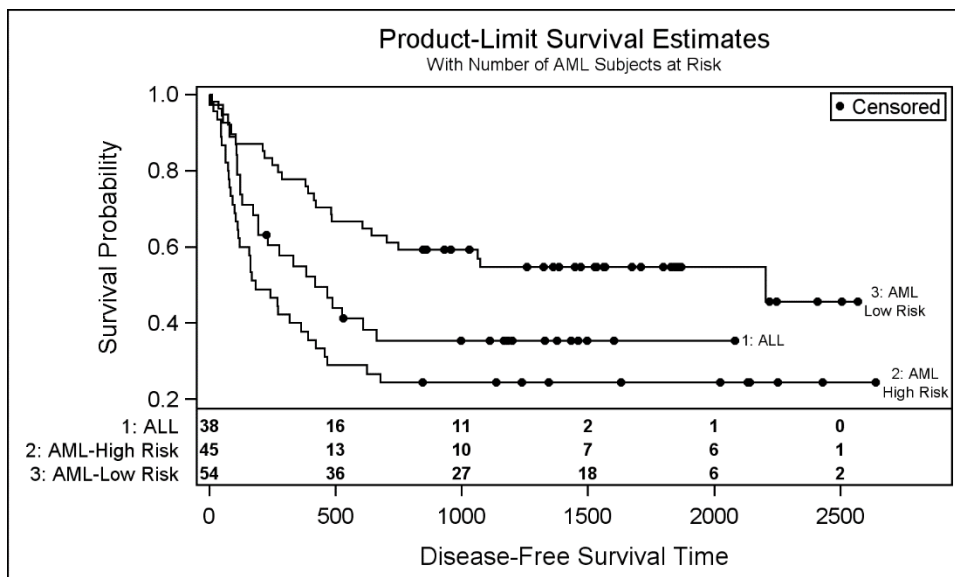

```
    title 'Product-Limit Survival Estimates';
    title2  h=0.8 'With Number of AML Subjects at Risk';
    proc sgplot data=SurvivalPlotData;
      step x=time y=survival / group=stratum lineattrs=(pattern=solid)
            name='s' curvelabel curvelabelattrs=(size=6) splitchar='-';
      scatter x=time y=censored / name='c'
            markerattrs=(symbol=circlefilled size=4);
      xaxistable atrisk / x=tatrisk location=inside class=stratum
```

```
        colorgroup=stratum separator valueattrs=(size=7 weight=bold)
        labelattrs=(size=8);
    keylegend 'c' / location=inside position=topright;
run;
```

In this case, markers are also used to identify the censored observations. So, I have chosen to use the CURVELABEL option with the SPLITCHAR option to identify the curves. This results in a clean and effective graph, without the need for a legend for the strata.

Relevant details are shown in the code snippet above. For full details, see Program 4_5.

4.6 Simple Forest Plot

This forest plot is a graphical representation of a meta-analysis of the results of randomized controlled trials.

4.6.1 Simple Forest Plot

The graph in Figure 4.6.1 shows a simple forest plot, with the odds ratio plot in the middle by study, along with the tabular display of the statistics on the right.

Figure 4.6.1 – Simple Forest Plot
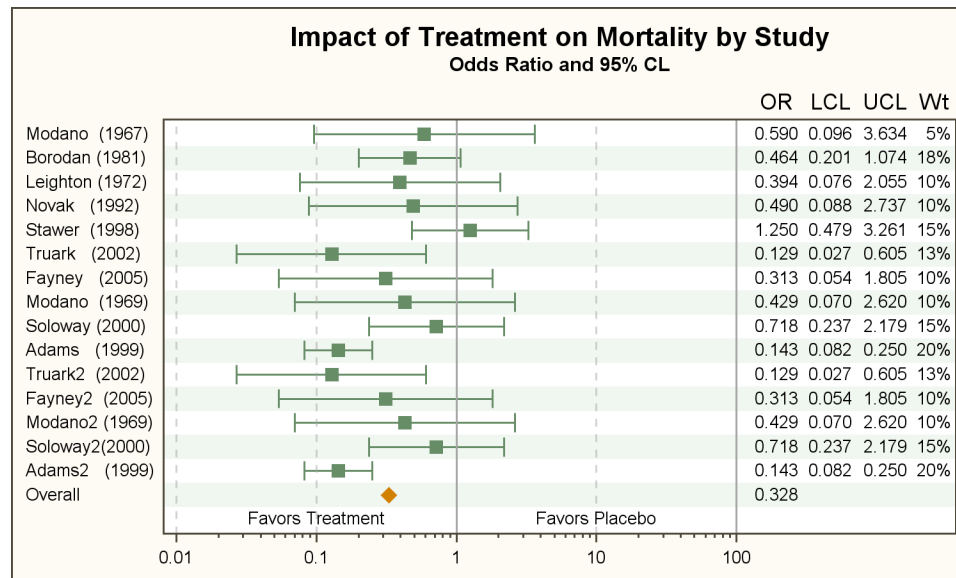

```
title "Impact of Treatment on Mortality by Study";
title2 h=8pt 'Odds Ratio and 95% CL';
proc sgplot data=forest noautolegend nocycleattrs;
  styleattrs datasymbols=(squarefilled diamondfilled);
  scatter y=study x=or / xerrorupper=ucl xerrorlower=lcl group=grp;
```

```
    yaxistable or lcl ucl wt / y=study location=inside position=right;
    refline 1 100  / axis=x noclip;
    refline 0.01 0.1 10 / axis=x lineattrs=(pattern=shortdash) noclip;
    text y=study x=xlbl text=lbl  / position=center contributeoffsets=none;
    xaxis type=log  max=100 minor display=(nolabel)  valueattrs=(size=7);
    yaxis display=(noticks nolabel)  fitpolicy=none reverse
          valueshalign=left colorbands=even )  valueattrs=(size=7);
run;
```

The data for this graph contains the odds ratio, the confidence limits, and the weight for each study. The studies are reclassified with "1" for individual study names and "2" for "Overall". We use this information to plot the graph by study using SCATTERPLOT and YAXISTABLE statements. Note that this graph uses the Analysis style that has an attribute priority of "None", and producing the display of varying markers by group.

We have used the TEXT statement to place the "Favors" strings at the bottom, using a study value of NBSP. Y-axis tick values are left-aligned, and the fit policy is set to "none" so that all tick values are displayed regardless of congestion. For full details, see Program 4_6.

4.6.2 Simple Forest Plot with Study Weights

The graph below shows a simple forest plot with Study Weights, where the markers in the odds ratio plot are sized by the weight of the study.

Figure 4.6.2 – Simple Forest Plot with Study Weights
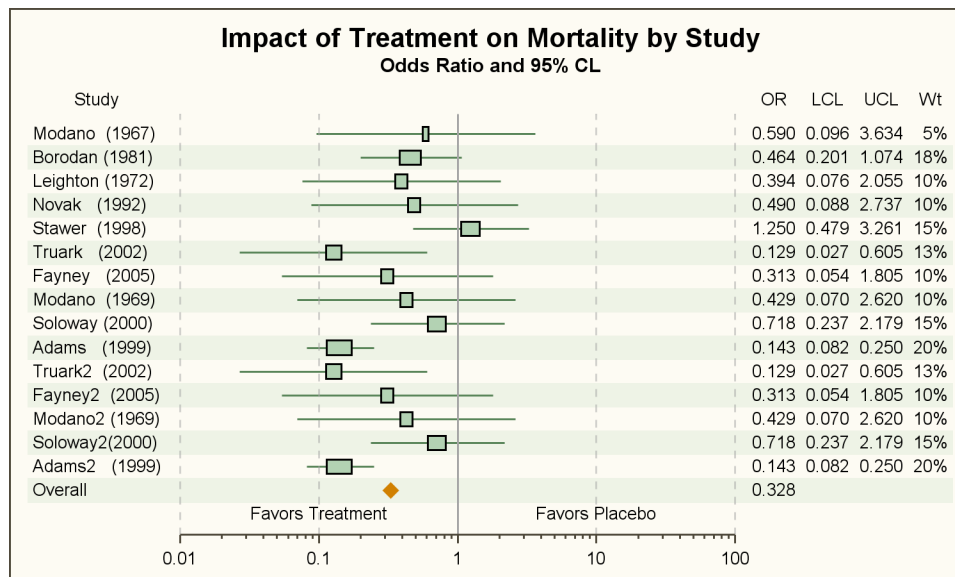

```
title "Impact of Treatment on Mortality by Study";
title2 h=8pt 'Odds Ratio and 95% CL';
proc sgplot data=forest noautolegend nocycleattrs nowall noborder;
```

```
  styleattrs axisextent=data;
  scatter y=study x=or2 / markerattrs=graphdata2(symbol=diamondfilled);
  highlow y=study low=lcl high=ucl / type=line;
  highlow y=study low=q1 high=q3 / type=bar barwidth=0.6;
  yaxistable study / y=study location=inside position=left
        labelattrs=(size=7);
  yaxistable or lcl ucl wt / y=study location=inside position=right;
  refline 1  / axis=x noclip;
  refline 0.01 0.1 10 100 / axis=x lineattrs=(pattern=shortdash noclip;
  text y=study x=xlbl text=lbl  / position=center contributeoffsets=none;
  xaxis type=log  max=100 minor display=(nolabel)  valueattrs=(size=7);
  yaxis display=none fitpolicy=none reverse valueshalign=left
    colorbands=even colorbandsattrs=Graphdatadefault(transparency=0.75);
run;
```

This graph uses a highlow plot to display the relative weights for each study and the confidence interval. The scatter plot uses the "OR2" variable, which is non-missing only for the "Overall" study. So, only the diamond marker is drawn by the scatter plot.

The width of each marker is proportional to the weight in linear scale. However, because we have used a log x-axis, the widths might not be represented accurately in log scale. So, this can provide a qualitative representation of the weight.

The graph has no wall or wall borders and the x-axis line is displayed only to the extent of the data by using the AXISEXTENT=DATA option on the STYLEATTRS statement. The y-axis is replaced by a YAXISTABLE on the left side so that the bands extend across the full graph.

Relevant details are shown in the code snippet above. For full details, see Program 4_6.

4.6.3 Simple Forest Plot with Study Weights in Grayscale

Figure 4.6.3 shows the same forest plot in grayscale medium.

Figure 4.6.3 – Simple Forest Plot with Study Weights in Grayscale
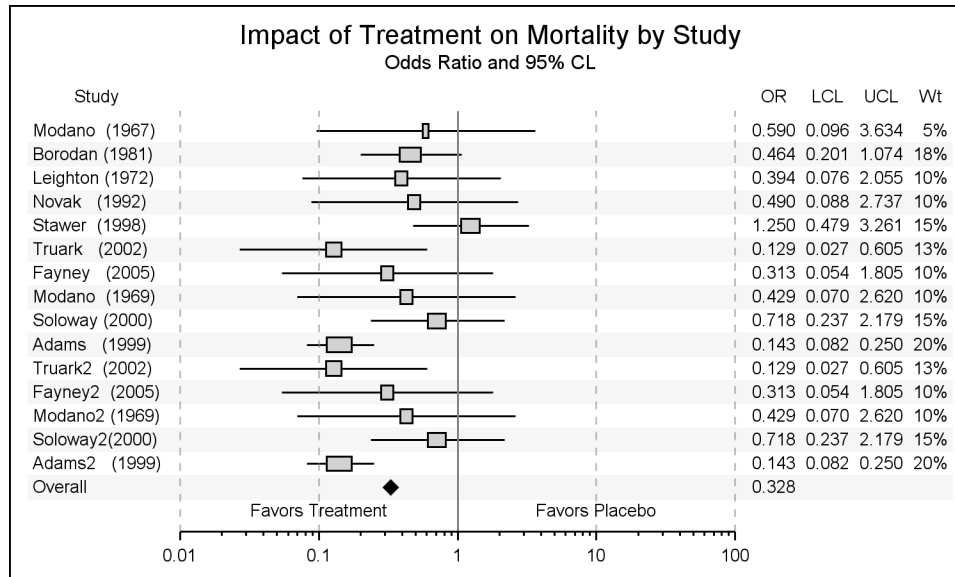

```
ods listing style=journal;
title "Impact of Treatment on Mortality by Study";
title2 h=8pt 'Odds Ratio and 95% CL';
proc sgplot data=forest noautolegend nocycleattrs nowall noborder;
  styleattrs axisextent=data;
  scatter y=study x=or2 / markerattrs=graphdata2(symbol=diamondfilled);
  highlow y=study low=lcl high=ucl / type=line;
  highlow y=study low=q1 high=q3 / type=bar barwidth=0.6;
  yaxistable study / y=study location=inside position=left
          labelattrs=(size=7);
  yaxistable or lcl ucl wt / y=study location=inside position=right
          labelattrs=(size=7);
  refline 1  / axis=x noclip;
  refline 0.01 0.1 10 100 / axis=x lineattrs=(pattern=shortdash)noclip;
  text y=study x=xlbl text=lbl  / position=center contributeoffsets=none;
  xaxis type=log  max=100 minor display=(nolabel)  valueattrs=(size=7);
  yaxis display=none fitpolicy=none reverse valueshalign=left
        colorbands=even valueattrs=(size=7)
        colorbandsattrs=Graphdatadefault(transparency=0.8);
run;
```

Rendering this graph in a grayscale medium does not pose a lot of challenges. Basically, we have set the ODS style to JOURNAL to produce the graph above. This is structurally similar to the graph shown in Section 4.6.2.

Relevant details are shown in the code snippet above. For full details, see Program 4_6.

4.7 Subgrouped Forest Plot

More recently, many of you have been asking for building a forest plot, where each study value has multiple subgroups. The example in Figure 4.7.1 shows two subgroups per observation.

For the graph shown below, only the hazard ratio plot in the middle is displayed using a plot statement. The tabular data is displayed in axis tables.

Figure 4.7.1 – Subgrouped Forest Plot
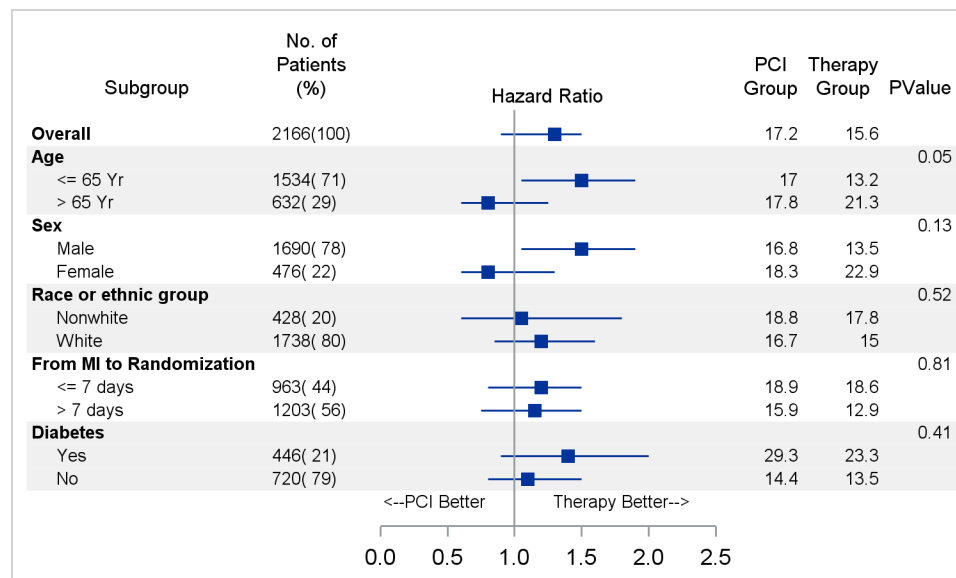

```
proc sgplot data=forest_subgroup_2 nowall noborder nocycleattrs
dattrmap=attrmap;
  styleattrs axisextent=data;
  refline ref / lineattrs=(thickness=13 color=cxf0f0f0);
  highlow y=obsid low=low high=high;
  scatter y=obsid x=mean / markerattrs=(symbol=squarefilled);
  scatter y=obsid x=mean / markerattrs=(size=0) x2axis;
  refline 1 / axis=x;
  text x=xl y=obsid text=text / position=center contributeoffsets=none;
  yaxistable subgroup  / location=inside position=left textgroup=id
        labelattrs=(size=8) textgroupid=text indentweight=indentWt;
  yaxistable countpct / location=inside position=left;
  yaxistable PCIGroup group pvalue / location=inside position=right;
  yaxis reverse display=none colorbands=odd
        colorbandsattrs=(transparency=1);
```

```
    xaxis display=(nolabel) values=(0.0 0.5 1.0 1.5 2.0 2.5);
    x2axis label='Hazard Ratio' display=(noline noticks novalues);
run;
```

The graph above displays the hazard ratio and confidence limits by subgroup, along with the number of patients in the study and other statistics. The key difference here is the display of the subgroups and values in the first column. The subgroup titles are displayed in a bold font, and the values are displayed in a normal font and indented to the right.

Table 4.7.2 - Data for Subgrouped Forest Plot

Obs	ObsId	Id	Subgroup	indentWt	Count	Percent	Mean	Low	High	PCIGroup	Group	PValue	ref
1	1	1	Overall	0	2166	100	1.3	0.90	1.50	17.2	15.6	.	.
2	2	1	Age	0	.	.	.	.	.	.	.	0.05	2
3	3	2	<= 65 Yr	1	1534	71	1.5	1.05	1.90	17.0	13.2	.	3
4	4	2	> 65 Yr	1	632	29	0.8	0.60	1.25	17.8	21.3	.	4
5	5	1	Sex	0	.	.	.	.	.	.	.	0.13	.
6	6	2	Male	1	1690	78	1.5	1.05	1.90	16.8	13.5	.	.
7	7	2	Female	1	476	22	0.8	0.60	1.30	18.3	22.9	.	.

The data for the graph is shown above. The study values come from the "Subgroup" column and are displayed by "ObsId" order. For subgroup labels like "Overall", the Id value is "1", and for the values in the subgroup, the ID is "2". This ID is used to control the attributes of the values that are displayed in the first column of the graph using the attribute map as defined below. Text attributes are defined by the value. Id=1 values are displayed with a bigger, bold font.

In addition, the indention of some of the values in column 1 are controlled by "IndentWt" column. The default indention value is 1/8 inch, and can be changed using the INDENT option. Actual indention amount is based on the INDENTWEIGHT * INDENT.

Figure 4.7.3 – Attribute Map for Font Attributes

Obs	id	value	textcolor	textsize	textweight
1	text	1	Black	7	bold
2	text	2	Black	5	normal

The second column contains a combination of patient count and percentage and is displayed by another YAXISTABLE statement. The hazard ratio graph in the middle is displayed using a highlow plot and a scatter plot. Then, the three columns on the right are displayed using another YAXISTABLE, with three columns.

The insets at the bottom, "<- PCI Better" and "Therapy Better ->", are displayed in a text plot using the "text", "xl", and "ObsId" columns. The text is center justified. See the full code for this information in the data set, available from the author's page at http://support.sas.com/matange.

Finally, the wide horizontal bands across the graph are drawn using the REFLINE statement with the "Ref" column. This column is a copy of the ObsId column where alternate 3 observations are set to missing. Reference lines are not drawn when the value is missing. Also note, the x-axis line is drawn only to the extent of the actual data, and not all the way using the AXISEXTENT option on the STYLEATTRS statement.

Relevant details are shown in the code snippet above. For full details, see Program 4_7.

4.8 Adverse Event Timeline by Severity

An Adverse Event Timeline graph by AEDECOD and Severity is useful to view the history of a specific subject in a study.

Figure 4.8.1 – Adverse Event Timeline by Severity
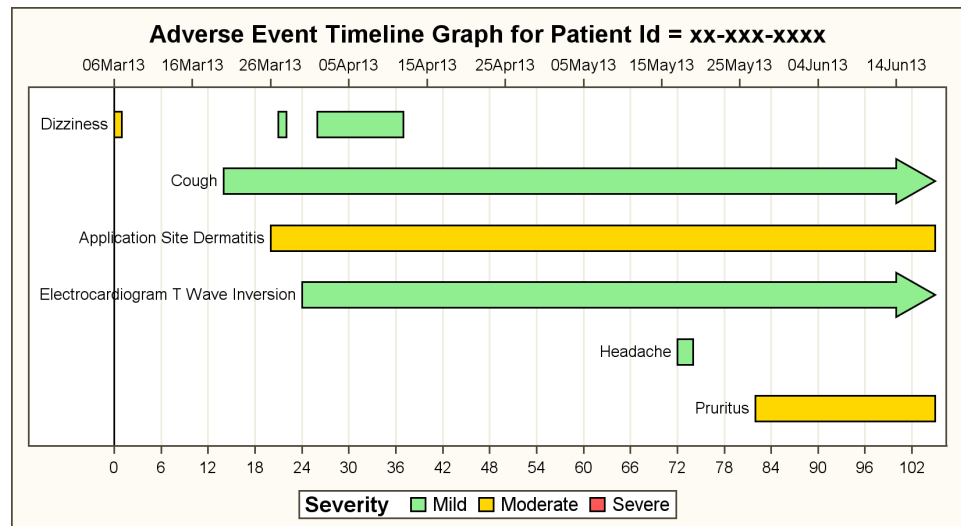

```
title "Adverse Event Timeline Graph for Patient Id = &pid";
proc sgplot data=ae3s dattrmap=attrmap;
  format stdate date7.;
  refline 0 / axis=x lineattrs=(color=black);
  highlow y=aedecod low=stday high=enday / type=bar group=aesev
          lineattrs=(color=black pattern=solid) barwidth=0.8
          lowlabel=aedecod highcap=highcap attrid=Severity
          nomissinggroup labelattrs=(color=black size=7);
  scatter y=aedecod x=stdate / x2axis markerattrs=(size=0);
  xaxis grid display=(nolabel) valueattrs=(size=7)
```

```
      values=(&minday to &maxday by 2) offsetmax=0.02 ;
 x2axis display=(nolabel) type=time valueattrs=(size=7) v
      alues=(&mindate to &maxdate) offsetmax=0.02;
yaxis  reverse  display=(noticks novalues nolabel);
run;
```

The graph above displays each adverse event as a bar segment over its duration. The color of the event is set by the severity. The source data is in CDISC, using the SDTM tabulation model format, as shown below.

Figure 4.8.2 – Data Set for Adverse Event Timeline Graph

Obs	aeseq	aedecod	aesev	aestdtc	aeendtc
1	1	Dizziness	Moderate	2013-03-06	2013-03-06
2	2	Cough	Mild	2013-03-20	
3	4	Dizziness	Mild	2013-03-27	2013-03-27
4	5	Electrocardiogram T Wave Inversion	Mild	2013-03-30	

The data has many columns, but the ones that we are using are aeseq, aedecod, aesev, aestdtc, and aeendtc. In the example above, all aestdtc values are present and assumed to be valid. If not, some data cleaning might be needed. In the DATA step, stdate is extracted from aestdtc and endate from aeendtc. If aeendtc is missing, the largest value of endate is used, and highcap is set to "FilledArrow" to indicate that the event does not have an end date. A valid end date is required to draw the event in the graph. The data set that is required for plotting the graph is shown below.

Figure 4.8.3 – Data Set for Adverse Event Timeline Graph with Caps

Obs	aeseq	aedecod	aesev	sev	stdate	endate	stday	enday	highcap
1	1	Dizziness	Moderate	2	06MAR2013	06MAR2013	0	1	
2	2	Cough	Mild	1	20MAR2013	18JUN2013	14	105	FilledArrow
3	6	Application Site Dermatitis	Moderate	2	26MAR2013	18JUN2013	20	105	
4	3	Dizziness	Mild	1	27MAR2013	27MAR2013	21	22	

The data set below is computed for creating the graph. Note that in this data set, we do not have any observations with Severity="Severe". However, the legend in the graph does have an entry for "Severe". These dummy observations do not have valid start and end values, so they are not actually drawn in the graph. The top x-axis is enabled by using a scatter plot assigned to the x2-axis. Macro variables are used to align the x- and x2-axes.

Observations with specific group values are assigned the color and other attributes from the GraphData1-12 style elements. These are assigned in the order in which they are encountered in the data. In this case, we are using specific colors for "Mild", "Moderate", and "Severe". If we just

change the colors of the style elements, we will get one of the three colors, but the color assignments can shift based on the order of the data.

To ensure consistent and reliable color assignment, we will use the Discrete Attribute Map data set. Colors and the specific values of the group values are explicitly assigned. Now, the group values will get the colors by value, and not those based on the order of the values in the data. In this case, LineColor is used both for lines and text.

Figure 4.8.4 – Discrete Attribute Map Data Set

Obs	Id	Show	Value	Fillcolor	Linecolor
1	Severity	Attrmap	Mild	lightgreen	darkgreen
2	Severity	Attrmap	Moderate	gold	cx9f7f00
3	Severity	Attrmap	Severe	lightred	darkred

Another benefit of using the attribute map is based on the SAS 9.4 option "Show" in the map data set. This applies to every map "ID" that is defined in the data set. In this case, there is only one "Severity". By default, only the values that occur in the data are included in the legend. But, if Show is set to "AttrMap", then all the values from the attribute map ID are shown in the legend. In this case, even though the aesev value of "Severe" never occurs in the data, it is still shown in the legend. Another benefit of this feature is that the values that are shown in the legend are sorted in the same order in which they appear in the attribute map. So, we can get a custom sorting of the legend by using this feature.

The highlow plot is ideally suited for such a use case, and provides support for drawing labels and arrowhead caps at each end. In this case, the LOWLABEL option is used to draw the event names. We have displayed the aedecod label only the first time. The HIGHCAP option is used to draw the arrowhead as shown for "Cough" at the right end. This indicates an event that does not have an end date in the data.

For the grayscale use case, we can change the highlow bar type to the default "Line". This will allow use of the line pattern as the visual element for the different severity values. Here is the graph, along with the appropriate attribute map.

Figure 4.8.5 – Adverse Event Timeline for Grayscale
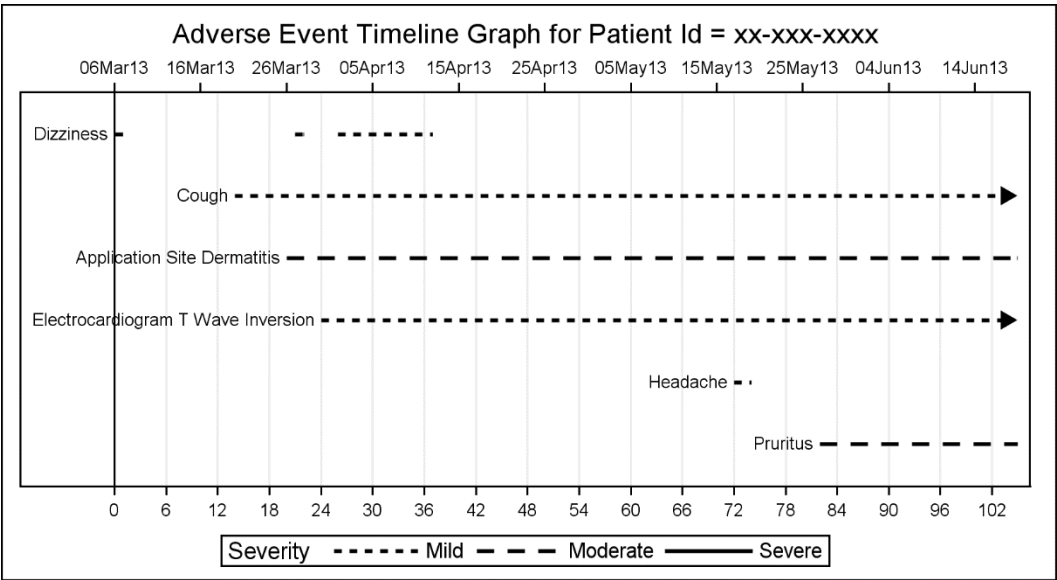
Figure 4.8.6 – Attribute Map for Grayscale Graph

Obs	Id	Show	Value	Fillcolor	LinePattern	LineThickness
1	Severity	Attrmap	Mild	lightgreen	shortdash	2
2	Severity	Attrmap	Moderate	gold	dash	2
3	Severity	Attrmap	Severe	lightred	solid	2

Relevant details are shown in the code snippet above. For full details, see Program 4_8.

4.9 Change in Tumor Size

The graph shown below is commonly known at a "waterfall chart" in the oncology domain. The graph displays the change in tumor size by treatment.

Figure 4.9.1 – Graph of Change in Tumor Size
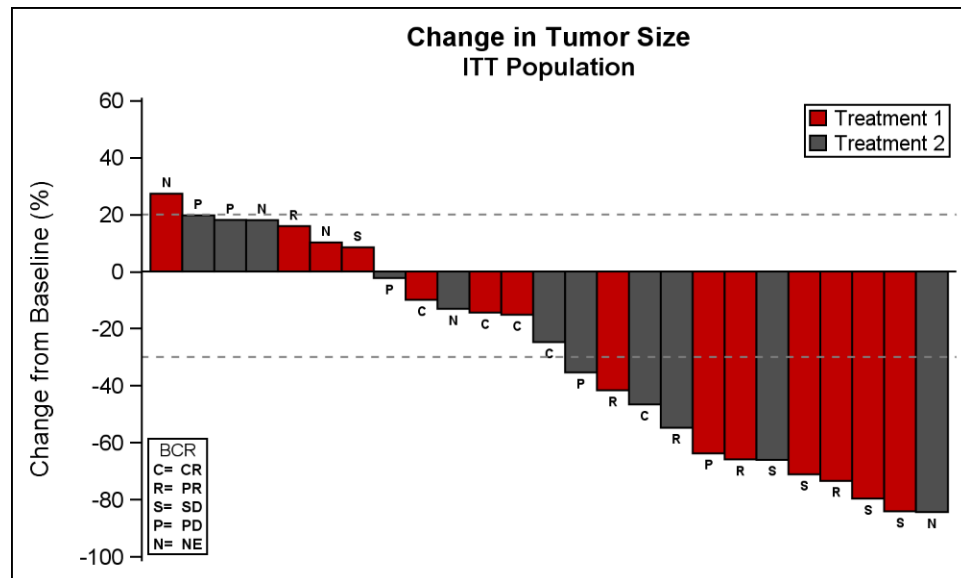

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=TumorSize nowall noborder;
  styleattrs datacolors=(cxbf0000 cx4f4f4f) datacontrastcolors=(black);
  vbar cid / response=change group=group categoryorder=respdesc
      datalabel=label datalabelattrs=(size=5) groupdisplay=cluster
      clusterwidth=1;
  refline 20 -30 / lineattrs=(pattern=shortdash);
  xaxis display=none;
  yaxis values=(60 to -100 by -20);
  inset ("C="="CR" "R="="PR" "S="="SD" "P="="PD" "N="="NE") / title='BCR'
      position=bottomleft border textattrs=(size=6)
      titleattrs=(size=7);
  keylegend / title='' location=inside position=topright across=1 border;
run;
```

The graph displays the change in tumor size in descending order of size increase for the population by treatment. The data is shown on the right.   The response type is shown at the end of the bar.

Figure 4.9.2 – Data for Waterfall Chart

Obs	Cid	Change	Id	Group	Label
1	1	-14.3910	1	Treatment 1	C
2	2	-65.9578	2	Treatment 1	R
3	3	-71.1846	3	Treatment 1	S
4	4	19.6850	4	Treatment 2	P
5	5	18.0494	5	Treatment 2	N

Confidence limits are shown at +20% and -30%. A partial response is generally indicated for tumor shrinkage of 30% or more; however, the author does not claim domain-specific expertise. See domain-centric papers for more information about such details.

The STYLEATTRS statement is used to control the colors for treatments 1 and 2. For specific assignment of colors by treatment, a discrete attribute map would be preferred.

A serious clinical graph does not necessarily have to have boring aesthetics. The graph below displays the same information using a different set of colors and presentation aspects, including bars with a textured look. The confidence region is displayed using a band plot with 50% transparency.

Figure 4.9.3 – Waterfall Chart with Alternative Appearance
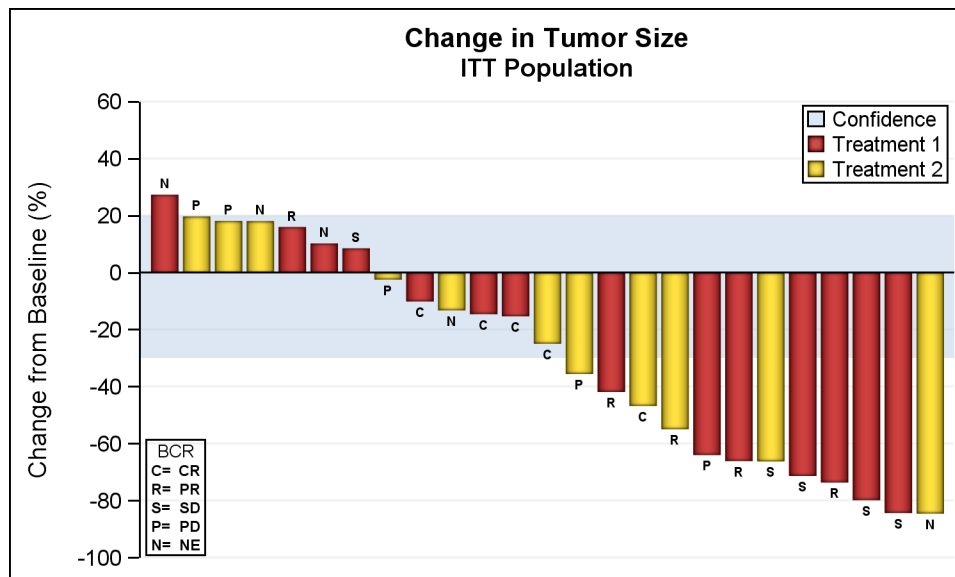

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=TumorSizeDesc nowall noborder;
```

```
    styleattrs datacolors=(cxbf0000 gold) datacontrastcolors=(black);
    band x=cid upper=20 lower=-30 /transparency=0.5
         filllegendlabel='Confidence';
    vbarparm category=cid  response=change / group=group datalabel=label
             datalabelattrs=(size=5 weight=bold) groupdisplay=cluster
             dataskin=pressed;
    xaxis display=none;
    yaxis values=(60 to -100 by -20) grid gridattrs=(color=cxf0f0f0);
    inset ("C="="CR" "R="="PR" "S="="SD" "P="="PD" "N="="NE") / title='BCR'
         position=bottomleft border textattrs=(size=6)
         titleattrs=(size=7);
    keylegend / title='' location=inside position=topright
         across=1 border opaque;
run;
```

A VBARPARM statement is used instead of a VBAR statement because we want to layer a band plot in the graph. Grid lines are enabled, and the legend has an opaque background. Group display of "Cluster" is used so that we can display the bar data labels.

Relevant details are shown in the code snippet above. For full details, see Program 4_9.

4.10 Injection Site Reaction

The graph in Figure 4.10.1 shows the incidence of injection site reaction by Time and Cohort.

4.10.1 Injection Site Reaction

The (simulated) data is shown on the right, with incidence by group over time.

Figure 4.10.1.1 – Graph of Injection Site Reaction
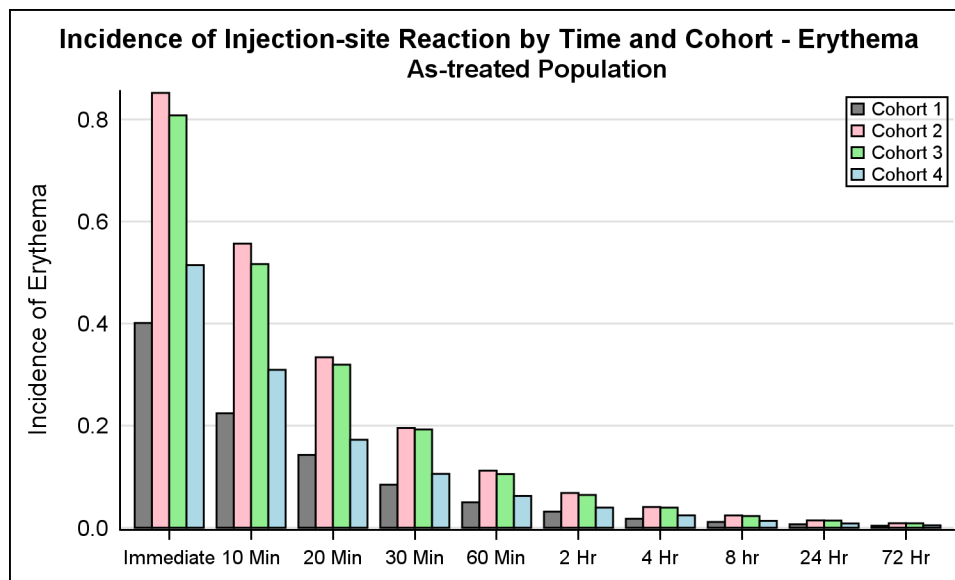

```
title 'Incidence of Injection-site Reaction by Time and Cohort -
Erythema';
title2 'As-treated Population';
ods listing style=listing;
proc sgplot data=Incidence nowall noborder;
  styleattrs datacolors=(gray pink lightgreen lightblue)
             datacontrastcolors=(black);
  vbar time / response=incidence group=group groupdisplay=cluster;
  xaxis discreteorder=data valueattrs=(size=8) fitpolicy=none
        display=(nolabel);
  yaxis grid display=(noticks);
  keylegend / title='' location=inside position=topright across=1 border
        autoitemsize valueattrs=(size=8);
run;
```

We have used a VBAR statement with Time as the category and Cohort (Group) as the group. The time values are treated as discrete, and each cluster of incidence bars is positioned at equidistant midpoints along the axis.

Figure 4.10.1.1 – Data Set for Injection Site Reaction Graph

Obs	Time	Group	Incidence
1	Immediate	Cohort 1	0.40112
2	Immediate	Cohort 2	0.85194
3	Immediate	Cohort 3	0.80790
4	Immediate	Cohort 4	0.51452

The STYLEATTRS statement is used to set the four colors for the group values. Y-axis grid lines are enabled, and the tick marks are removed.

4.10.2 Injection Site Reaction in Grayscale

The graph above uses the JOURNAL2 style that is suitable for submissions to journals that are published in grayscale medium. The group classifications are displayed using fill patterns.

Figure 4.10.2 – Graph of Injection Site Reaction in Grayscale
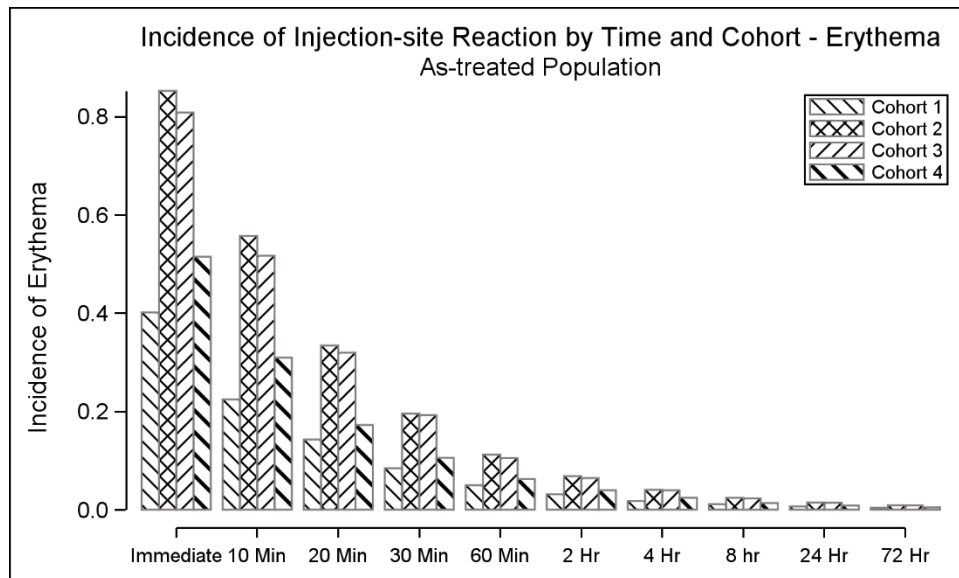

```
title 'Incidence of Injection-site Reaction by Time and Cohort - ';
title2 'As-treated Population';
ods listing style=Journal2;
proc sgplot data=Incidence nowall noborder;
  styleattrs datacolors=(gray pink lightgreen lightblue )
             datacontrastcolors=(black) axisextent=data;
  vbar time / response=incidence group=group groupdisplay=cluster
```

```
                baselineattrs=(thickness=0) outlineattrs=(color=gray);
  xaxis discreteorder=data valueattrs=(size=8) fitpolicy=none
        display=(nolabel);
  yaxis offsetmin=0.04 grid display=(noticks);
  keylegend / title='' location=inside position=topright across=1 border
              autoitemsize valueattrs=(size=8);
run;
```

We have used a new SAS 9.4 feature to display the axis only for the extent of the data using the AXISEXTENT=DATA on the STYLEATTRS statement. This produces results that are preferred by many users.

Relevant details are shown in the code snippet above. For full details, see Program 4_10.

4.11 Distribution of Maximum LFT by Treatment

The graph below shows the distribution of LFT values by Test and Treatment.

4.11.1 Distribution of Maximum LFT by Treatment with Multi-Column Data

The graph shows the distribution of LFT values by Test and Treatment using multi-column data.

Figure 4.11.1.1 – Distribution of Maximum LFT by Treatment
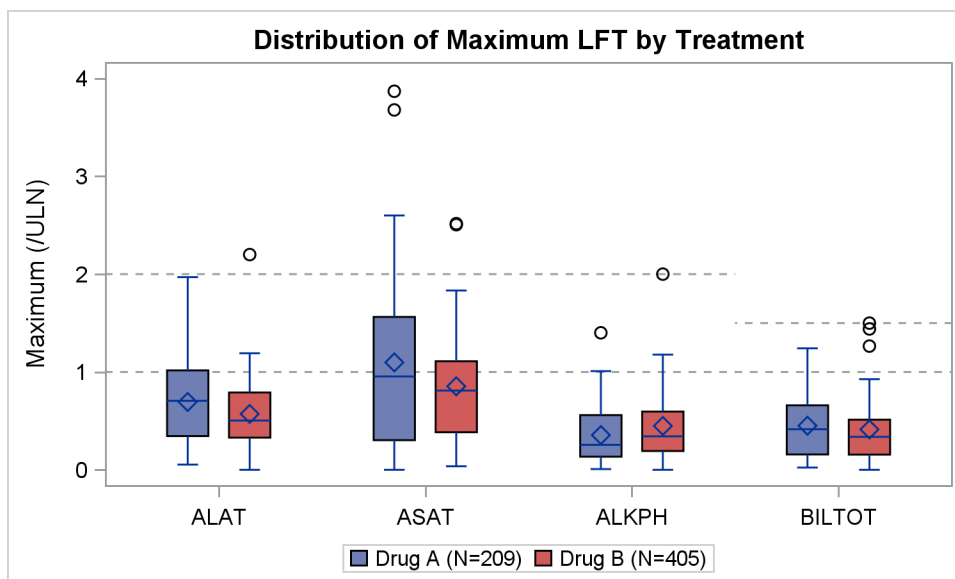

```
    title 'Distribution of Maximum LFT by Treatment';
    footnote j=l 'Level of concern is 2.0 for ALAT, ASAT, ALKPH and 1.5
    for BILTOT';
```

```
proc sgplot data=LFT;
  refline 1 / lineattrs=(pattern=shortdash);
  dropline x='BILTOT' y=2.0 / dropto=y discreteoffset=-0.5;
  dropline x='BILTOT' y=1.5 / y2axis dropto=y discreteoffset=-0.5;
  vbox a / category=test discreteoffset=-0.15 boxwidth=0.2 name='a'
          legendlabel='Drug A (N=209)';
  vbox b / category=test discreteoffset= 0.15 boxwidth=0.2 name='b'
          legendlabel='Drug B (N=405)';
  vbox a / category=test y2axis transparency=1;
  vbox b / category=test y2axis transparency=1;
  keylegend 'a' 'b';
  xaxis display=(nolabel);
  y2axis display=none;
run;
```

The graph above uses two VBOX statements, one each for the values for drugs A and B.

Figure 4.11.1.2 – Data for Graph Using Multiple Columns

Obs	Test	A	B
1	ALAT	1.05198	0.97755
2	ASAT	0.78177	0.59554
3	ALKPH	0.20475	0.20589
4	BILTOT	0.12868	0.10760
5	ALAT	1.00211	1.19132

The levels of concern for the lab tests are different, so we have used the DROPLINE statement to draw the levels differently for the upper and lower values. Discrete offset is used to start the drop line halfway between the lab values.

4.11.2 Distribution of Maximum LFT by Treatment Grayscale with Group Data

This graph displays the Distribution of Maximum LFT graph by Treatment group in grayscale.

Figure 4.11.2.1 – Distribution of Maximum LFT by Treatment
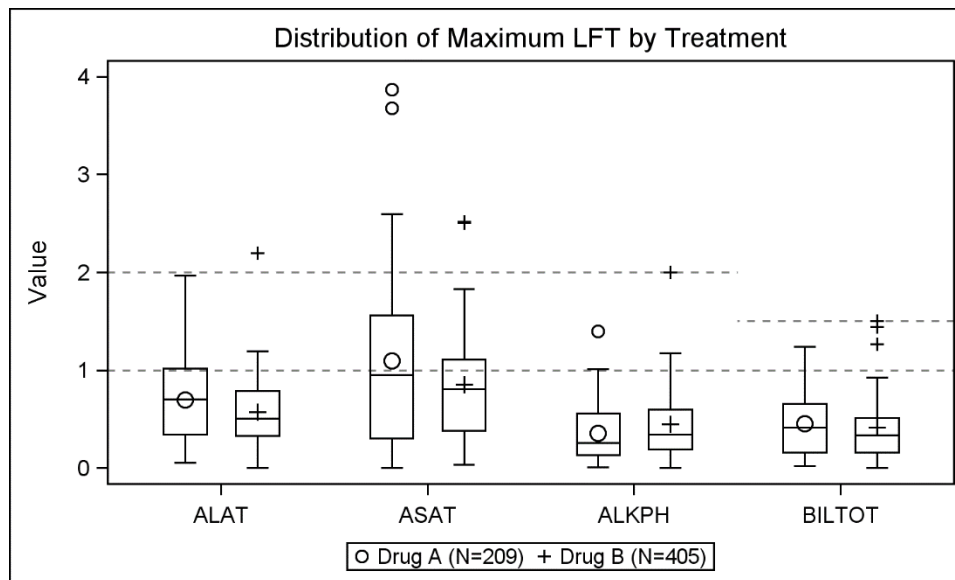

```
title 'Distribution of Maximum LFT by Treatment';
proc sgplot data=lft_Grp;
  styleattrs datalinepatterns=(solid);
  refline 1 / lineattrs=(pattern=shortdash);
  dropline x='BILTOT' y=2.0 / dropto=y discreteoffset=-0.5;
  dropline x='BILTOT' y=1.5 / y2axis dropto=y discreteoffset=-0.5;
  vbox value / category=test group=drug groupdisplay=cluster nofill;
  scatter x=test y=out / y2axis group=drug name='a';
  keylegend 'a';
  xaxis display=(nolabel);
  y2axis display=none min=0 max=4;
run;
```

In this example, the data is arranged by group, instead of by multi-column as in 4.11.1. We are using empty boxes in a black and white medium using the Journal style. We have set all lines to solid, so we need another way to indicate the treatment name.

Figure 4.11.2.2 – Data for Graph Using Group Data

Obs	Test	Drug	Value	Out
1	ALAT	Drug A (N=209)	1.05198	5
2	ALAT	Drug B (N=405)	0.97755	5
3	ASAT	Drug A (N=209)	0.78177	5
4	ASAT	Drug B (N=405)	0.59554	5
5	ALKPH	Drug A (N=209)	0.20475	5

Here we used a scatter overlay, with Y=OUT, on a column that has all values > 4. We also set the y-axis MAX=4 in order to remove these fake markers while still retaining them in the legend.

Relevant details are shown in the code snippet above. For full details, see Program 4_11.

4.12 Clark Error Grid

The Clark Error Grid graph is used to quantify the clinical accuracy of blood glucose levels that are generated by the meters. The sensor response and the reference value are plotted on the grid.

4.12.1 Clark Error Grid

The graph includes demarcated zones that indicate the divergence of the meter values from reference values. Zone "A" demarcates the zone where the divergence is < 20%. Zone "B" has divergence > 20%, but not leading to improper treatment. Other zones indicate dangerous or confusing results.

Figure 4.12.1 – Clark Error Grid for Blood Glucose Measurement Accuracy
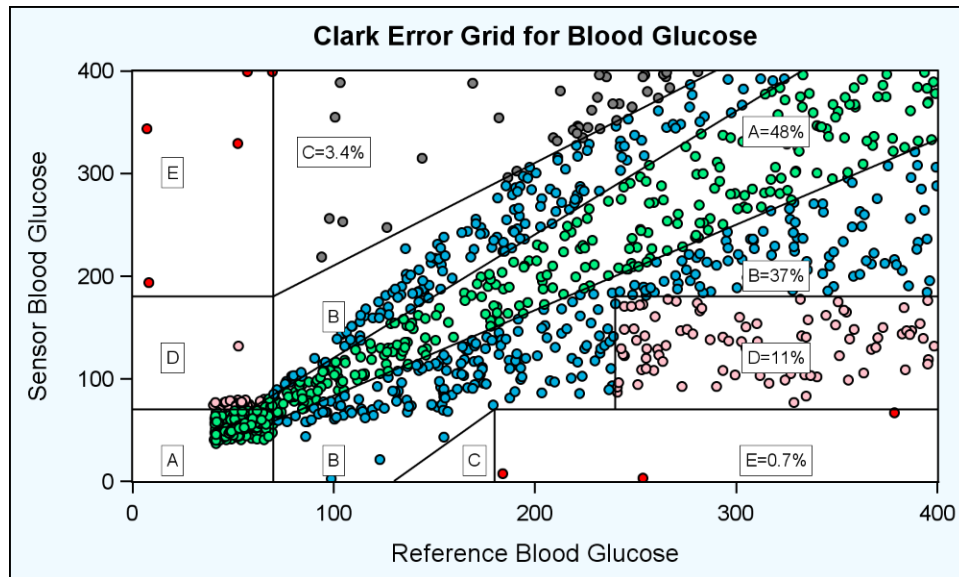
```
title 'Clark Error Grid for Blood Glucose';
proc sgplot data=plotZoneCount noautolegend dattrmap=attrmap;
  scatter x=x y=y / group=zone attrid=A filledoutlinedmarkers
          markerattrs=(symbol=circlefilled size=5);
  series x=rfbg y=sbg / group=id nomissinggroup
         lineattrs=graphdatadefault(color=black) ;
  text x=xl y=yl text=label / backfill fillattrs=(color=white) outline;
  xaxis min=0 max=400 offsetmin=0 offsetmax=0
        label='Reference Blood Glucose';
  yaxis min=0 max=400 offsetmin=0 offsetmax=0
        label='Sensor Blood Glucose';
run;
```

The data for this graph includes the measured and reference glucose level observations, data for zone boundaries and the zone labels, and data for zone labels.

The scatter plot in the program is used to draw the metered glucose values by reference. The series plot is used to display the boundaries of each zone, and the text plot is used to display the zone name. The text plot is optimized for display of textual items in a graph. A discrete attributes map is used to color the markers in each zone appropriately.

4.12.2 Clark Error Grid in Grayscale

The same graph as in Section 4.12.1 is rendered here for a grayscale medium. The key difference in approach is that we need to ensure the correct decoding of the data in the five zones. Here, we have used filled markers for each zone as specified in the STYLEATTRS statement. The marker fill color is set to "White".

Figure 4.12.2 – Clark Error Grid in Grayscale
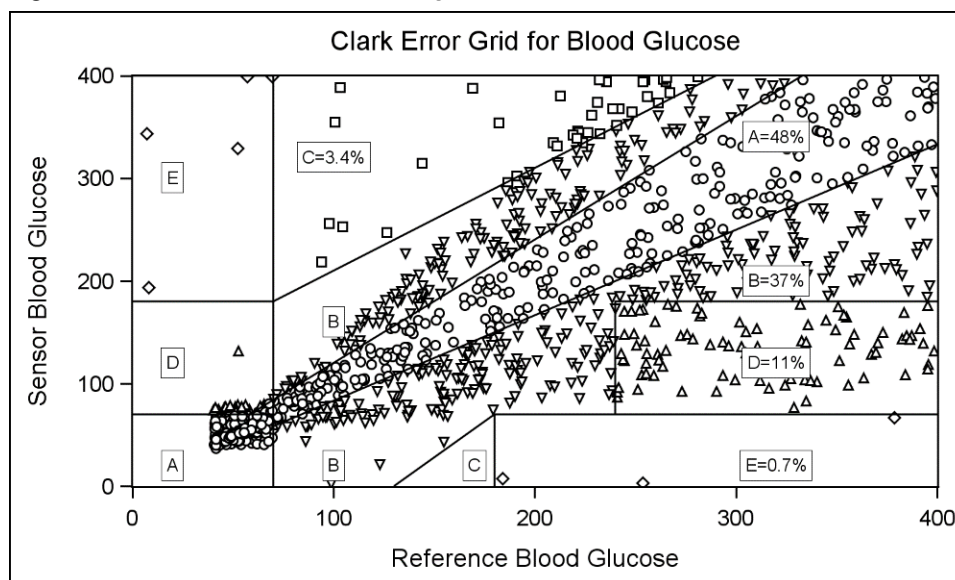
```
ods listing style=journal;
title 'Clark Error Grid for Blood Glucose';
proc sgplot data=plotZoneCount noautolegend;
  styleattrs datasymbols=(trianglefilled circlefilled
             squarefilled diamondfilled triangledownfilled);
  scatter x=x y=y / group=zone attrid=A markerattrs=(size=5)
          filledoutlinedmarkers markerfillattrs=(color=white);
  series x=rfbg y=sbg / group=id nomissinggroup;
  text x=xl y=yl text=label / backfill fillattrs=(color=white) outline;
  xaxis min=0 max=400 offsetmin=0 offsetmax=0
        label='Reference Blood Glucose';
  yaxis min=0 max=400 offsetmin=0 offsetmax=0
        label='Sensor Blood Glucose';
  run;
```

The attribute map is not used here because the zones are clearly marked in the graph itself. It is only helpful to have different markers in each zone, but not necessary. However, if it does become necessary to place the same markers across different graphs for each zone, this can be ensured by using a discrete attribute map.

All axis offsets are set to zero to ensure the zone boundaries touch the axes. This also removes the effect of any offset contributions preferred by the text plot.

Relevant details are shown in the code snippet above. For full details, see Program 4_12.

4.13 The Swimmer Plot

This "swimmer plot" displays the response of the tumor to the study drug over time in months. Each horizontal bar represents one subject in the study.

4.13.1 The Swimmer Plot for Tumor Response over Time

This graph shows the tumor response by subject over time[1]. Each horizontal bar in the graph represents one subject. The inset line indicates complete or partial response with start and end times.

Figure 4.13.1.1 – The Swimmer Plot for Tumor Response over Time
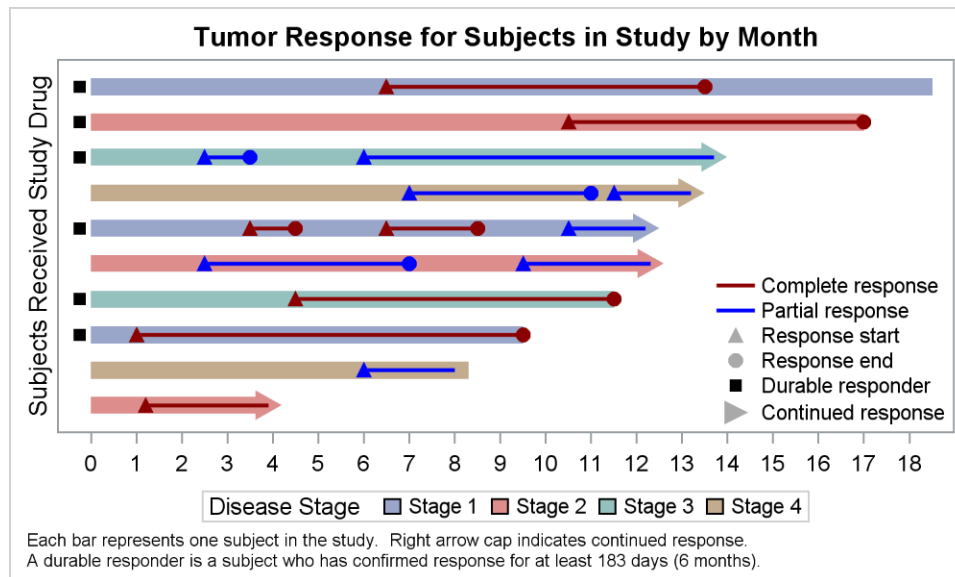

```
title 'Tumor Response for Subjects in Study by Month';
proc sgplot data= swimmer dattrmap=attrmap nocycleattrs;
  highlow y=item low=low high=high / highcap=highcap type=bar group=stage
          fill nooutline name='stage' nomissinggroup transparency=0.3;
  highlow y=item low=startline high=endline / group=status
          name='status' nomissinggroup attrid=statusC;
  scatter y=item x=start / name='s' legendlabel='Response start'
          markerattrs=(symbol=trianglefilled size=8 color=darkgray);
  scatter y=item x=end / name='e' legendlabel='Response end'
          markerattrs=(symbol=circlefilled size=8 color=darkgray);
  scatter y=item x=xmin / name='x' legendlabel='Continued response '
          markerattrs=(symbol=trianglerightfilled
          size=12 color=darkgray);
  scatter y=item x=durable / name='d' legendlabel='Durable responder'
          markerattrs=(symbol=squarefilled size=6 color=black);
  scatter y=item x=start / group=status attrid=status
          markerattrs=(symbol=trianglefilled size=8);
```

```
scatter y=item x=end / group=status attrid=status
        markerattrs=(symbol=circlefilled size=8);
xaxis display=(nolabel) label='Months'
      values=(0 to 20 by 1) valueshint;
yaxis reverse display=(noticks novalues noline)
      label='Subjects Received ...';
keylegend 'stage' / title='Disease Stage';
keylegend 'status' 's' 'e' 'd' / noborder location=inside
          position=bottomright across=1 linelength=20;
run;
```

An arrowhead on the right indicates continuing response. The bar contains durations over which the "Complete" or "Partial" response is indicated, with a start and end time. The disease stage is indicated by the color of the bar, with a legend showing the unique values below the x-axis. An inset is included to decode the different markers in the event bar. A "Durable" response is indicated by the square marker on the left end of the bar.

Note that the start and end points for each response are represented by colored markers inside each event bar. However, the same points are shown in grayscale in the inset table. This is achieved by first plotting the markers in a gray color, and overdrawing those by colored markers using GROUP=Status. The scatter plots that plot the gray markers are the ones that are included in the inset.

Also note the existence of a "right arrow" marker in the inset indicating the continuing event. This is done by including a scatter plot with a right triangle marker in the plot, but the data for this marker is missing. However, it is included in the inset.

The structure of the data set that is needed for the graph is shown below.

Figure 4.13.1.2 – Data Set for Tumor Response Graph

Obs	Item	Stage	Low	High	Highcap	Status	Start	End	Durable	Startline	Endline	Xmin
1	1	Stage 1	0	18.5		Complete response	6.5	13.5	-0.25	6.5	13.5	.
2	2	Stage 2	0	17.0		Complete response	10.5	17.0	-0.25	10.5	17.0	.
3	3	Stage 3	0	14.0	FilledArrow	Partial response	2.5	3.5	-0.25	2.5	3.5	.
4	3		0	14.0	FilledArrow	Partial response	6.0	.	.	6.0	13.7	.
5	4	Stage 4	0	13.5	FilledArrow	Partial response	7.0	11.0	.	7.0	11.0	.
6	4		0	13.5	FilledArrow	Partial response	11.5	.	.	11.5	13.2	.
7	5	Stage 1	0	12.5	FilledArrow	Complete response	3.5	4.5	-0.25	3.5	4.5	.
8	5		0	12.5	FilledArrow	Complete response	6.5	8.5	.	6.5	8.5	.
9	5		0	12.5	FilledArrow	Partial response	10.5	.	.	10.5	12.2	.
10	6	Stage 2	0	12.6	FilledArrow	Partial response	2.5	7.0	.	2.5	7.0	.

Note, although the program for this graph is longer than some other ones, it can be built one part at a time.

- First, plot the full duration from Low to High by Item using a grouped highlow plot with a High Cap and TYPE=BAR. Include this in the outside legend.
- Layer the individual "Response" events from Startline to Endline by Status using a high-low bar with the default line type. Include this in the inset legend.
- Layer the Start and End events in a gray color. Include these in the inset legend.
- Layer the Start and End events again using GROUP=Status.
- Add a scatter plot with missing data to include the "Right Arrow" in the legend.

The Discrete Attribute Map data set contains two maps, one for the colored graph "StatusC", and one for the grayscale graph called "StatusJ". AttrId=StatusC is used in this graph. For full details, see Program 4_12.

4.13.2 The Swimmer Plot for Tumor Response over Time in Grayscale

The tumor response graph is shown in grayscale. The disease stage is shown on the left as we cannot use a color indicator.

Patterned lines are used to draw the response events, and a YAXISTABLE is used to draw the stage labels on the left. ATTRID=StatusJ is used in this graph. For full details, see Program 4_13.

Figure 4.13.2 – The Swimmer Plot for Tumor Response over Time in Grayscale
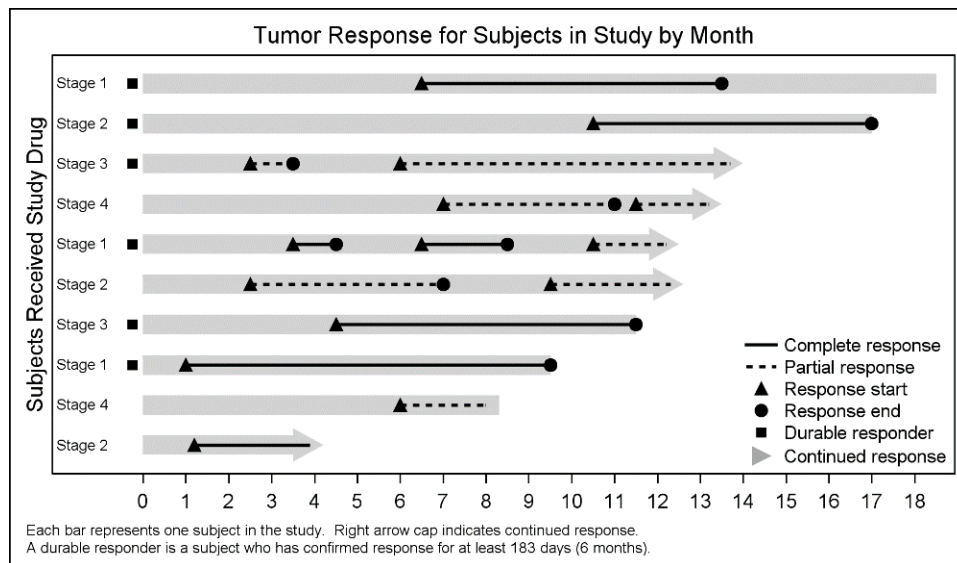

```
ods listing style=journal;
title 'Tumor Response for Subjects in Study by Month';
proc sgplot data= swimmer dattrmap=attrmap nocycleattrs;
```

```
   styleattrs datalinepatterns=(solid shortdash);
   highlow y=item low=low high=high / highcap=highcap type=bar group=stage
           nooutline lineattrs=(color=black) fillattrs=(color=lightgray)
           name='stage' barwidth=1 nomissinggroup fill;
   highlow y=item low=startline high=endline / group=status name='status'
           lineattrs=(thickness=2) nomissinggroup  attrid=statusJ;
   scatter y=item x=start / name='s' legendlabel='Response start'
           markerattrs=(symbol=trianglefilled size=8);
   scatter y=item x=end / name='e' legendlabel='Response end'
           markerattrs=(symbol=circlefilled size=8);
   scatter y=item x=xmin / name='x' legendlabel='Continued response'
           markerattrs=(symbol=trianglerightfilled size=12
           color=darkgray);
   scatter y=item x=durable / name='d' legendlabel='Durable responder'
           markerattrs=(symbol=squarefilled size=6 color=black);
   scatter y=item x=start / group=status attrid=statusJ;
   scatter y=item x=end / group=status attrid=statusJ
   yaxistable stage / location=inside position=left nolabel
              attrid=statusJ;
   xaxis display=(nolabel) label='Months'
         values=(0 to 20 by 1) valueshint;
   yaxis reverse display=(noticks novalues noline)
         label='Subjects Received ...';
   keylegend 'status' 's' 'e' 'd' 'x' / noborder location=inside across=1
           position=bottomright linelength=20;
run;
```

4.14 CDC Chart for Length and Weight Percentiles

The CDC chart for length and weight for boys and girls from birth to 36 months is widely used in pediatric practices to track vital statistics. This graph is shown in Figure 4.14.1, and the entire chart is created using the SGPLOT procedure. The purpose is primarily to evaluate the features of the procedure.

Figure 4.14.1 – CDC Chart for Length and Weight Percentiles
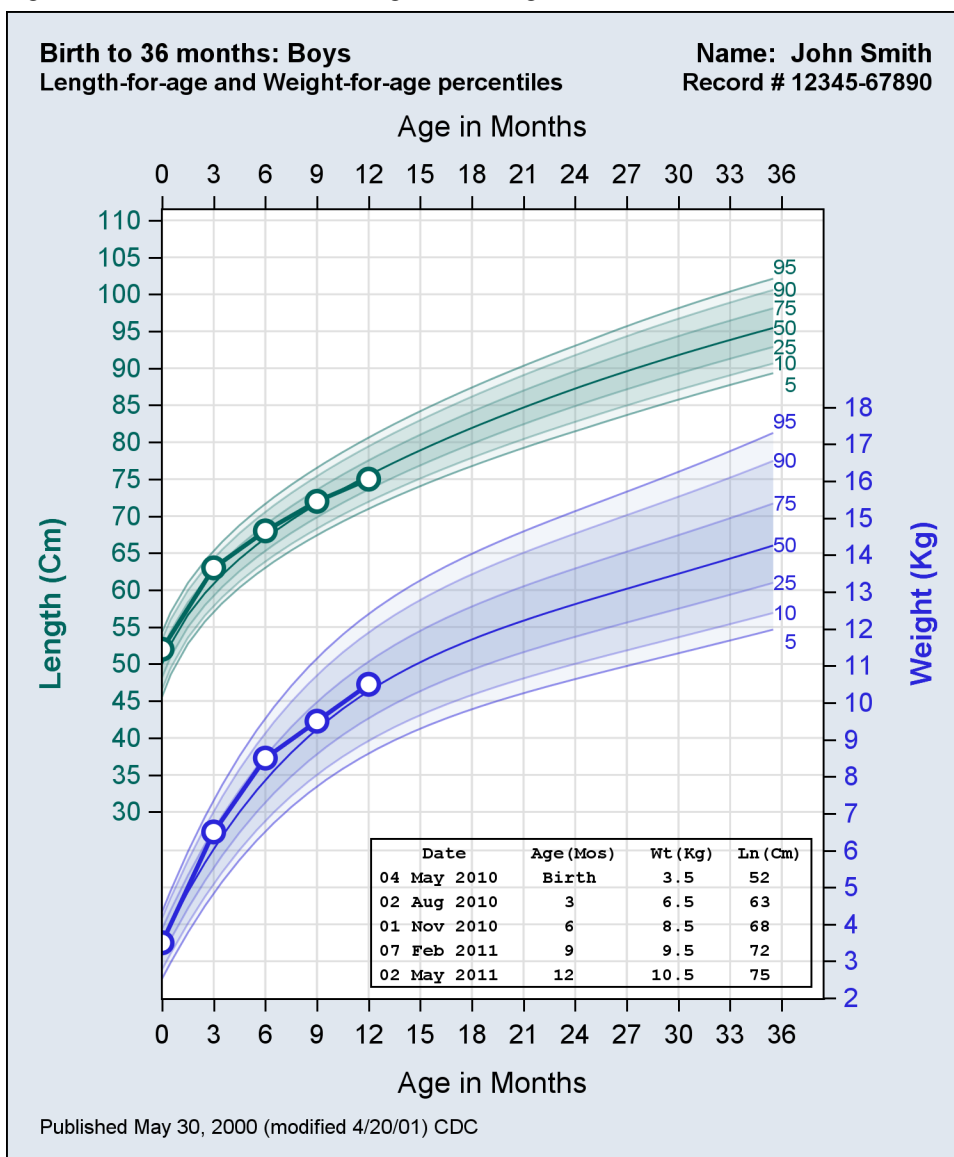
The graph above renders the full CDC chart for Length and Weight Percentiles from the data for one subject. The original graph was a bit taller, but I shrank it to fit this page. The data that is required is created by appending the CDC percentile data with the historical data for one subject. The CDC data is included in the file named "4_14_CDC_Cleaned.csv".

The CDC data for the percentile curves is shown below. Only a few of the observations are displayed to conserve space. Also, the data contains all the columns for 5, 10, 25, 50, 75, 90, and 95 percentiles, but only a few columns are included to fit in the space.

Figure 4.14.2 – Data for CDC Chart

Sex	Agemos	W5	W25	W75	W95	H5	H25	H75	H95
1	0	2.52690402	3.150611082	3.879076559	4.34029274	45.5684091	48.189373814	51.771257485	54.307211972
1	0.5	2.964655655	3.597395573	4.387422565	4.910130108	48.558092059	50.979188895	54.440543134	56.999077373
1	1.5	3.774848862	4.428872952	5.327327567	5.967101615	52.726106587	54.979104409	58.350594078	60.964653792

The historical data for the subject is appended at the bottom of the curve data, using the column names Sex, Age, Height, and Length, as shown below.

Figure 4.14.3 – Data for CDC Chart

Obs	Sex	Agemos	W5	W50	W95	H5	H50	H95	Age	Height	Weight
36	Male	34.5	11.86229971	14.1150324	17.10619066	88.703007448	94.808229231	101.43177049	.	.	.
37	Male	35.5	11.98045644	14.25779618	17.30646132	89.332418366	95.446369813	102.11744722	.	.	.
38	Male	.	.	.	.	.	.	.	0	52	3.5
39	Male	.	.	.	.	.	.	.	3	63	6.5

```
title j=l h=9pt 'Birth to 36 months: Boys' j=r "Name:  John Smith";
title2 j=l h=8pt "Length-for-age and Weight-for-age percentiles" j=r
"Record # 12345-67890";
footnote j=l h=7pt "Published May 30, 2000 (modified 4/20/01) CDC";
proc sgplot data=Chart_Patient noautolegend;
  where sex=1;
    refline 3 4 5 6 / axis=y2 lineattrs=graphgridlines;

    /*--Curve bands--*/
    band x=agemos lower=w5  upper=w95 / y2axis fillattrs=graphdata1
        transparency=0.9;
    band x=agemos lower=w10 upper=w90 / y2axis fillattrs=graphdata1
        transparency=0.8;
    band x=agemos lower=w25 upper=w75 / y2axis fillattrs=graphdata1
        transparency=0.8;

    /*--Curves--*/
    series x=agemos y=w5 / y2axis lineattrs=graphdata1 transparency=0.5;
    series x=agemos y=w10 / y2axis lineattrs=graphdata1 transparency=0.7;
    series x=agemos y=w25 / y2axis lineattrs=graphdata1 transparency=0.7;
    series x=agemos y=w50 / y2axis x2axis lineattrs=graphdata1;
    series x=agemos y=w75 / y2axis lineattrs=graphdata1 transparency=0.7;
```

```
  series x=agemos y=w90 / y2axis lineattrs=graphdata1 transparency=0.7;
  series x=agemos y=w95 / y2axis lineattrs=graphdata1 transparency=0.5;
```

The program that is required to draw all the elements of this graph is long, but easy to understand.
So, I have shown it in parts across the following pages. The first part of the program is shown
above, with titles, footnotes, and percentile curves for Weight. The bands are drawn with three
transparent overlays to create the appearance of color gradation. The curves are overlaid on the
bands.

```
  /*--Curve labels--*/
  text x=agemos y=w5  text=l5  / y2axis textattrs=graphdata1
               position=right;
  text x=agemos y=w10 text=l10 / y2axis textattrs=graphdata1
               position=right;
  text x=agemos y=w25 text=l25 / y2axis textattrs=graphdata1
               position=right;
  text x=agemos y=w50 text=l50 / y2axis textattrs=graphdata1
               position=right;
  text x=agemos y=w75 text=l75 / y2axis textattrs=graphdata1
               position=right;
  text x=agemos y=w90 text=l90 / y2axis textattrs=graphdata1
               position=right;
  text x=agemos y=w95 text=l95 / y2axis textattrs=graphdata1
               position=right;

  /*--Patient datas--*/
  series x=age y=weight / lineattrs=graphdata1(thickness=2)
         y2axis markers markerattrs=(symbol=circlefilled size=11)
         filledoutlinedmarkers markerfillattrs=(color=white)
         markeroutlineattrs=graphdata1(thickness=2);
```

The code section above draws the curve labels for the percentile curves on the right. This is
overlaid by the historical subject weight data as a series plot. The code for Height is shown below.

```
  /*--Curve bands--*/
  band x=agemos lower=h5  upper=h95 / fillattrs=graphdata3
  transparency=0.9;
  band x=agemos lower=h10 upper=h90 / fillattrs=graphdata3
  transparency=0.8;
  band x=agemos lower=h25 upper=h75 / fillattrs=graphdata3
  transparency=0.8;

  /*--Curves--*/
  series x=agemos y=h5 / lineattrs=graphdata3(pattern=solid)
         transparency=0.5;
  series x=agemos y=h10 /lineattrs=graphdata3(pattern=solid)
         transparency=0.7;
  series x=agemos y=h25 /lineattrs=graphdata3(pattern=solid)
         transparency=0.7;
  series x=agemos y=h50 /lineattrs=graphdata3(pattern=solid) x2axis;
  series x=agemos y=h75 /lineattrs=graphdata3(pattern=solid)
```

```
          transparency=0.7;
   series x=agemos y=h90 /lineattrs=graphdata3(pattern=solid)
          transparency=0.7;
   series x=agemos y=h95 /lineattrs=graphdata3(pattern=solid)
   transparency=0.5;

   /*--Curve labels--*/
   text x=agemos y=h5  text=l5  / textattrs=graphdata3
        position=bottomright;
   text x=agemos y=h10 text=l10 / textattrs=graphdata3 position=right;
   text x=agemos y=h25 text=l25 / textattrs=graphdata3 position=right;
   text x=agemos y=h50 text=l50 / textattrs=graphdata3 position=right;
   text x=agemos y=h75 text=l75 / textattrs=graphdata3 position=right;
   text x=agemos y=h90 text=l90 / textattrs=graphdata3 position=right;
   text x=agemos y=h95 text=l95 / textattrs=graphdata3 position=topright;

   /*--Patient datas--*/
   series x=age y=height /
          lineattrs=graphdata3(pattern=solid thickness=2)
          markers markerattrs=(symbol=circlefilled size=11)
          filledoutlinedmarkers markerfillattrs=(color=white)
          markeroutlineattrs=graphdata3(thickness=2);
```

The Height (Length) and Weight data ranges are different, and these need to be plotted with
different vertical scales and axis details. We can do that by using two separate Y-axes for each
column. Here we used the Y2AXIS for Weight and YAXIS for Height. This breaks the link
between the two variables scales, thus allowing us to draw the Height and Weight curves and data
independently.

```
   /*--Table--*/
   inset "    Date      Age(Mos)   Wt(Kg)  Ln(Cm)"
         "04 May 2010    Birth      3.5     52"
         "02 Aug 2010      3        6.5     63"
         "01 Nov 2010      6        8.5     68"
         "07 Feb 2011      9        9.5     72"
         "02 May 2011     12       10.5     75" / border
         textattrs=(family='Courier' size=6 weight=bold)
         position=bottomright;

   xaxis grid offsetmin=0  integer values=(0 to 36 by 3);
   x2axis grid offsetmin=0 integer values=(0 to 36 by 3);
   yaxis  grid offsetmin=0.25 offsetmax=0.0 label="Length (Cm)" integer
         values=(30 to 110 by 5) labelattrs=graphdata3(weight=bold)
         valueattrs=graphdata3;
        y2axis offsetmin=0.0 offsetmax=0.25 label="Weight (Kg)" integer
         values=(2 to 18 by 1) labelattrs=graphdata1(weight=bold)
         valueattrs=graphdata1;
   run;
```

Note the options on the YAXIS and the Y2AXIS statements. The Y2AXIS has
OFFSETMAX=0.25, which means that all items that are associated with it are displayed only in the

lower 75% of the graph height. This causes all the "Weight" related items and the axis (drawn in blue) to be drawn in the lower part.

Similarly, the YAXIS has OFFSETMIN =0.25, so all the "Height or Length" related items are drawn in the upper part of the graph. More importantly, the scaling for each axis is independent, allowing us to draw different tick values on the axes. To make the graph easier to read, we have taken care to position the Y grid lines so that they line up with the values on each side.

The program snippet above also shows how we can include the historical data as a tabular display in the chart for easy reference. I have used the INSET statement to create a tabular display. Although the values here are hardcoded, we can use macro variables assigned from the DATA step.

Relevant details are shown in the code snippet above. For full details, see Program 4_14.

4.15 Summary

The graphs discussed in this chapter represent a large fraction of the graphs commonly used in the clinical trials industry and in Health and Life Sciences in general. Most of these are "single-cell" graphs where the main data is displayed in one cell in the middle, along with other information.

In this chapter, we have used the SAS 9.4 SGPLOT procedure, which provides you with a large selection of plot statements that can be used to create many graphs on their own. Many of these plot statements can be combined in creative ways to create almost any graph that might be needed. Some new statements, such as the axis tables, and options newly added to SAS 9.4 make it much easier to create these graphs.

The SG Annotate facility further enhances your ability to create custom graphs using the SGPLOT procedure. Although we have not used it in these examples, annotation can be very useful to add some custom details that are otherwise hard to do using plot layers.

Group attributes such as colors or marker symbol shapes can be assigned by specific group values using the Attribute Map feature. This ensures that attributes are correctly mapped regardless of the data order, or whether some groups are present or not.

---

[1] My graph is based on ideas presented in a paper. See Phillips, Stacey D. 2014. "Swimmer Plot: Tell a Graphical Story of Your Time to Response Data Using PROC SGPLOT." *Proceedings of the Pharmaceutical Industry SAS Users Group (PharmaSug) 2014 Conference*. San Diego, CA: SAS Institute Inc. Available at http://www.pharmasug.org/2014-proceedings.html.

Customizing the Kaplan-Meier Survival Plot

Excerpt from *SAS/STAT 14.2 User's Guide*

The Kaplan-Meier plot displays patient survival over time for one or more groups of patients. The Kaplan-Meier plot is heavily used in medical, pharmaceutical, and life-sciences research. It is a plot that researches can easily customize in a variety of ways. This chapter shows you how to customize the Kaplan-Meier plot through a series of examples. It discusses four types of examples: specifying procedure options, modifying graph templates by using macro variables, modifying graph templates by using macros, and changing styles. Each example is designed to be small, simple, self-contained, and easy to copy and use "as is" or with minor modifications.

For more SAS/STAT chapters with ODS and ODS Graphics examples, please visit these resources.

http://support.sas.com/documentation/onlinedoc/stat/142/ods.pdf
http://support.sas.com/documentation/onlinedoc/stat/142/odsgraph.pdf
http://support.sas.com/documentation/onlinedoc/stat/142/templt.pdf

Warren F. Kuhfeld, Analytical Solutions Manager at SAS, supports SAS procedures, writes new procedure code, and provides leadership in the usage of ODS and ODS Graphics. In addition, he writes and maintains ODS and ODS Graphics documentation as well as a family of macros for experimental design and marketing research. A SAS user since 1979, Warren received his Ph.D. in psychometrics from the University of North Carolina at Chapel Hill. He has presented at SUGI, SAS Global Forum, SEUGI, SAS France Forum, and SAS Belux Forum. He has also been a frequent presenter at the American Marketing Association's Advanced Research Techniques Forum.

support.sas.com/kuhfeld

Chapter 23
Customizing the Kaplan-Meier Survival Plot

Contents

Overview

The LIFETEST procedure is a nonparametric procedure for analyzing survival data. You can use PROC LIFETEST to compute the Kaplan-Meier curve (1958), which is a nonparametric maximum likelihood estimate of the survivor function. The Kaplan-Meier plot (also called the product-limit survival plot) is a popular tool in medical, pharmaceutical, and life sciences research. The Kaplan-Meier plot contains step functions that represent the Kaplan-Meier curves of different samples (strata). The Kaplan-Meier plot has many other features that you can add or change through procedure options, graph templates, and style templates. This chapter explores these features in detail but does not explain how to interpret the graphs or the underlying analysis. For more information about PROC LIFETEST and the Kaplan-Meier plot, see Chapter 71, "The LIFETEST Procedure."

This chapter shows you how to modify the Kaplan-Meier plot through a series of examples. It discusses four types of examples: specifying procedure options, modifying graph templates by using macro variables, modifying graph templates by using macros, and changing styles. Most examples do not go into detail about the tools that underlie the template changes. Each example is designed to be small, simple, self-contained, and easy to copy and use "as is" or with minor modifications. Subsequent sections provide more details about the macro variables and macros that are used to modify the graph templates. You can use the simple examples to make a wide variety of changes without reading or understanding the detailed descriptions at the end of this chapter.

Statistical procedures produce tables by using the Output Delivery System (ODS) and produce graphs by using ODS Graphics. Procedures produce graphs as automatically as they produce tables, and graphs and tables are integrated in the ODS output. Graphs that are produced by ODS Graphics are controlled by options, the data object (the matrix of information that is graphed), a style template, and a graph template. A style template is a SAS program that controls the overall appearance of graphs, including colors, line and marker styles, sizes, fonts, and so on. A graph template is a SAS program, written in the Graph Template Language

(GTL), that provides a detailed specification of the layout and contents of each graph. Each graph that is created when ODS Graphics is enabled is controlled by a graph template.[1]

If you want to modify a graph template, you usually use the TEMPLATE procedure to display the template of interest, and then you copy it into your editor, modify it, and submit it to SAS to compile. Then, when you run your procedure, it uses the new template. The PROC LIFETEST survival plot is the only plot in SAS for which you have another alternative available for template modification. SAS provides the survival plot templates in a series of macros and macro variables that are modular and easier to modify than the original templates. This chapter provides numerous examples of using these macros and macro variables.

The data that are used in this chapter come from 137 bone marrow transplant patients in a study by Klein and Moeschberger (1997) and are available in the BMT data set in the Sashelp library. At the time of transplant, each patient is classified in one of three risk categories: ALL (acute lymphoblastic leukemia), AML (acute myelocytic leukemia)–Low Risk, and AML–High Risk. The endpoint of interest is the disease-free survival time, which is the time in days until death, relapse, or the end of the study. The variable Group represents the patient's risk category, the variable T represents the disease-free survival time, and the variable Status is the censoring indicator. A status of 1 indicates an event time, and a status of 0 indicates a censored time.

Controlling the Survival Plot by Specifying Procedure Options

This section provides a series of examples that use ODS Graphics and the PLOTS= option in the PROC LIFETEST statement to control the appearance of the survival plot. Other examples use formats and the ORDER= option to control the order of the groups.

Enabling ODS Graphics and the Default Kaplan-Meier Plot

You can use the following statements to enable ODS Graphics and run PROC LIFETEST:

```
ods graphics on;

proc lifetest data=sashelp.BMT;
   time T * Status(0);
   strata Group;
run;
```

ODS Graphics is enabled for this step and all subsequent steps until it is disabled. ODS Graphics remains enabled throughout the examples in this chapter.

You specify in the TIME statement that the disease-free survival time is recorded in the variable T. You further specify that the variable Status indicates censoring and 0 indicates a censored time. Separate survivor functions are displayed for each group in the Group variable, which you specify in the STRATA statement.

The plot in Figure 23.1 consists of three step functions, one for each of the three groups of patients. The plot shows that patients in the AML–Low Risk group have longer disease-free survival than patients in the ALL and AML–High Risk groups.

---

[1]ODS Graphics might or might not be enabled by default. ODS Graphics is usually enabled by default in the SAS windowing environment and disabled when you invoke SAS in other ways. However, these defaults can be changed in a number of ways. ODS Graphics is enabled in the first example in this chapter by the ODS GRAPHICS ON statement and remains enabled throughout the chapter.

Figure 23.1 Default Kaplan-Meier Plot
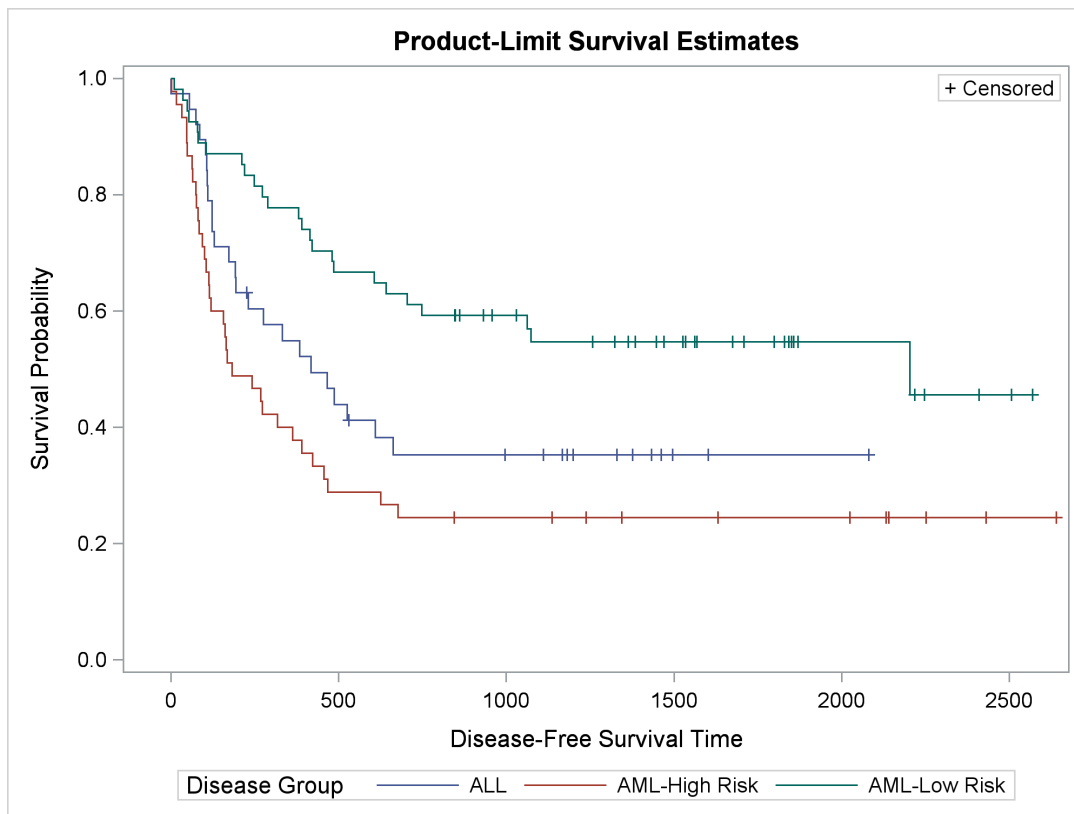
The following step, which explicitly specifies the default PLOTS=SURVIVAL option, is equivalent to the preceding step:

```
proc lifetest data=sashelp.BMT plots=survival;
   time T * Status(0);
   strata Group;
run;
```

The PLOTS= option enables you to control the graphs that a procedure produces. You can use it to request nondefault graphs and specify options for some graphs. You can specify graph names (PLOTS=SURVIVAL), graph options (PLOTS=SURVIVAL(ATRISK OUTSIDE)), and suboptions (PLOTS=SURVIVAL(ATRISK OUTSIDE(0.15))). The PLOTS= option is described in the section "PROC LIFETEST Statement" on page 5215 in Chapter 71, "The LIFETEST Procedure."

Individual Survival Plots

You can use the STRATA=INDIVIDUAL option to request individual survival plots. By default, the STRATA=OVERLAY option produces the plot of overlaid step functions displayed in Figure 23.1. You can run the same analysis but request the results in three separate graphs, one per patient group, as follows:

```
proc lifetest data=sashelp.BMT plots=survival(strata=individual);
   time T * Status(0);
   strata Group;
run;
```

The first of the three survival plots is displayed in Figure 23.2. To conserve space, the other graphs are not displayed.

Figure 23.2 One of Three Individual Plots
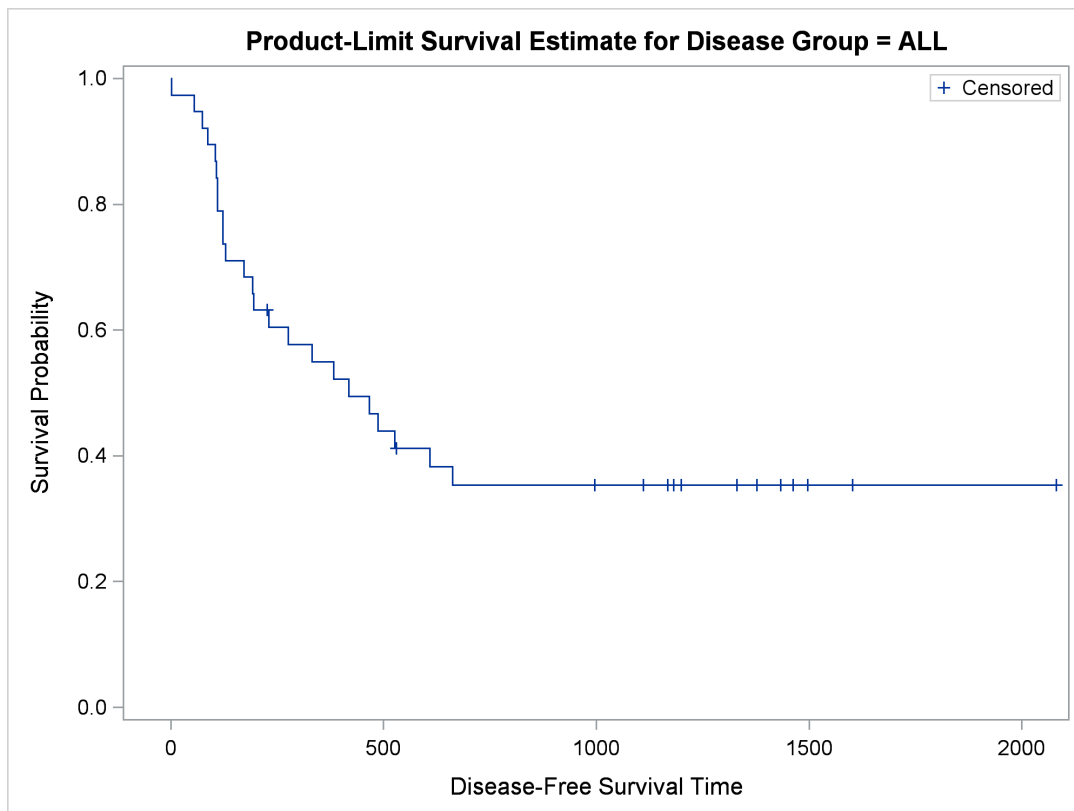
You can use the STRATA=PANEL option as follows to display the results in separate panels of a single graphical display:

```
proc lifetest data=sashelp.BMT plots=survival(strata=panel);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.3.

Figure 23.3 Individual Plots Displayed in a Panel
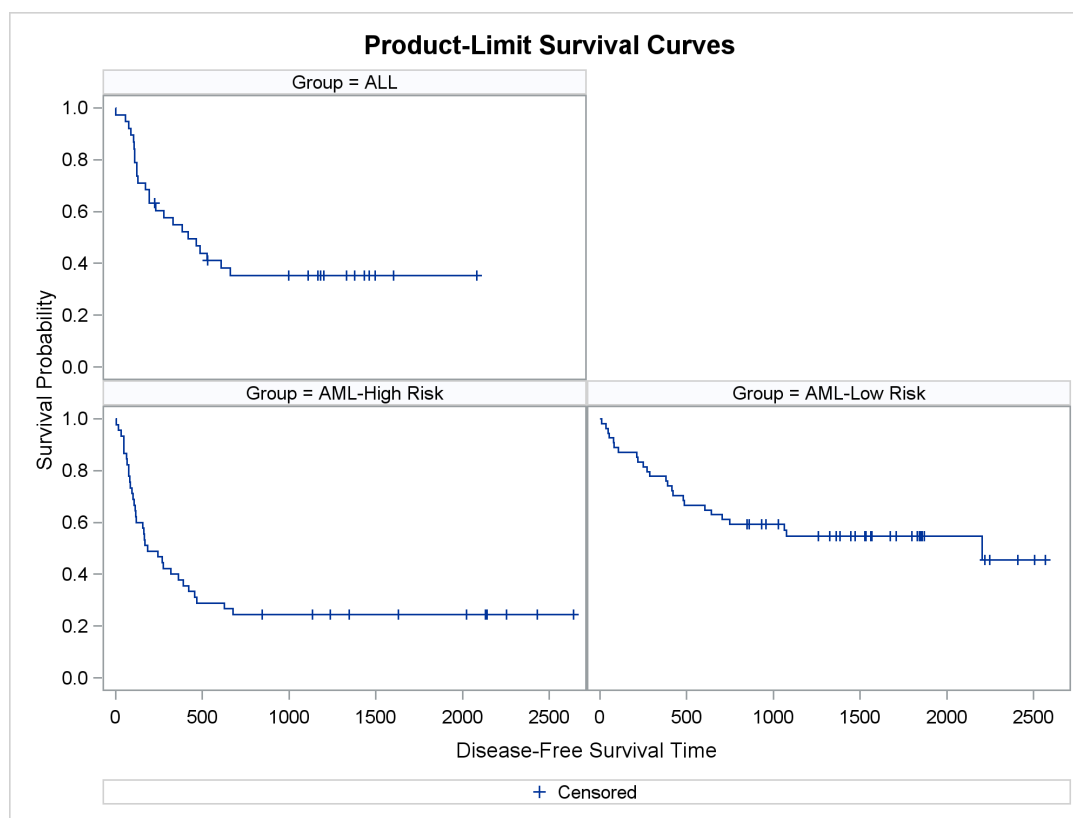
The rest of this chapter discusses overlaid plots such as the one displayed in Figure 23.1.

Hall-Wellner Confidence Bands and Homogeneity Test

You can use the following statements to add Hall-Wellner confidence bands (Hall and Wellner 1980) to Figure 23.1 and display the *p*-value from a test that the strata are homogeneous:

```
proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.4. The Hall-Wellner confidence bands extend to the last event times. The small *p*-value supports rejecting the hypothesis that the groups are homogeneous.

Figure 23.4 Confidence Bands and Homogeneity Test
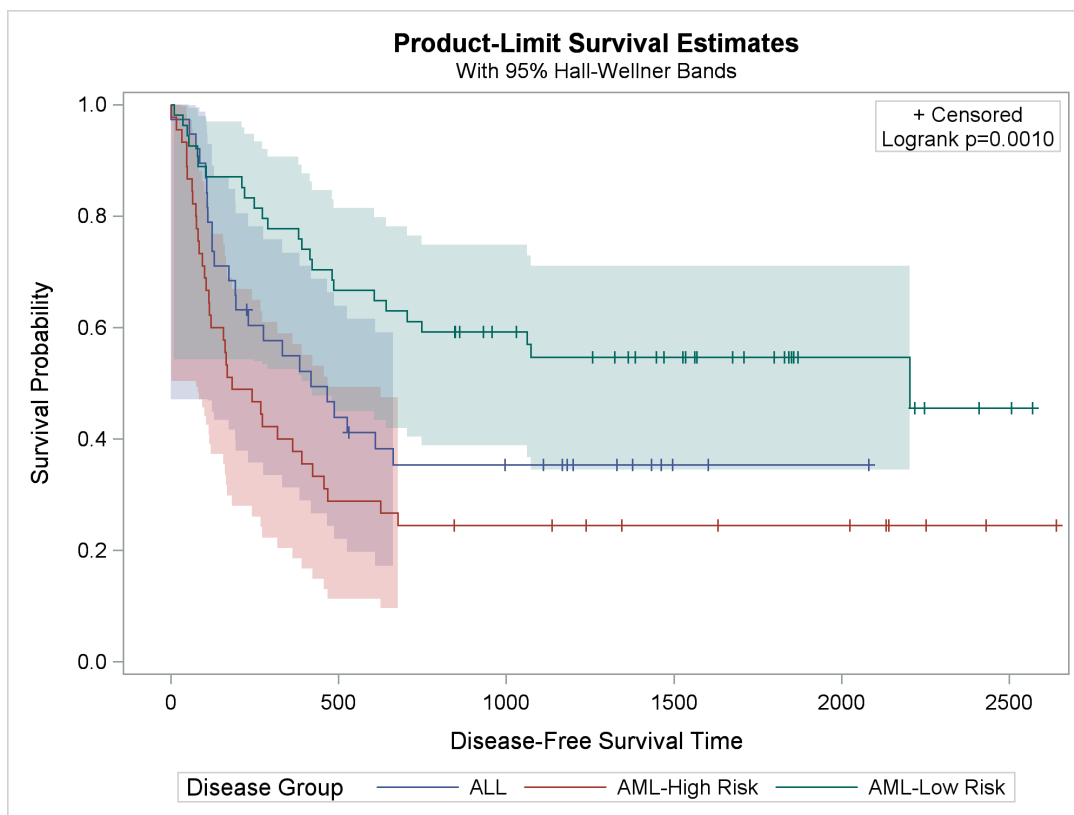
Equal-Precision Bands

You can use the following statements to add equal-precision bands to the plot:

```
proc lifetest data=sashelp.BMT plots=survival(cb=ep test);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.5.

Figure 23.5 Equal-Precision Bands
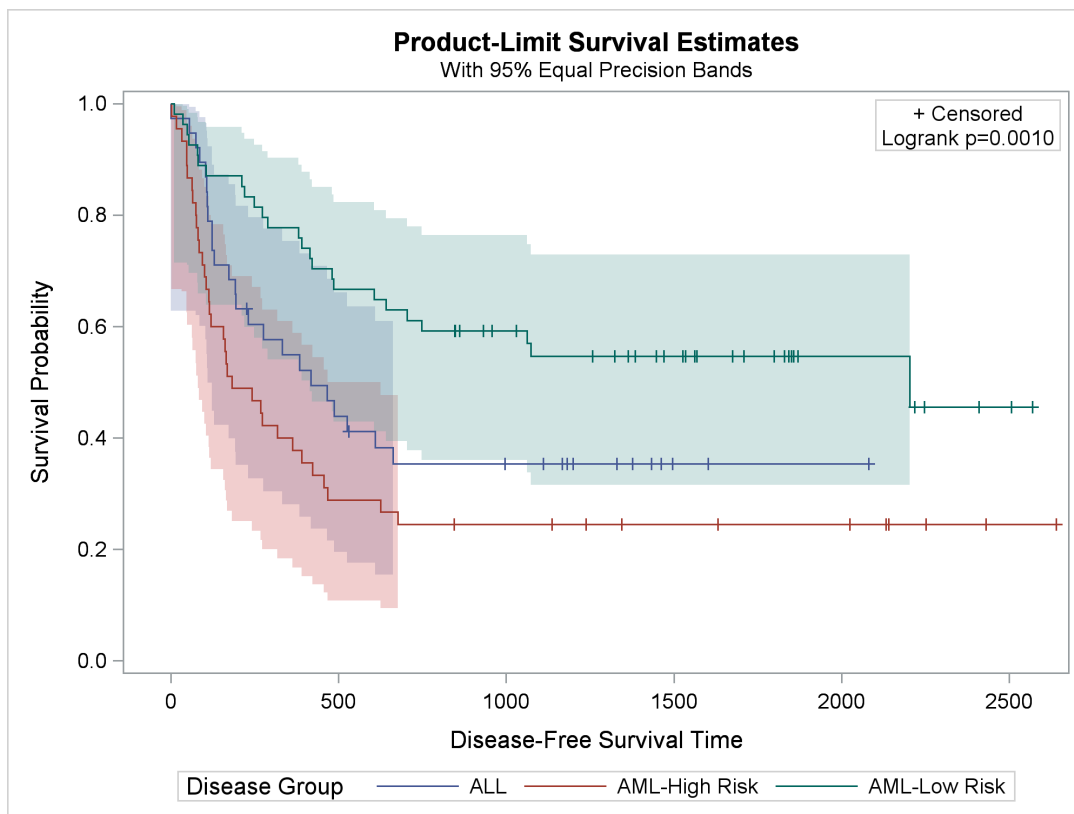
You can use the following statements to add both Hall-Wellner and equal-precision bands to the plot:

```
proc lifetest data=sashelp.BMT plots=survival(cb=all test);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.6.

Figure 23.6 Hall-Wellner and Equal-Precision Bands
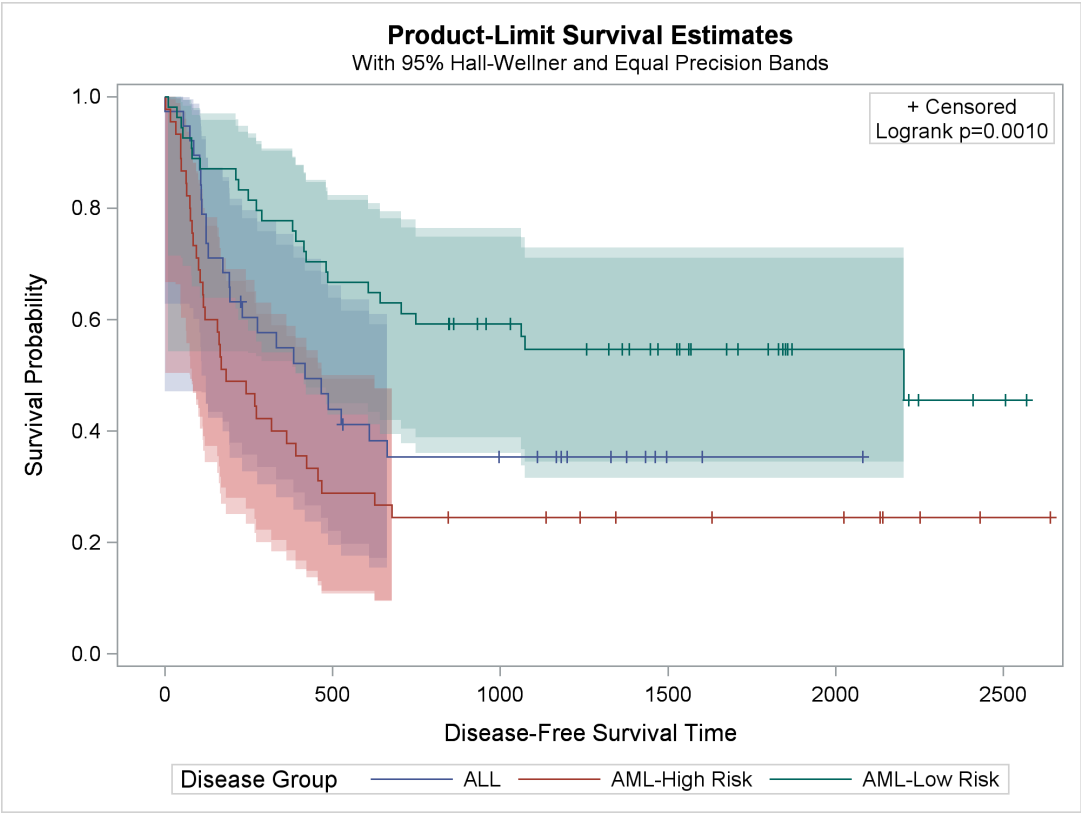
Displaying the Patients-at-Risk Table inside the Plot

You can add the patients-at-risk table to the Kaplan-Meier plot as follows:

```
proc lifetest data=sashelp.BMT plots=survival(cb=hw test atrisk);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.7. By default, the at-risk table is displayed inside the body of the plot. This table shows the number of patients who are at risk for each group for each of the different times. For these data, the default survival times at which at-risk values are displayed are 0 to 2500 by 500. You will see how to specify other values in subsequent examples.

Figure 23.7 At-Risk Table inside the Plot
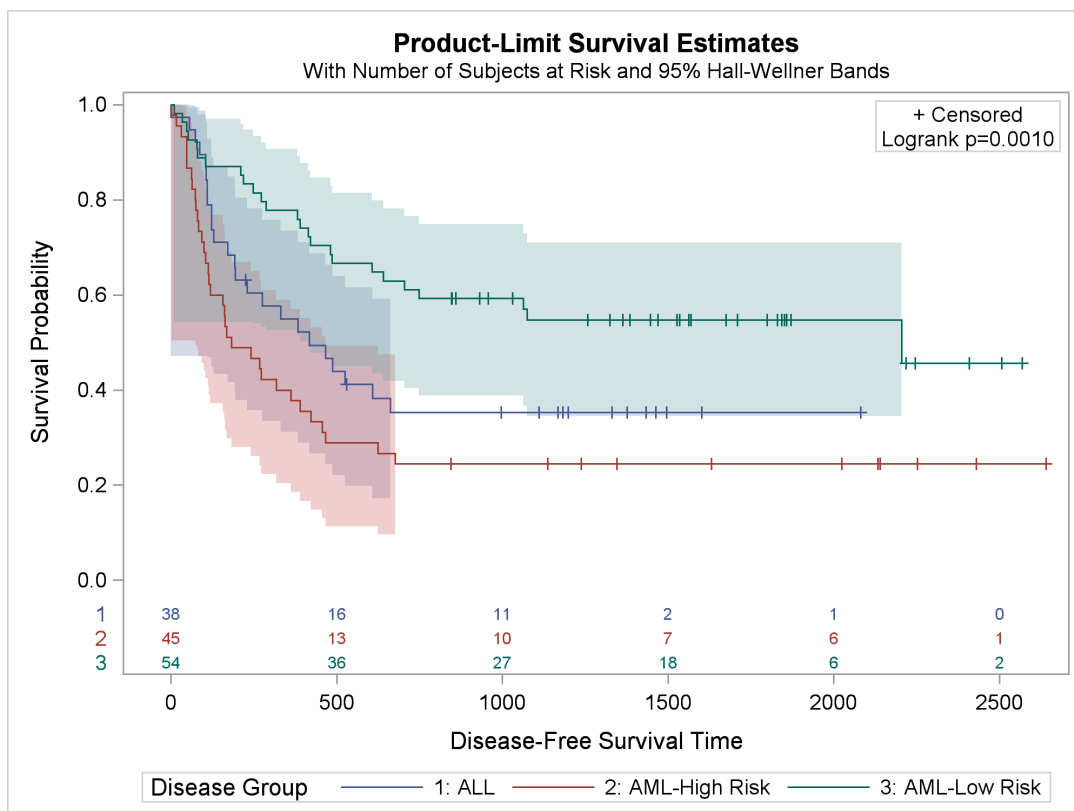
The group labels for the at-risk table are group numbers, and these numbers appear in the legend. Numbers are used rather than the actual labels because the length of the longest label (13) is greater than the default that is set by the maximum label length option (MAXLEN=12). You can display labels rather than the group numbers by specifying a MAXLEN= value equal to the maximum group label length as follows:

```
proc lifetest data=sashelp.BMT plots=survival(cb=hw test atrisk(maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.8. The legend entries and the order of the rows in the at-risk table correspond to the sort order of the values of the Group variable.

Figure 23.8 At-Risk Table with Labels
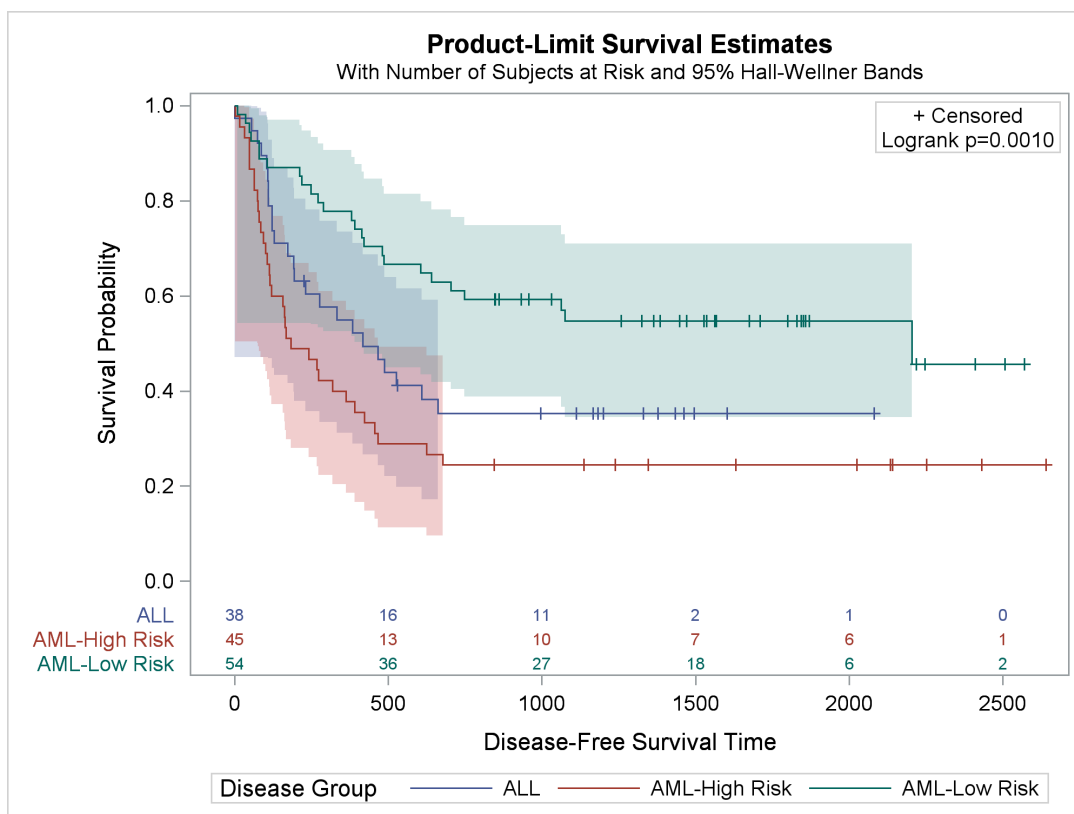
Displaying the Patients-at-Risk Table outside the Plot

You can use the PLOTS=SURVIVAL(OUTSIDE) option to display the at-risk table outside the body of the plot. The option OUTSIDE(0.15) reserves 15% of the vertical graph window for the at-risk table. This example illustrates that the PLOTS= option has options nested within options and options nested within those nested options. The following step produces the plot in Figure 23.9:

```
proc lifetest data=sashelp.BMT
              plots=survival(atrisk(maxlen=13 outside(0.15)));
   time T * Status(0);
   strata Group;
run;
```

Figure 23.9 Moving the At-Risk Table outside the Plot
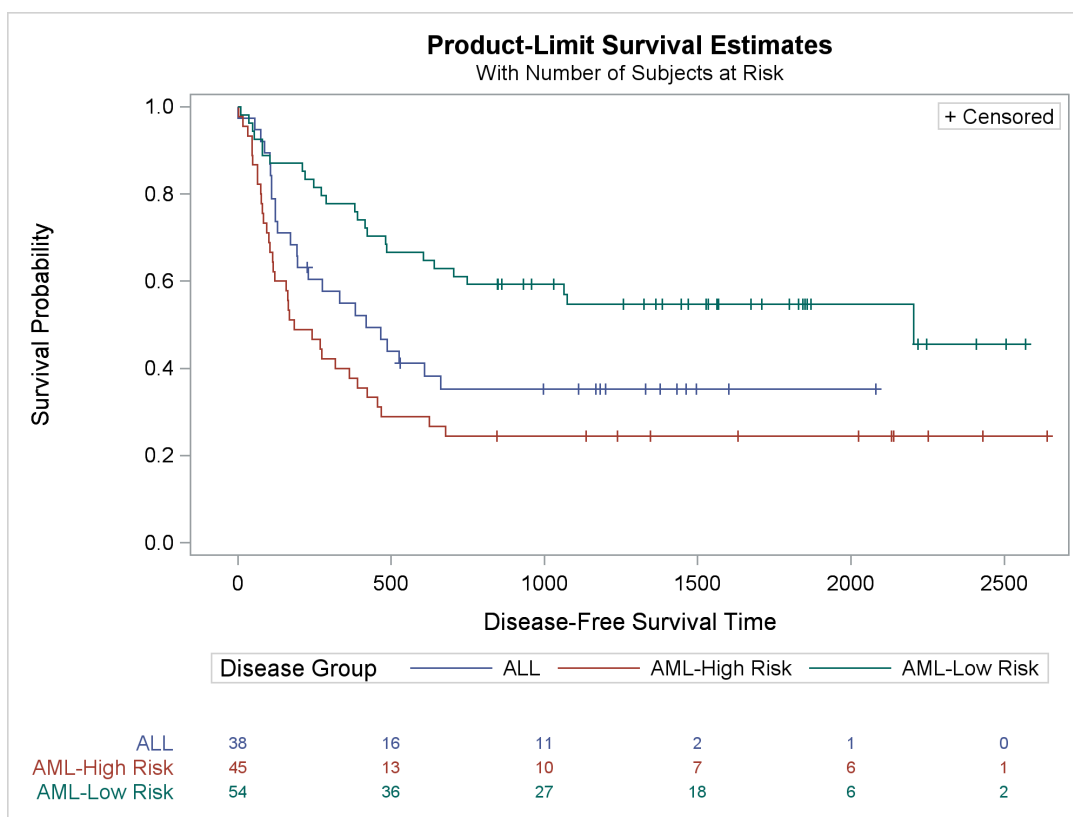
Modifying At-Risk Table Times

The following step explicitly controls the time values at which the at-risk values are displayed by using the PLOTS=SURVIVAL(ATRISK=0 TO 3000 BY 1000) option:

```
proc lifetest data=sashelp.BMT
               plots=survival(atrisk(maxlen=13 outside)=0 to 3000 by 1000);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.10.

Figure 23.10 Specifying At-Risk Values
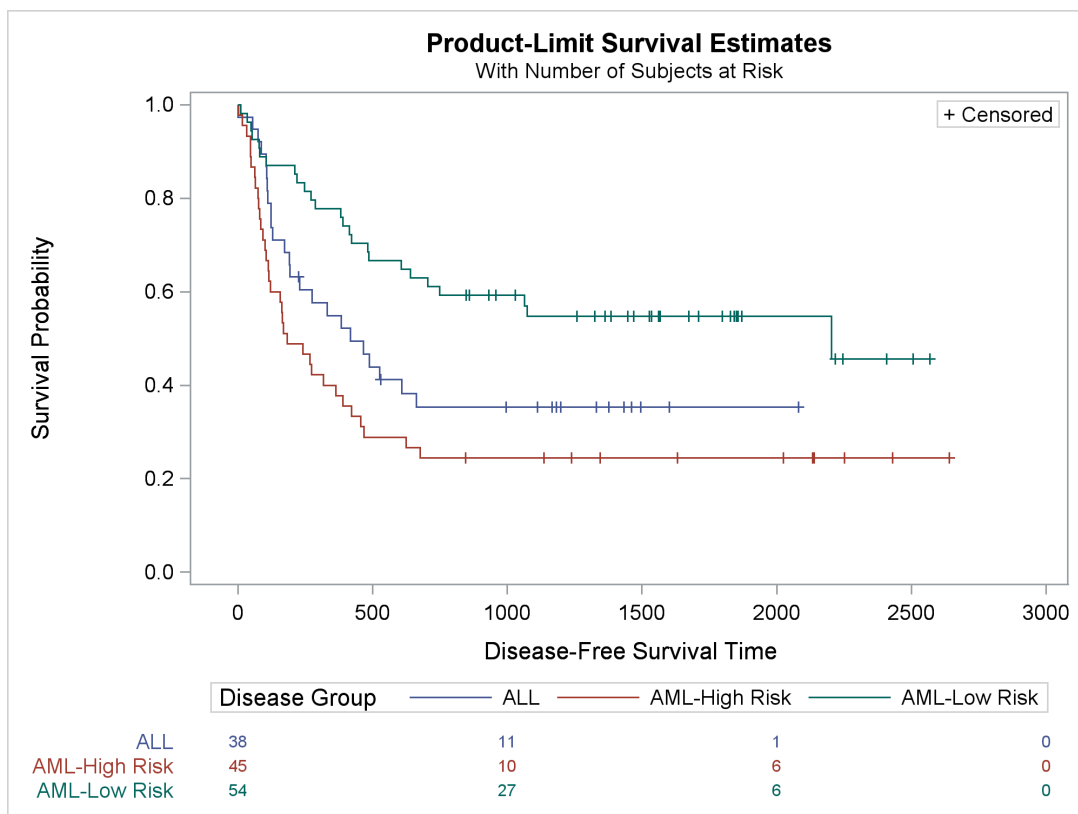
You can specify at-risk values that do not correspond to the original time axis tick marks. You can use the PLOTS=SURVIVAL(ATRISK(ATRISKTICK)) option to add tick marks that correspond to the specified at-risk values:

```
proc lifetest data=sashelp.BMT plots=survival(atrisk
   (atrisktick maxlen=13 outside)=0 500 750 1000 1250 1500 1750 2000 2500);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.11.

Figure 23.11 Controlling At-Risk Tick Marks
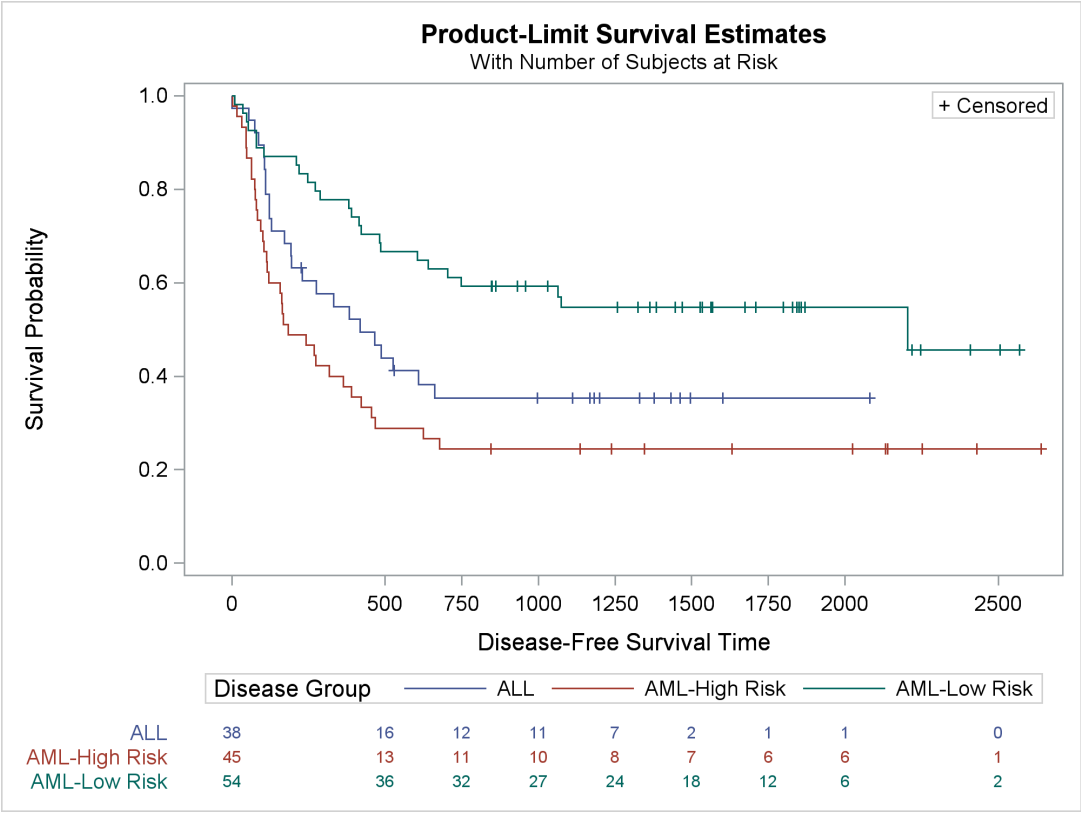
You can display tick values only at those times that are given in the ATRISK= list:

```
proc lifetest data=sashelp.BMT plots=survival(atrisk
   (atrisktickonly maxlen=13 outside)=0 1250 2500);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.12.

Figure 23.12 Controlling At-Risk Tick Marks

	0	1250	2500
ALL	38	7	0
AML-High Risk	45	8	1
AML-Low Risk	54	24	2
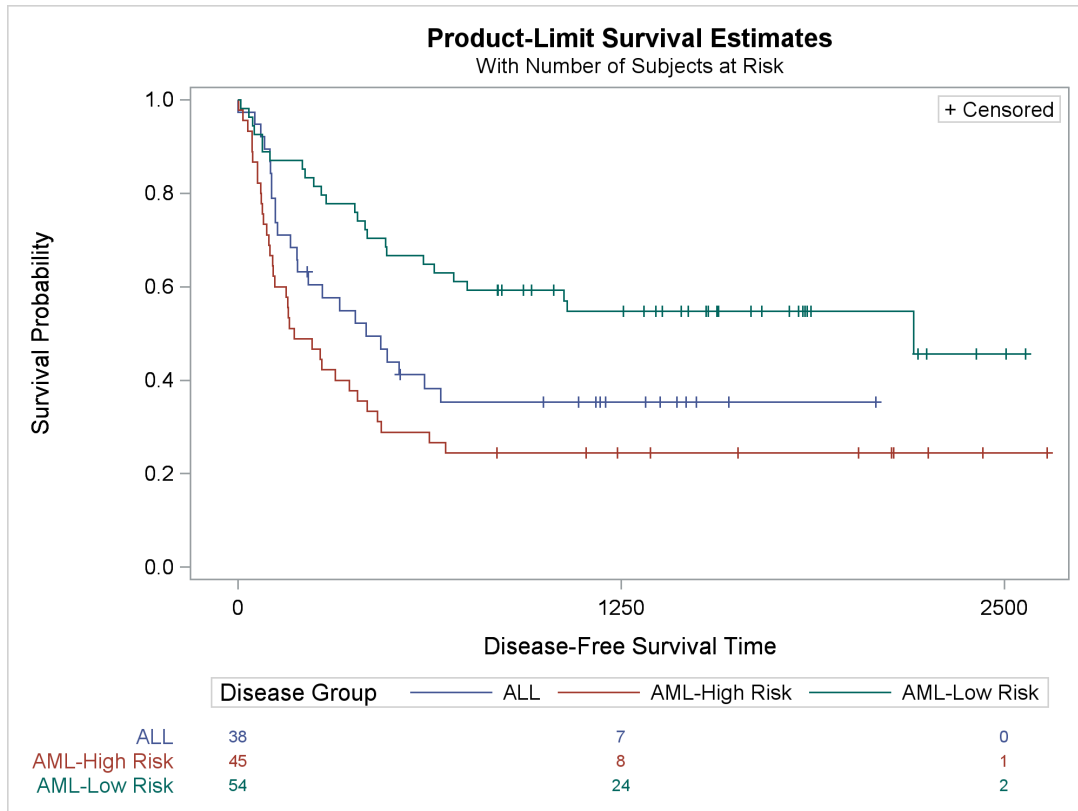
Reordering the Groups

You can change the order of the legend entries by first changing each original group value to a new value in the desired order and then running the analysis with a FORMAT statement to provide the original values. In this example, the order is changed to AML–Low Risk (the top function), followed by ALL (the middle function), followed by AML–High Risk. With this ordering, there is a clearer correspondence between the functions, the at-risk table, and the legend. The following steps illustrate this reordering:

```
proc format;
   invalue bmtnum    'AML-Low Risk' = 1   'ALL' = 2   'AML-High Risk' = 3;
   value    bmtfmt   1 = 'AML-Low Risk'   2 = 'ALL'   3 = 'AML-High Risk';
run;

data BMT(drop=g);
   set sashelp.BMT(rename=(group=g));
   Group = input(g, bmtnum.);
run;

proc lifetest data=BMT plots=survival(cl test atrisk(maxlen=13));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

The PROC FORMAT step has two statements. The INVALUE statement creates an informat that maps the values of the original Group variable into integers that have the correct order. The VALUE statement creates a format that maps the integers back to the original values. The informat is used with the INPUT function in the DATA step to create a new integer Group variable. The FORMAT statement assigns the BMTFMT format to the Group variable so that the actual risk groups are displayed in the analysis. You specify the ORDER=INTERNAL option in the STRATA statement to sort the Group values based on internal order (the order specified by the integers, which are the internal unformatted values). This example also illustrates the CL option, which displays pointwise confidence limits for the survival curve (instead of the Hall-Wellner confidence bands). The results are displayed in .

Figure 23.13 Controlling Legend Order
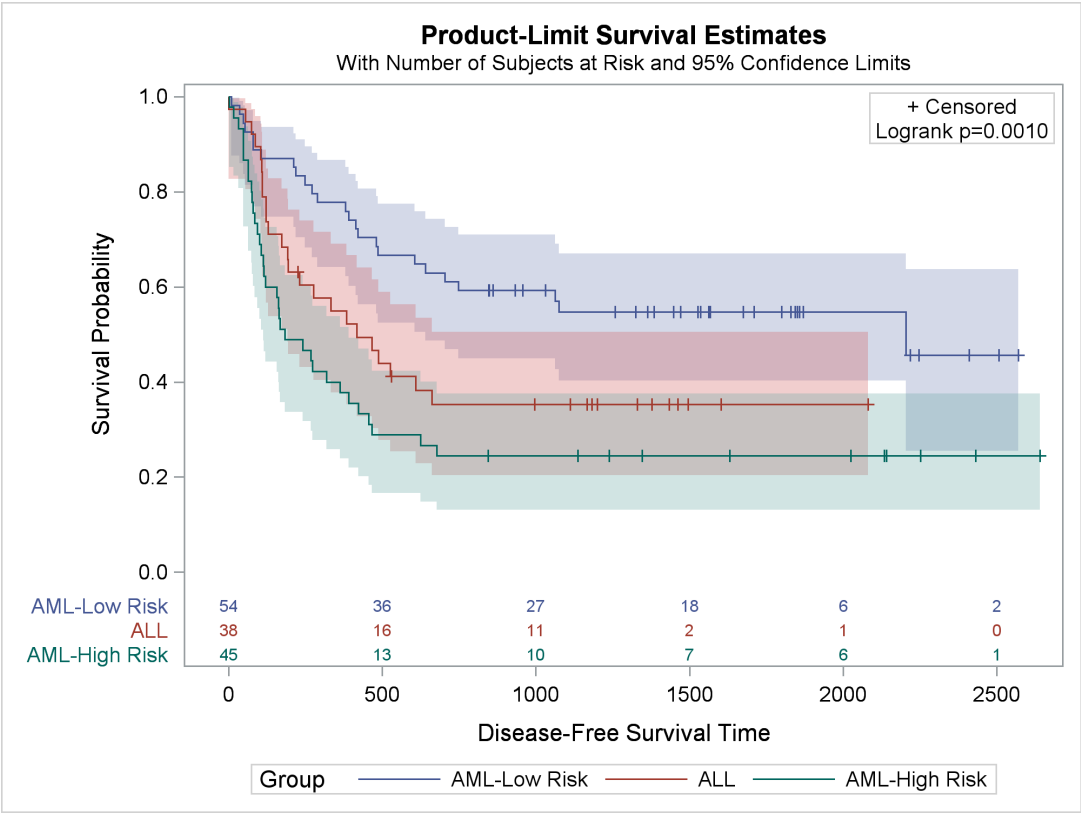
You can submit the following steps to display ALL first, followed by AML–Low Risk and then AML–High Risk:

```
proc format;
   invalue bmtnum 'ALL' = 1  'AML-Low Risk' = 2  'AML-High Risk' = 3;
   value   bmtfmt 1 = 'ALL'  2 = 'AML-Low Risk'  3 = 'AML-High Risk';
run;

data BMT(drop=g);
   set sashelp.BMT(rename=(group=g));
   Group = input(g, bmtnum.);
run;

proc lifetest data=BMT plots=survival(cl test atrisk(maxlen=13));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

The results are displayed in Figure 23.14.

Figure 23.14 Controlling Legend Order
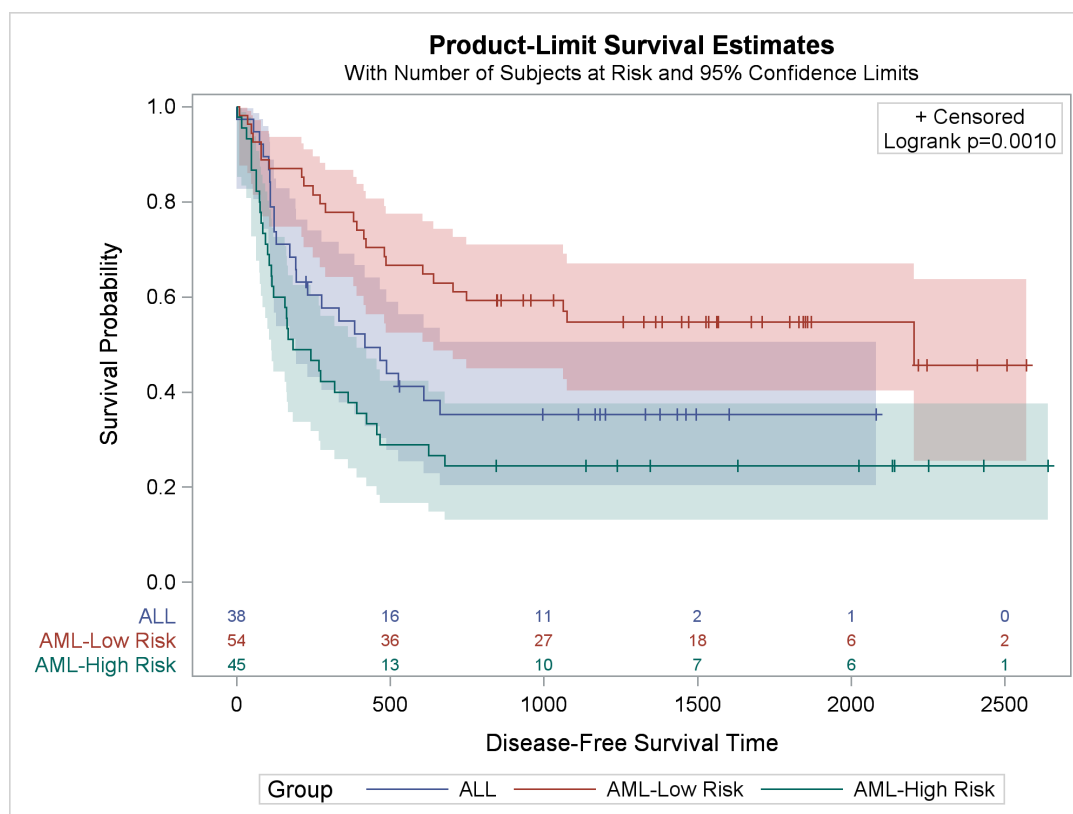
Suppressing the Censored Observations

You can use the PLOTS=SURVIVAL(NOCENSOR) option to suppress the display of censored observations as follows:

```
proc lifetest data=sashelp.BMT
               plots=survival(nocensor test atrisk(maxlen=13));
    time T * Status(0);
    strata Group;
run;
```

The results are displayed in Figure 23.15.

Figure 23.15 Censored Values Not Displayed
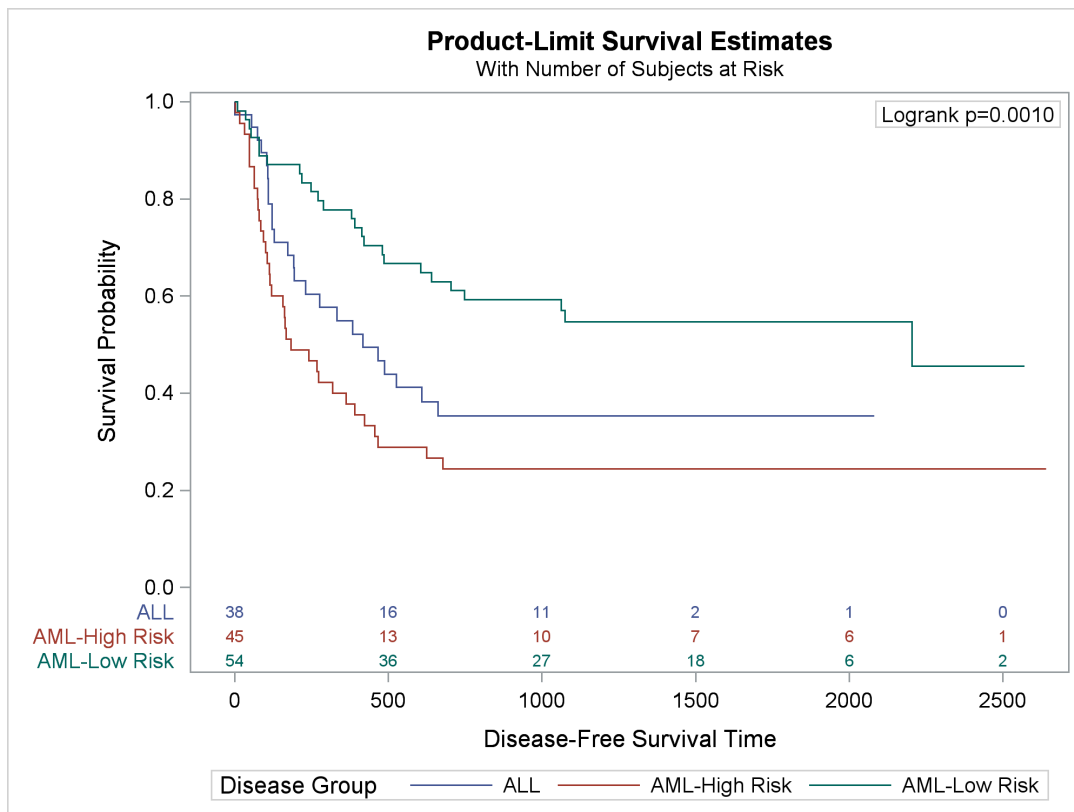
Failure Plots

All the discussion up to this point has been about survival plots. You can instead plot failure probabilities by using the PLOTS=SURVIVAL(FAILURE) option as follows:

```
proc lifetest data=sashelp.BMT
      plots=survival(cb=hw failure test atrisk(maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.16.

Figure 23.16 Failure Plot
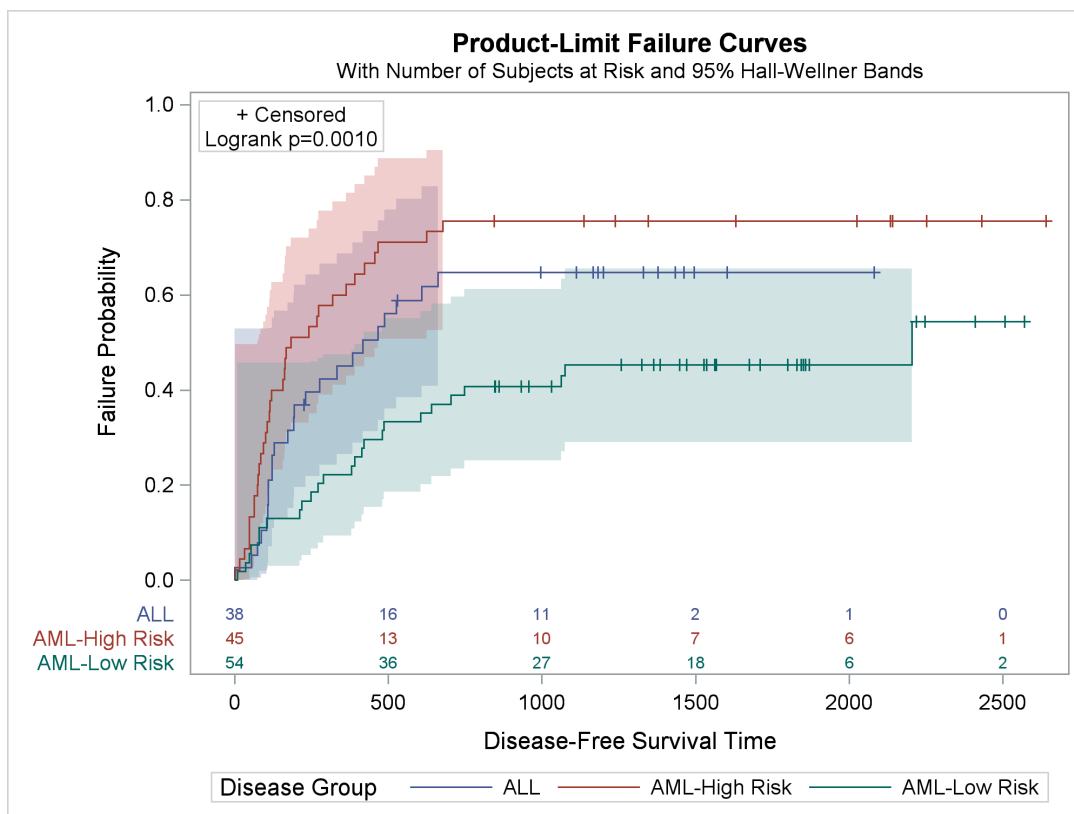
Controlling the Survival Plot by Modifying Graph Templates

The preceding section illustrates the PLOTS= options for controlling the survival plot. If you need to make modifications that are not shown in that section, this section shows how to modify the survival plot by using macros and macro variables to modify graph templates.

The Modularized Templates

SAS provides the two templates that are used to make the survival plot in a modularized form.[2] The modularized version of the two survival plot templates is available in the SAS sample library and is on the Web at http://support.sas.com/documentation/onlinedoc/stat/ex_code/142/templft.html.

The file defines the macro %ProvideSurvivalMacros, which defines a series of macros and macro variables. The %ProvideSurvivalMacros macro contains a %GLOBAL statement, a series of %LET statements, and several macro definitions. It ends with a call to the %CompileSurvivalTemplates macro (which is defined inside the %ProvideSurvivalMacros macro), which compiles the two survival plot templates.[3] By using these macros and macro variables, you can easily specify single changes that modify both templates. All the statements in this file are displayed and explained in more detail in the section "Graph Templates, Macros, and Macro Variables" on page 851.

The %ProvideSurvivalMacros macro provides a way to provide (and in subsequent steps restore) the default macros and macro variables. The macros and macro variables are designed so that you can make most changes by submitting just a few lines of SAS code. Hence, you should not modify any of the statements while they are inside the %ProvideSurvivalMacros macro. Rather, you should use this macro only to provide all the default macros and macro variables. You should modify the individual macros and macro variables outside the context of the %ProvideSurvivalMacros macro.[4] The reasons for this will become clearer as you work through the examples. Before you modify anything, you must submit the %ProvideSurvivalMacros macro definition from the sample library to SAS. You can both store the macros in a temporary file and submit them to SAS by submitting the following statements:

```
data _null_;
   %let url = //support.sas.com/documentation/onlinedoc/stat/ex_code/142;
   infile "http:&url/templft.html" device=url;

   file 'macros.tmp';
   retain pre 0;
   input;
   _infile_ = tranwrd(_infile_, '&amp;', '&');
   _infile_ = tranwrd(_infile_, '&lt;' , '<');
   if index(_infile_, '</pre>') then pre = 0;
   if pre then put _infile_;
   if index(_infile_, '<pre>')  then pre = 1;
run;

%inc 'macros.tmp' / nosource;
```

The TRANWRD function calls change the HTML coding of the ampersand (`'&'`) and the less than sign `'<'`) to the actual characters (`'&'` and `'<'`).

---

[2]The two templates that PROC LIFETEST uses are named **Stat.Lifetest.Graphics.ProductLimitSurvival** and **Stat.Lifetest.Graphics.ProductLimitSurvival2**.

[3]You might wonder why these macros are not simply made available in the SAS autocall library. The autocall library provides macros that you can run. In this context, you do not need to simply run a macro. You need to copy it, extract parts of it, modify those parts, and submit the modified statements. That is not convenient with the autocall library.

[4]However, there might be something that you *always* want to change. For example, if you always want the survival plot to be entitled 'Kaplan-Meier Plot', then you can modify the title once inside the %ProvideSurvivalMacros macro. This is not illustrated in this chapter. All examples illustrate ad hoc changes that are made outside the context of the %ProvideSurvivalMacros macro.

Submitting these statements only defines the %ProvideSurvivalMacros macro. It does not make any of its component macros and macro variables available. The URL macro variable is used to avoid an overly long INFILE statement.

You can provide the default macros and macro variables by running the following macro:

%ProvideSurvivalMacros

Running this macro provides the default macros and macro variables (or restores them if you have previously submitted the %ProvideSurvivalMacros macro).[5] The %ProvideSurvivalMacrosmacro also runs the %CompileSurvivalTemplates macro and hence replaces any compiled survival plot templates that you might have created in the past. You can recompile the templates by submitting the following macro:

%CompileSurvivalTemplates

This macro runs PROC TEMPLATE and compiles the templates from all the macros and macro variables in the %ProvideSurvivalMacros macro along with any that you modified. Running this macro produces two compiled templates that are stored in a special SAS data file called an item store. For more information about SAS item stores, see the section "SAS Item Stores" on page 879. Assuming that you have not modified your ODS path by using an ODS PATH statement, compiled templates are stored in an item store in the Sasuser library. Files in the Sasuser library persist across SAS sessions until they are deleted. When you are done with a modified template, it is wise to clean up all remnants of it by restoring the default macros and by deleting the modified templates from the Sasuser template item store. You can delete the modified templates (so that SAS can only find the original templates) by running the following step:

```
proc template;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival  /
          store=sasuser.templat;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival2 /
          store=sasuser.templat;
run;
```

This step deletes the compiled templates from the item store sasuser.templat. You can omit the STORE= option if you are using the default ODS path, but it is good practice to explicitly control which templates are deleted. Deleting the compiled templates does not change any of the macros or macro variables. Only the compiled templates (not the macros or macro variables) affect the graph when you run PROC LIFETEST. For more information about compiled templates, item stores, and cleanup, see the section "SAS Item Stores" on page 879.

---

[5]Semicolons are not needed after a macro call like this one, so they are not used in these examples.

Changing the Plot Title

Here is a simple, complete program (except for retrieving the %ProvideSurvivalMacros macro from the sample library) with setup, macro variable modifications to change the title, and cleanup:

```
/*-- Original Macro Variable Definitions ---------------------------------
%let TitleText0 = METHOD " Survival Estimate";
%let TitleText1 = &titletext0 " for " STRATUMID;
%let TitleText2 = &titletext0 "s";
------------------------------------------------------------------------*/

                                        /* Make the macros and macro     */
%ProvideSurvivalMacros                  /* variables available.          */

%let TitleText0 = "Kaplan-Meier Plot";  /* Change the title.             */
%let TitleText1 = &titletext0 " for " STRATUMID;
%let TitleText2 = &titletext0;

%CompileSurvivalTemplates               /* Compile the templates with    */
                                        /* the new title.                */

proc lifetest data=sashelp.BMT          /* Perform the analysis and make */
              plots=survival(cb=hw test); /* the graph.                  */
   time T * Status(0);
   strata Group;
run;

%ProvideSurvivalMacros                  /* Optionally restore the default */
                                        /* macros and macro variables.    */

proc template;                          /* Delete the modified templates. */
   delete Stat.Lifetest.Graphics.ProductLimitSurvival  / store=sasuser.templat;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival2 / store=sasuser.templat;
run;
```

The results are displayed in Figure 23.17. You can see that the graph title is now 'Kaplan-Meier Plot'.

There are multiple title macro variables because two different types of plots are defined in the survival plot templates. The first macro variable, TitleText0, contains the text that is the same for both types of plots. The second macro variable, TitleText1, contains the title for the single-stratum case. The third macro variable, TitleText2, contains the title for the multiple-strata case. Both TitleText1 and TitleText2 use the common text defined in TitleText0. Both TitleText0 and TitleText2 were changed from their original definition; the definition of TitleText1 was copied from the %ProvideSurvivalMacros macro. You must provide all relevant %LET statements when you modify TitleText0. In this case it is TitleText0 and TitleText2, but it is easy to copy all three and then just modify what you need. Alternatively, when you know the number of strata, you can modify only TitleText1 or TitleText2.

Figure 23.17 Kaplan-Meier Plot Title Modification

Modifying the Axis

The following statements modify the default tick value list for the Y axis from the default increment of 0.2 to have an increment of 0.25 and also change the Y-axis label to 'Survival':

```
/*-- Original Macro Variable Definitions ----------------------------------
%let yOptions   = label="Survival Probability" shortlabel="Survival"
                  linearopts=(viewmin=0 viewmax=1
                              tickvaluelist=(0 .2 .4 .6 .8 1.0));
----------------------------------------------------------------------*/


%ProvideSurvivalMacros

%let yOptions = label="Survival"
                linearopts=(viewmin=0 viewmax=1
                            tickvaluelist=(0 .25 .5 .75 1));

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.18.

Figure 23.18 Y-Axis Modification

The following statements modify the Y axis so that tick marks start at 0.2:

```
%ProvideSurvivalMacros

%let yOptions = label="Survival"
                linearopts=(viewmin=0.2 viewmax=1
                            tickvaluelist=(0 .2 .4 .6 .8 1.0));

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```

You only need to change the value of the VIEWMIN= option, in this case from 0 to 0.2. You do not need to modify the tick value list. The VIEWMIN= option (not the tick value list) controls the smallest value shown on the axis. The results are displayed in Figure 23.19.

Figure 23.19 Y Axis, First Tick Change

Changing the Line Thickness

The following steps modify the line thickness for the step functions in the survival plot:

```
%ProvideSurvivalMacros

%let StepOpts = lineattrs=(thickness=2.5);

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.20.

Figure 23.20 Changing Line Thickness

By default, the StepOpts macro variable is null.

Changing the Group Color

SAS styles control the colors displayed in graphs. The style elements `GraphData1`, `GraphData2`, ..., `GraphData12` control the appearance of groups of observations such as the survival step plots in the Kaplan-Meier plot. You can override these colors by using the GraphOpts macro variable (which is null by default). By default, the colors for the first three groups in the HMTLBlue style are shades of blue, red, and green. You can change them to a pure green, red, and blue as follows:

```
%ProvideSurvivalMacros

%let GraphOpts = DataContrastColors=(green red blue)
                 DataColors=(green red blue);

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT
              plots=survival(cb=hw test atrisk(outside maxlen=13));
    time T * Status(0);
    strata Group;
run;
```

The results are displayed in Figure 23.21. The DATACONTRASTCOLORS= option specifies the contrast colors, which are used for markers and lines. The DATACOLORS= option specifies the colors, which are used for shaded areas such as confidence bands.

Figure 23.21 Named Color Specifications

The original colors (as shown in Figure 23.33) are more subtle than those shown in Figure 23.21. If you want to change the order of the original colors by using this approach, then you need to know what they are so that you can specify them. The graph colors for the HTMLBlue and Statistical styles are extracted from the style in the section "Displaying a Style and Extracting Color Lists" on page 868 and displayed in Figure 23.36. The section "Modifying Color Lists" on page 871 shows you how to change the graph template to specify the original colors in a different order. The section "Swapping Colors among Style Elements" on page 872 shows you how to use a macro to change a style template to specify the original colors in a different order (without having to extract and specify the color names).

Changing the Line Pattern

You can change the line patterns as follows:

```
%ProvideSurvivalMacros

%let GraphOpts = attrpriority=none
                 DataLinePatterns=(ShortDash MediumDash LongDash);

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT
              plots=survival(cb=hw test atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.22.

Figure 23.22 Changing the Line Patterns

ALL	38	16	11	2	1	0
AML-High Risk	45	13	10	7	6	1
AML-Low Risk	54	36	27	18	6	2

Other values for the DATALINEPATTERNS= option are provided in the section "The Macro Variables" on page 853. You must use the option ATTRPRIORITY=NONE when you want to have varying line patterns in an ATTRPRIORITY=COLOR style like HTMLBlue or Pearl. In an ATTRPRIORITY=COLOR style, groups are not distinguished by line patterns, and the line patterns for second and subsequent groups match the line pattern for the first group.

Changing the Font

You can change the Y-axis, X-axis, and title fonts as follows:

```
%ProvideSurvivalMacros

/*-- Original Macro Variable Definitions ---------------------------------
%let TitleText0 = METHOD " Survival Estimate";
%let TitleText1 = &titletext0 " for " STRATUMID;
%let TitleText2 = &titletext0 "s";
%let yOptions   = label="Survival Probability"
                  shortlabel="Survival"
                  linearopts=(viewmin=0 viewmax=1
                              tickvaluelist=(0 .2 .4 .6 .8 1.0));
%let xOptions   = shortlabel=XNAME
                  offsetmin=.05
                  linearopts=(viewmax=MAXTIME tickvaluelist=XTICKVALS
                              tickvaluefitpolicy=XTICKVALFITPOL);
-------------------------------------------------------------------------*/

%let tatters   = textattrs=(size=12pt weight=bold family='arial');
%let TitleText0 = METHOD " Survival Estimate";
%let TitleText1 = &titletext0 " for " STRATUMID / &tatters;
%let TitleText2 = &titletext0 "s" / &tatters;

%let yOptions   = label="Survival Probability"
                  shortlabel="Survival"
                  labelattrs=(size=10pt weight=bold)
                  tickvalueattrs=(size=8pt)
                  linearopts=(viewmin=0 viewmax=1
                              tickvaluelist=(0 .2 .4 .6 .8 1.0));

%let xOptions   = shortlabel=XNAME
                  offsetmin=.05
                  labelattrs=(size=10pt weight=bold)
                  tickvalueattrs=(size=8pt)
                  linearopts=(viewmax=MAXTIME tickvaluelist=XTICKVALS
                              tickvaluefitpolicy=XTICKVALFITPOL);

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```
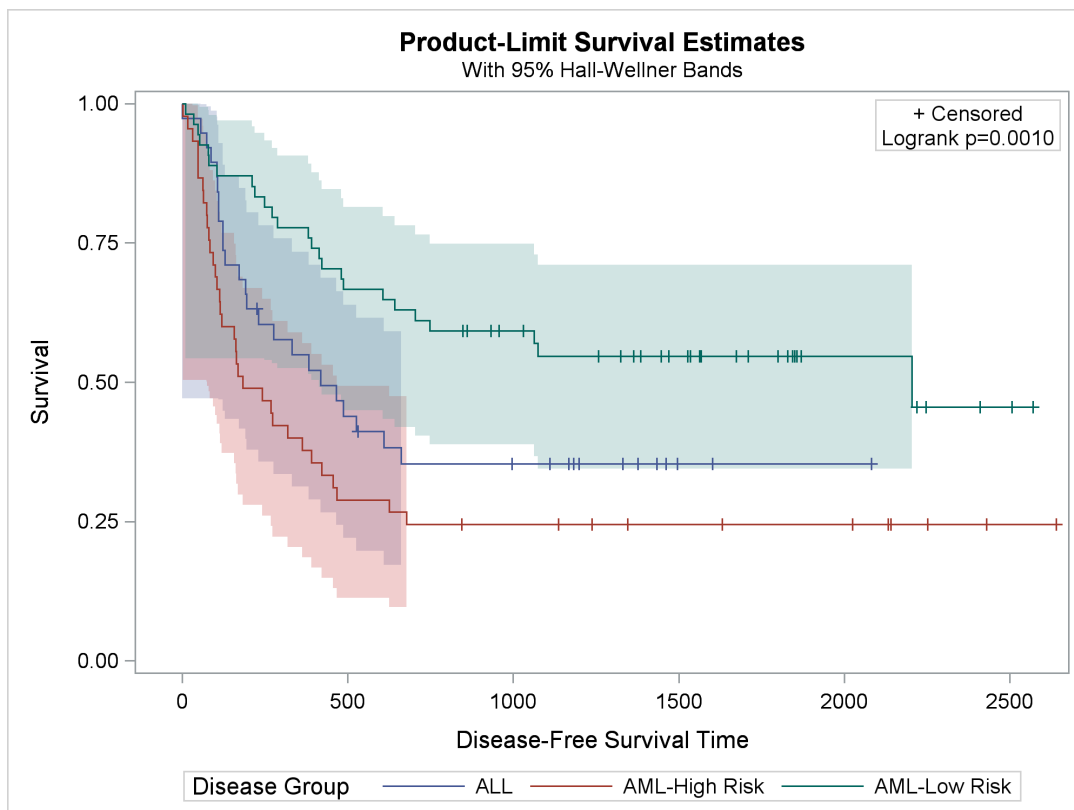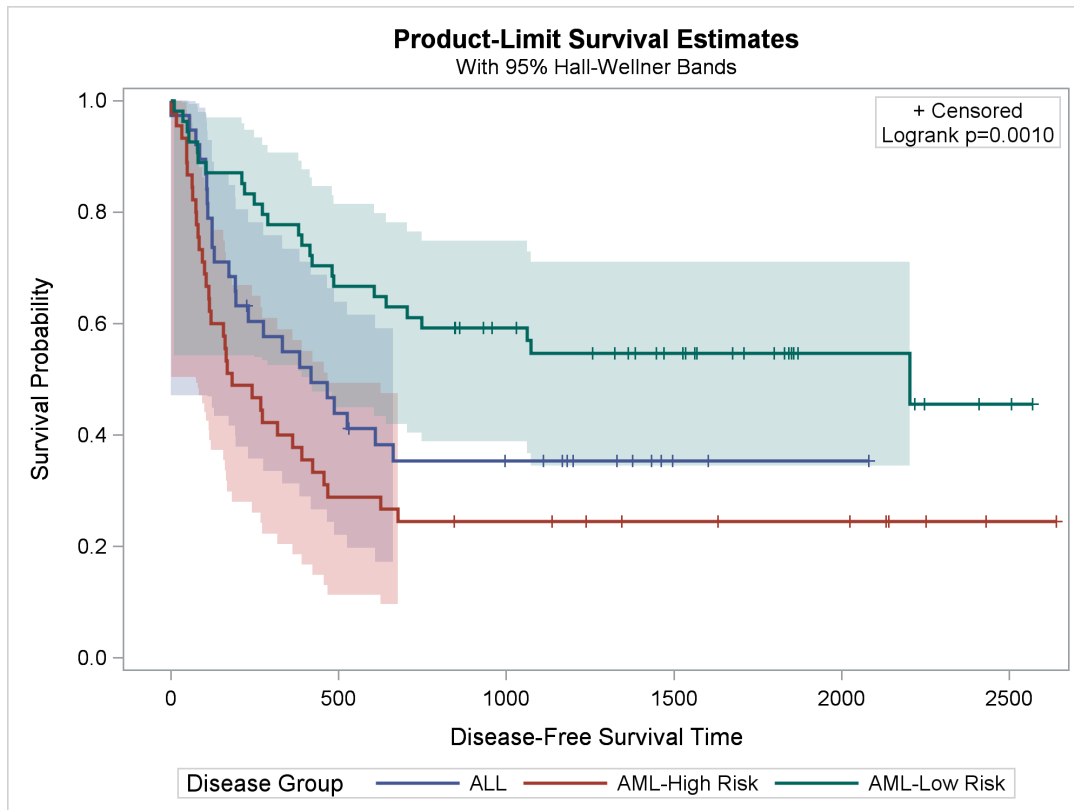
The results are displayed in Figure 23.23.

Figure 23.23 Changing the Fonts

Font options include the following:

COLOR=*style-reference* | *color*
FAMILY=*style-reference* | *'string'*
SIZE=*style-reference* | *dimension*
STYLE=*style-reference* | **NORMAL** | **ITALIC**
WEIGHT=*style-reference* | **NORMAL** | **BOLD**

Fonts vary from installation to installation. Sample font strings include: 'Times New Roman', 'Courier New', 'Arial', and 'Calibri'. For more information about text and label attribute options, see *SAS Graph Template Language: Reference*. For information about changing fonts in ODS styles, see the section "Displaying a Style and Extracting Font Information" on page 874. ODS Graphics can use a single style element in more than one place in a graph; this example shows how to change individual graph components.

Changing the Legend and Inset Position

This example shows you how to move the legend inside the plot (to the top right) and move the homogeneity test and censored value legend to the bottom right of the plot:

```
%ProvideSurvivalMacros

/*-- Original Macro Variable Definitions ---------------------------------
%let InsetOpts  = autoalign=(TOPRIGHT BOTTOMLEFT TOP BOTTOM)
                  border=true BackgroundColor=GraphWalls:Color Opaque=true;
%let LegendOpts = title=GROUPNAME location=outside;
----------------------------------------------------------------------*/

%let InsetOpts  = autoalign=(BottomRight)
                  border=true BackgroundColor=GraphWalls:Color Opaque=true;
%let LegendOpts = title=GROUPNAME location=inside across=1 autoalign=(TopRight);

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT
              plots=survival(cb=hw test atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

This example shows you how to replace the AUTOALIGN=(TOPRIGHT BOTTOMLEFT TOP BOT-TOM) option in the macro variable InsetOpts with AUTOALIGN=(BOTTOMRIGHT) and add the AU-TOALIGN=(TOPRIGHT) option to the LegendOpts macro variable. You can also add the option ACROSS=1 to the LegendOpts macro variable to stack all legend entries vertically (with just one element in each row).

The results are displayed in Figure 23.24.

Figure 23.24 Controlling Legend Placement

Changing How the Censored Points Are Displayed

By default, PROC LIFETEST displays a plus sign to indicate censoring. This example illustrates how to change the plus sign to a small filled circle in both the step plots and the inset box. The following steps change the template and create Output 23.25:

```
/*-- Original Macro Variable Definitions --------------------------------
%let Censored   = markerattrs=(symbol=plus);
%let CensorStr  = "+ Censored";
----------------------------------------------------------------------*/

%ProvideSurvivalMacros

%let censored  = markerattrs=(symbol=circlefilled size=3px);
%let censorstr = "(*ESC*){Unicode '25cf'x} Censored"
                 / textattrs=GraphValueText(family=GraphUnicodeText:FontFamily);

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT plots=survival(cb=hw atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

Figure 23.25 Survival Plot with a Modified Display of Censoring

The Unicode Consortium (http://unicode.org/) provides a list of character codes. Also see Figure 22.2.7 in Chapter 22, "ODS Graphics Template Modification," for information about the Unicode specification for other markers. Although some Unicode characters are supported in some fonts, you should always specify a Unicode font when using special characters.

Adding a Y-Axis Reference Line

You can add a horizontal reference line to the survival plot by adding the following statement to the template:

```
referenceline y=0.5;
```

You can do this by using the %StmtsTop macro. By default, this macro is empty. You can use the %StmtsTop macro to add new statements to the beginning of the block of statements that define the appearance of the graph. In contrast, you can use the %StmtsBottom macro to provide statements at the end of the statement block. ODS Graphics draws statements in the order in which they appear; therefore, reference lines should be drawn first so they do not obscure other parts of the graph.

The following step creates the plot in Figure 23.26:

```
%ProvideSurvivalMacros

%macro StmtsTop;
   referenceline y=0.5;
%mend;

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```

Figure 23.26 Horizontal Reference Line

Changing the Homogeneity Test Inset

This example modifies the contents of the %pValue macro. The original %pValue macro definition is as follows:

```
%macro pValue;
   if (PVALUE < .0001)
      entry TESTNAME " p " eval (PUT(PVALUE, PVALUE6.4));
   else
      entry TESTNAME " p=" eval (PUT(PVALUE, PVALUE6.4));
   endif;
%mend;
```

The following example directly specifies the test name (replacing the internal name 'Logrank' with 'Log Rank') and adds blank spaces around the equal sign:

```
%ProvideSurvivalMacros

%macro pValue;
   if (PVALUE < .0001)
      entry "Log Rank p "   eval (PUT(PVALUE, PVALUE6.4));
   else
      entry "Log Rank p = " eval (PUT(PVALUE, PVALUE6.4));
   endif;
%mend;

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.27.

Figure 23.27 Cosmetic Inset Entry Change

Because this template modification replaces a character string that is more appropriately set by PROC LIFETEST, you should clean up afterward as follows:

```
%ProvideSurvivalMacros

proc template;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival  /
          store=sasuser.templat;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival2 /
          store=sasuser.templat;
run;
```

Suppressing the Second Title and Adding a Footnote

The following steps add an ENTRYFOOTNOTE statement to the %StmtsBeginGraph macro and suppress the second title:

```
%ProvideSurvivalMacros

%let ntitles = 1;
%macro StmtsBeginGraph;
   entryfootnote halign=left "Acme Company %sysfunc(date(),worddate.)" /
               textattrs=GraphDataText;
%mend;

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT
              plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```
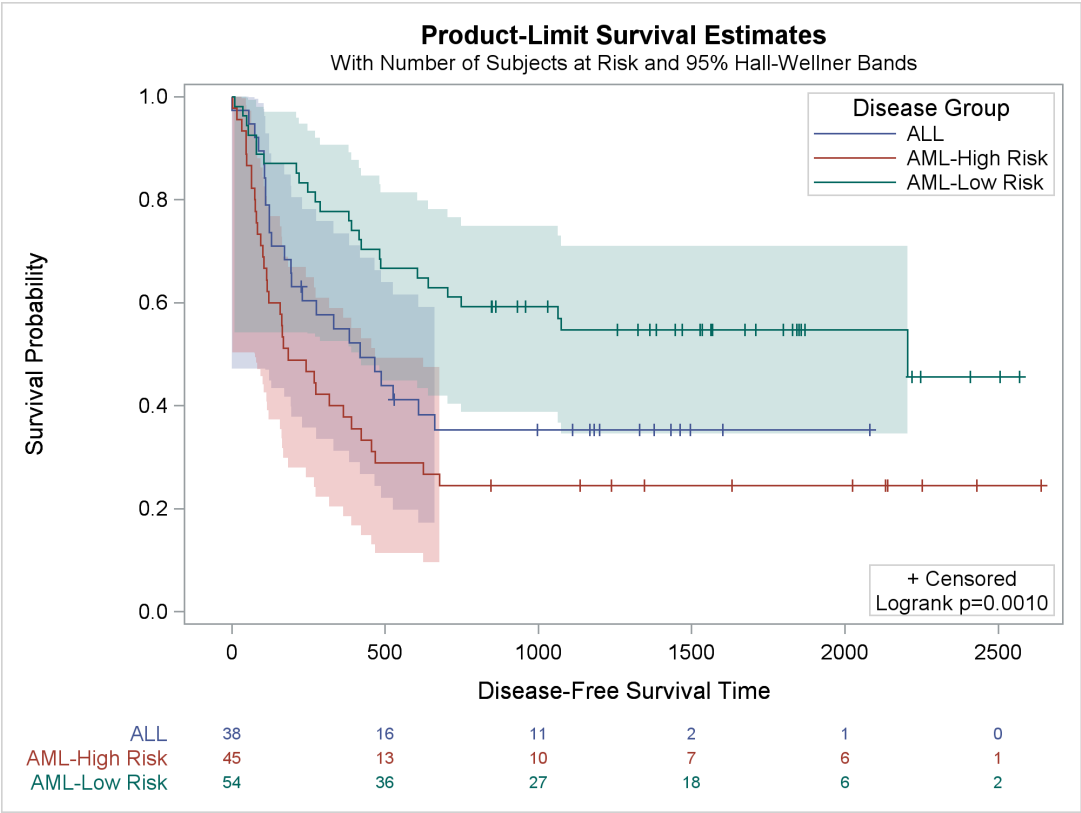
The results are displayed in Figure 23.28.

Figure 23.28 Footnote but No Second Title

By default, the nTitles macro variable is 2, and all titles are displayed. Setting nTitles to 1 suppresses the

second title. You can add titles or footnotes to the plot by adding them to the %StmtsBeginGraph macro. This example adds a footnote that consists of a company name followed by the current date, formatted by using the WORDDATE format. The **GraphDataText** style element is used; it has a smaller font than the default style element, **GraphFootnoteText**.

Adding a Small Inset Table with Event Information

This example shows you how to modify the template to produce the plot displayed in Output 23.29. This new plot has an inset table in the top right corner that shows the number of observations and the number of events in each stratum. The legend has been moved inside the plot and combined with the old inset table that shows the marker for censored observations.[6] Also, the title is changed to 'Kaplan-Meier Plot'.

```
%ProvideSurvivalMacros

%let TitleText2 = "Kaplan-Meier Plot";
%let LegendOpts = title="+ Censored"
                  location=inside autoalign=(Bottom);
%let InsetOpts  = ;

%macro StmtsBottom;
   dynamic %do i = 1 %to 3; StrVal&i NObs&i NEvent&i %end;;
   layout gridded / columns=3 border=TRUE autoalign=(TopRight);
      entry ""; entry "Event"; entry "Total";
      %do i = 1 %to 3;
         %let t = / textattrs=GraphData&i;
         entry halign=right Strval&i &t; entry NEvent&i &t; entry NObs&i &t;
      %end;
   endlayout;
%mend;

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT plots=survival(cb=hw atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.29.

---

[6]This legend is wide and might not be displayed if your graph is small. If the legend is not displayed, try increasing the size of the graph by specifying the WIDTH= or HEIGHT= option in the ODS GRAPHICS statement.

Figure 23.29 New Inset Table with Event Information

The macro variable TitleText2, which controls the title for the multiple-strata plot, is changed. You can change all three title macro variables, as is done in the construction of Figure 23.17, or you can change only TitleText2 when you have multiple overlaid strata, as in this example. The LegendOpts macro variable value was changed from TITLE=GROUPNAME LOCATION=OUTSIDE to display the censored value legend in place of the legend title and to display the legend inside the bottom of the plot. When the InsetOpts macro variable is null, the usual inset that contains the censored value and *p*-value is not displayed.

The %StmtsBottom macro (null by default) is replaced with a macro that creates the new inset table. This macro adds statements to the bottom of the templates. If you ignore for a moment most of the options, the core of the generated statements is as follows:

```
dynamic StrVal1 NObs1 NEvent1 StrVal2 NObs2 NEvent2 StrVal3 NObs3 NEvent3;
layout gridded / columns=3;
   entry "";          entry "Event";   entry "Total";
   entry Strval1;     entry NEvent1;   entry NObs1;
   entry Strval2;     entry NEvent2;   entry NObs2;
   entry Strval3;     entry NEvent3;   entry NObs3;
endlayout;
```

The macro first constructs a DYNAMIC statement that includes the names of the dynamic variables that contain some of the results. PROC LIFETEST creates these dynamic variables and sets them to values, but you must declare them in your template before using them. For more information about these dynamic variables, see the section "Additional Dynamic Variables" on page 864. The macro then constructs a 4 × 3 grid that contains a table consisting of a title line and a row for each stratum (which consists of the stratum

label, the number of events, and the total number of subjects). The full layout that the %StmtsBottom macro generates, with all the options, is as follows:

```
dynamic StrVal1 NObs1 NEvent1 StrVal2 NObs2 NEvent2 StrVal3 NObs3 NEvent3;
layout gridded / columns=3 border=TRUE autoalign=(TopRight);
   entry "";
   entry "Event";
   entry "Total";
   entry halign=right Strval1 / textattrs=GraphData1;
   entry NEvent1 / textattrs=GraphData1;
   entry NObs1 / textattrs=GraphData1;
   entry halign=right Strval2 / textattrs=GraphData2;
   entry NEvent2 / textattrs=GraphData2;
   entry NObs2 / textattrs=GraphData2;
   entry halign=right Strval3 / textattrs=GraphData3;
   entry NEvent3 / textattrs=GraphData3;
   entry NObs3 / textattrs=GraphData3;
endlayout;
```

Adding an External Table with Event Information

This example adds a table to the plot that displays a summary of event information. The following statements create Figure 23.30:

```
%ProvideSurvivalMacros

%let GraphOpts = DesignHeight=DefaultDesignWidth;

%SurvivalSummaryTable

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT
              plots=survival(cb=hw atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

Figure 23.30 External Event Table

The GraphOpts macro variable specifies the option DESIGNHEIGHT=DEFAULTDESIGNWIDTH. At the default graph size (the size at which the graph is designed), this option sets the graph height to the default graph width of 640 pixels. The macro %SurvivalSummaryTable adds new statements to the graph templates that display the number of subjects, number of events, number of censored observations, median survival time, and 95% confidence limits for the median survival time. For more information about the %SurvivalSummaryTable macro, see the section "Event Table Macros" on page 861.

Suppressing the Legend

The plot in Figure 23.30 has a legend. However, the plot displays values in the tables by using colors that match the colors of the step functions, so you do not need the legend. The next statements show how to remove the legend:

```
%ProvideSurvivalMacros

%let GraphOpts  = DesignHeight=500px;
%let LegendOpts = ;

%SurvivalSummaryTable

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT
              plots=survival(cb=hw atrisk(maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

Figure 23.31 Plot with Legend Removed

The legend is suppressed when the LegendOpts macro variable is null. This example also illustrates changing the design height to 500 pixels and moving the at-risk table back inside the body of the plot.

Kaplan-Meier Plot with Event Table and Other Customizations

This example combines a number of features from previous examples. The order of the strata levels in the tables is ALL, AML–Low Risk, and AML–High Risk (see the section "Reordering the Groups" on page 820). The title is set to 'Kaplan-Meier Plot' (see the section "Changing the Plot Title" on page 827). The second title line is suppressed (see the section "Suppressing the Second Title and Adding a Footnote" on page 843). The graph height is set to 500 pixels (see the section "Suppressing the Legend" on page 848). The legend and the inset box that contains the legend for censored values are both suppressed (see the sections "Suppressing the Legend" on page 848 and "Adding a Small Inset Table with Event Information" on page 844). The event table is displayed outside the plot (see the section "Adding an External Table with Event Information" on page 846) and the at-risk table is displayed inside the plot (see the section "Displaying the Patients-at-Risk Table inside the Plot" on page 814).

```
proc format;
    invalue bmtnum 'ALL' = 1  'AML-Low Risk' = 2  'AML-High Risk' = 3;
    value   bmtfmt 1 = 'ALL'  2 = 'AML-Low Risk'  3 = 'AML-High Risk';
run;

data BMT(drop=g);
    set sashelp.BMT(rename=(group=g));
    Group = input(g, bmtnum.);
run;

%ProvideSurvivalMacros

%let TitleText2 = "Kaplan-Meier Plot";
%let nTitles    = 1;
%let GraphOpts  = DesignHeight=500px;
%let LegendOpts = ;
%let InsetOpts  = ;

%SurvivalSummaryTable

%CompileSurvivalTemplates

proc lifetest data=BMT plots=survival(cb=hw atrisk(maxlen=13));
    time T * Status(0);
    strata Group / order=internal;
    format group bmtfmt.;
run;
```

The results are displayed in Figure 23.32.

Figure 23.32 Kaplan-Meier Plot with Extensive Customizations

Compiled Template Cleanup

The following step restores all the default macros and macro variables and deletes the modified templates:

```
%ProvideSurvivalMacros

proc template;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival  /
          store=sasuser.templat;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival2 /
          store=sasuser.templat;
run;
```

For more information about deleting compiled templates, see the section "SAS Item Stores" on page 879.

Graph Templates, Macros, and Macro Variables

The %ProvideSurvivalMacros macro and the macros and macro variables that it provides have the following properties:

- Many options, including most of the options that are specified in multiple places in the templates, are extracted to macro variables.

- The %CompileSurvivalTemplates macro provides the main body of the two templates. You can call it to compile the templates after making changes.

  - The template **Stat.Lifetest.Graphics.ProductLimitSurvival** provides the survival template when the at-risk table is inside the body of the plot.
  - The template **Stat.Lifetest.Graphics.ProductLimitSurvival2** provides the survival template when the at-risk table is outside the body of the plot.[7]

  The two templates share many statements, and a macro %DO loop creates both versions.

- The portion of the templates for the table for the *p*-values is stored in the macro %pValue.

- The portion of the templates for the single-stratum case is stored in the macro %SingleStratum.

- The portion of the templates for the multiple-strata case is stored in the macro %MultipleStrata.

- The macro %AtRiskLatticeStart begins the two-cell lattice that contains the plot above the table when the at-risk table is outside the body of the plot.

- The macro %AtRiskLatticeEnd ends the two-cell lattice that contains the plot and the table when the at-risk table is outside the body of the plot.

- Some empty macros (%StmtsBeginGraph, %StmtsTop, and %StmtsBottom) are provided to enable you to add statements and options to strategic places in the templates.

- The %SurvTabHeader, %SurvivalTable, and %SurvivalSummaryTable macros enable you to easily add more GTL statements to the Kaplan-Meier plot templates to display event information for each stratum.

---

[7]The macros do not affect any graph that uses graph templates other than the two templates that are modified here. The macros do not affect the STRATA=PANEL plot that uses the template **Stat.Lifetest.Graphics.ProductLimitSurvivalPanel** or the failure plot that uses the template **Stat.Lifetest.Graphics.ProductLimitFailure**.

This organization makes it easy to identify the relevant parts of the templates, modify these parts, and recompile the templates. A small portion of the %ProvideSurvivalMacros macro follows:

```
%macro ProvideSurvivalMacros;

   %global atriskopts bandopts censored censorstr classopts
           graphopts groups insetopts legendopts ntitles stepopts tiplabel
           tips titletext0 titletext1 titletext2 xoptions yoptions;

   %let TitleText0 = METHOD " Survival Estimate";
   %let TitleText1 = &titletext0 " for " STRATUMID;
   %let TitleText2 = &titletext0 "s";            /* plural: Survival Estimates */

   %let yOptions    = label="Survival Probability" shortlabel="Survival"
                      linearopts=(viewmin=0 viewmax=1
                                 tickvaluelist=(0 .2 .4 .6 .8 1.0));

   %let xOptions    = shortlabel=XNAME offsetmin=.05
                      linearopts=(viewmax=MAXTIME tickvaluelist=XTICKVALS
                                 tickvaluefitpolicy=XTICKVALFITPOL);
   . . .

   %macro CompileSurvivalTemplates; . . . %mend;
   %macro pValue;                   . . . %mend;
   %macro SingleStratum;            . . . %mend;
   %macro MultipleStrata;           . . . %mend;
   . . .
%CompileSurvivalTemplates
%mend;
```

The Macro Variables

The macros and macro variables are designed so that most of the time you need to modify only the macro variables and not the larger macros. However, you have the full flexibility to modify both. You can modify any of the following macro variables:

```
%let TitleText0 = METHOD " Survival Estimate";
%let TitleText1 = &titletext0 " for " STRATUMID;
%let TitleText2 = &titletext0 "s";          /* plural: Survival Estimates */
%let nTitles    = 2;

%let yOptions   = label="Survival Probability" shortlabel="Survival"
                  linearopts=(viewmin=0 viewmax=1
                              tickvaluelist=(0 .2 .4 .6 .8 1.0));

%let xOptions   = shortlabel=XNAME offsetmin=.05
                  linearopts=(viewmax=MAXTIME tickvaluelist=XTICKVALS
                              tickvaluefitpolicy=XTICKVALFITPOL);

%let Tips       = rolename=(_tip1= ATRISK _tip2=EVENT)
                  tiplabel=(_tip1="Number at Risk" _tip2="Observed Events")
                  tip=(x y _tip1 _tip2);
%let TipLabel   = tiplabel=(y="Survival Probability");
%let StepOpts   = ;

%let Groups     = group=STRATUM index=STRATUMNUM;

%let BandOpts   = displayTail=false &groups modelname="Survival";

%let InsetOpts  = autoalign=(TOPRIGHT BOTTOMLEFT TOP BOTTOM)
                  border=true BackgroundColor=GraphWalls:Color Opaque=true;
%let LegendOpts = title=GROUPNAME location=outside;

%let AtRiskOpts = display=(label) valueattrs=(size=7pt);
%let ClassOpts  = class=CLASSATRISK colorgroup=CLASSATRISK;

%let Censored   = markerattrs=(symbol=plus);
%let CensorStr  = "+ Censored";

%let GraphOpts  = ;
```

The %ProvideSurvivalMacros macro declares that these macro variables are global in scope, so you can assign values to them in your programs and have them affect the internal macros. These macro variables specify a variety of GTL options; for more information, see *SAS Graph Template Language: Reference*. The macro variables are as follows.

TitleText0 provides the common text that is used in the title for the single-stratum and multiple-strata cases. METHOD is a dynamic variable that PROC LIFETEST sets. In these examples, the value of METHOD is 'Product-Limit'; the product-limit method is also known as the Kaplan-Meier (1958) method.

TitleText1 provides the title text for the single-stratum title (relying on TitleText0).

TitleText2 provides the title text for the multiple-strata title (relying on TitleText0).

nTitles specifies the number of titles. Set the macro variable nTitles to 1 to suppress the second title line or 0 to suppress all title lines. You can add titles to the plot by adding ENTRYTITLE statements to the top of the %StmtsBeginGraph macro even when you suppress the usual titles by setting the nTitles macro variable to 0 or 1. By default, nTitles equals 2.

yOptions provides the Y-axis options. The LABEL= option provides the axis label. The SHORTLA-BEL= option provides the axis label for small plots when the LABEL= option label is too long. The LINEAROPTS= option specifies linear axis options. This and most other axes are linear axes; alternatives include log-scale axes. The VIEWMIN=0 and VIEWMAX=1 options ensure that the axis goes from 0 to 1 even when the actual results have a more restricted range. The TICKVALUELIST= option provides the tick values. Standard SAS number list abbreviations like 0 TO 1 BY 0.2 are not valid in the GTL.

xOptions provides the X-axis options. The LABEL= option is not provided, so the axis label comes from the column label in the ODS data object. You can add a LABEL= option or other axis options if you want. The SHORTLABEL= option provides the axis label for small plots when the label is too long. The short label comes from a dynamic variable that PROC LIFETEST provides. The OFFSETMIN= option ensures that there is extra space between the axis and the minimum tick mark. The LINEAROPTS= option specifies linear axis options. The VIEWMAX= option ensures that the axis goes to the value in the MAXTIME dynamic variable set by PROC LIFETEST. The TICKVALUELIST= option provides the tick values in a dynamic variable. The TICKVALUEFITPOLICY= option provides, in a dynamic variable, the approach for handling dense tick marks. Approaches include rotation, staggering, and thinning.

Tips provides options for tooltips for the step plots. Tooltips are text boxes that appear in HTML output when you rest your mouse pointer over part of the plot when the IMAGEMAP=ON option is specified in the ODS GRAPHICS statement. Tooltips are provided for the X- and Y-axis columns. Additional columns that are assigned roles (and hence are eligible to use as tooltips) include the at-risk and event columns. These columns are given the tooltip labels 'Number at Risk' and 'Observed Events'. Unless you are specifically interested in tooltips, you probably do not need to modify this macro variable.

TipLabel provides a label for the Y-axis tooltip. Unless you are specifically interested in tooltips, you do not need to modify this macro variable.

StepOpts provides options for the step functions. This macro variable is null by default. You can use this option to control the line thickness (for example, LINEATTRS=(THICKNESS=2.5)) and other aspects of the step functions.

Groups   provides the name of the data object columns that provide group names and the index that provides the order of the group names. You will probably never need to modify this macro variable.

BandOpts   provides the group information for band plots. You will probably never need to modify this macro variable.

InsetOpts   provides options for the inset table that provides the censored value legend and the homogeneity test *p*-value. The AUTOALIGN= option specifies the places in the plot where the inset table can be positioned. If your preferred placement is somewhere other than the top right corner, you can modify the automatic placement list. The BORDER= option displays a border around this table. The BACKGROUNDCOLOR= option controls the table background. By default, it matches the background color for the rest of the plot by using the `GraphWalls:Color` style reference. The OPAQUE=TRUE option specifies an opaque table that hides any graphical elements that are behind the table. You can set the InsetOpts macro variable to null to suppress the usual inset that contains the censored value and *p*-value.

LegendOpts   provides options for the external legend that identifies the strata. The title comes from a dynamic variable GroupName that the procedure sets. By default, the legend is outside the plot. Specify LOCATION=INSIDE and an AUTOALIGN= option such as the one provided in the InsetOpts macro variable if you want the legend to appear inside the plot. You can set the LegendOpts macro variable to null to suppress the legend.

AtRiskOpts   provides options for the at-risk table. The option DISPLAY=(LABEL) limits the display to labels. VALUEATTRS=(SIZE=7PT) specifies a font size of seven points.

ClassOpts   provides the options that are used in the at-risk table to distinguish groups of observations.

Censored   provides the marker (a plus sign) that is displayed in the plot to indicate censored observations.

CensorStr   provides the character for the inset table that shows how censored observations appear in the plot.

GraphOpts   provides options for the template BEGINGRAPH statement. By default, the GraphOpts macro variable is null. The following options are particularly useful:

- **ATTRPRIORITY=AUTO | NONE | COLOR** specifies the priority for varying the attributes that distinguish groups of observations. AUTO honors the setting that is otherwise in effect. COLOR varies only the color attribute. NONE simultaneously varies colors, markers, and lines. Styles such as HMTLBlue and Pearl are ATTRPRIORITY=COLOR styles, whereas styles such as DEFAULT, Statistical, Listing, and RTF are ATTRPRIORITY=NONE styles.

- **DATACOLORS=(***color-list***)** specifies the list of colors (which control confidence bands) to replace the graph data colors from the GraphData1–GraphData*N* style elements.

- **DATACONTRASTCOLORS=(***color-list***)** specifies the list of contrast colors (which control markers and lines) to replace the graph data contrast colors from the GraphData1–GraphData*N* style elements.

- **DATALINEPATTERNS=(***line-pattern-list***)** specifies the list of line patterns to replace the graph data line patterns from the GraphData1–GraphData*N* style elements. There are 46 line patterns, and you can specify each pattern by using an integer in the range 1 to 46. Some patterns have names associated with them. You can specify either the name or the number for the following number/name pairs: 1 Solid, 2 ShortDash, 4

MediumDash, 5 LongDash, 8 MediumDashShortDash, 14 DashDashDot, 15 DashDot-Dot, 20 Dash, 26 LongDashShortDash, 34 Dot, 35 ThinDot, 41 ShortDashDot, and 42 MediumDashDotDot.

- **DESIGNHEIGHT=***height* sets the graph height. You can set the graph height to the default graph width of 640 pixels by specifying the option DESIGN-HEIGHT=DEFAULTDESIGNWIDTH. Or you can specify a size in pixels, such as DESIGNHEIGHT=500PX. Although the graph is designed at the specified height, you can resize it for the actual display by using the WIDTH= and HEIGHT= options in the ODS GRAPHICS statement. By default, DESIGNHEIGHT=480PX.

- **DESIGNWIDTH=***width* sets the graph width. You can set the graph width to the default graph height of 480 pixels by specifying the option DESIGN-WIDTH=DEFAULTDESIGNHEIGHT. Or you can specify a size in pixels, such as DESIGNWIDTH=600PX. Although the graph is designed at the specified width, you can resize it for the actual display by using the WIDTH= and HEIGHT= options in the ODS GRAPHICS statement. By default, DESIGNWIDTH=640PX.

The Smaller Macros

The %ProvideSurvivalMacros macro provides four small macros that are easy for you to modify:

```
%macro StmtsBeginGraph; %mend;
%macro StmtsTop;        %mend;
%macro StmtsBottom;     %mend;

%macro pValue;
   if (PVALUE < .0001)
      entry TESTNAME " p " eval (PUT(PVALUE, PVALUE6.4));
   else
      entry TESTNAME " p=" eval (PUT(PVALUE, PVALUE6.4));
   endif;
%mend;
```

By default, the %StmtsBeginGraph, %StmtsTop, and %StmtsBottom macros are empty. You can use them to add new statements to the BEGINGRAPH block or to the beginning or end of the block of statements that define the appearance of the graph.

The %pValue macro is used to control the display of the *p*-value from the homogeneity test.

The Larger Macros

The examples and information up to this point have illustrated how you can make simple changes to the survival plot. It is unlikely that you will ever have to do more than that. If you need to make changes to the overall layout of the graph, then you must modify one of the other macros. The %CompileSurvivalTemplates macro, which is the macro that compiles all the pieces that you modified, is as follows:

```
%macro CompileSurvivalTemplates;
   %local outside;
   proc template;
      %do outside = 0 %to 1;
         define statgraph
            Stat.Lifetest.Graphics.ProductLimitSurvival%scan(2,2-&outside);
            dynamic NStrata xName plotAtRisk
               %if %nrbquote(&censored) ne %then plotCensored;
               plotCL plotHW plotEP labelCL labelHW labelEP maxTime xtickVals
               xtickValFitPol rowWeights method StratumID classAtRisk
               plotTest GroupName Transparency SecondTitle TestName pValue
               _byline_ _bytitle_ _byfootnote_;
            BeginGraph %if %nrbquote(&graphopts) ne %then / &graphopts;;

            if (NSTRATA=1)
               %if &ntitles %then %do;
                  if (EXISTS(STRATUMID)) entrytitle &titletext1;
                  else                   entrytitle &titletext0;
                  endif;
               %end;

               %if &ntitles gt 1 %then %do;
                  %if not &outside %then if (PLOTATRISK=1);
                     entrytitle "With Number of Subjects at Risk" /
                                textattrs=GRAPHVALUETEXT;
                  %if not &outside %then %do; endif; %end;
               %end;

               %StmtsBeginGraph
               %AtRiskLatticeStart
               layout overlay / xaxisopts=(&xoptions) yaxisopts=(&yoptions);
                  %StmtsTop
                  %SingleStratum
                  %StmtsBottom
               endlayout;
               %AtRiskLatticeEnd

            else
               %if &ntitles %then %do; entrytitle &titletext2; %end;
               %if &ntitles gt 1 %then %do;
                  if (EXISTS(SECONDTITLE))
                     entrytitle SECONDTITLE / textattrs=GRAPHVALUETEXT;
                  endif;
               %end;

               %StmtsBeginGraph
               %AtRiskLatticeStart
               layout overlay / xaxisopts=(&xoptions) yaxisopts=(&yoptions);
                  %StmtsTop
                  %MultipleStrata
                  %StmtsBottom
               endlayout;
               %AtRiskLatticeEnd(class)

            endif;
```

```
            if (_BYTITLE_) entrytitle _BYLINE_ / textattrs=GRAPHVALUETEXT;
            else if (_BYFOOTNOTE_) entryfootnote halign=left _BYLINE_; endif;
            endif;
            EndGraph;
        end;
      %end;
    run;
  %mend;
```

The macro %DO loop compiles the following two templates:

- **Stat.Lifetest.Graphics.ProductLimitSurvival** when the macro variable Outside is 0 and %SCAN(2,2-&OUTSIDE) is null
- **Stat.Lifetest.Graphics.ProductLimitSurvival2** when the macro variable Outside is 1 and %SCAN(2,2-&OUTSIDE) is 2

The primary difference between these templates is that when the macro variable Outside is 1, a LAYOUT LATTICE statement block is used to place the at-risk table outside the graph. When Outside is 1, the macros %AtRiskLatticeStart and %AtRiskLatticeEnd provide the LAYOUT LATTICE statement block (two cells, plot above and at-risk table below) and the LAYOUT OVERLAY statement block for the at-risk table. The %AtRiskLatticeStart and %AtRiskLatticeEnd macros are defined as follows:

```
%macro AtRiskLatticeStart;
  %if &outside %then %do;
    layout lattice / rows=2 rowweights=ROWWEIGHTS
                     columndatarange=union rowgutter=10;
    cell;
  %end;
%mend;

%macro AtRiskLatticeEnd(useclassopts);
  %if &outside %then %do;
    endcell;
    cell;
      layout overlay / walldisplay=none xaxisopts=(display=none);
        axistable x=TATRISK value=ATRISK / &atriskopts
                  %if &useclassopts ne %then &classopts;;
      endlayout;
    endcell;
  endlayout;
  %end;
%mend;
```

The %CompileSurvivalTemplates macro relies on two other macros: %SingleStratum for the single-stratum case and %MultipleStrata for the multiple-strata case. The %SingleStratum macro is as follows:

```
%macro SingleStratum;
   if (PLOTHW=1 AND PLOTEP=0)
      bandplot LimitUpper=HW_UCL LimitLower=HW_LCL x=TIME /
         displayTail=false modelname="Survival" fillattrs=GRAPHCONFIDENCE
         name="HW" legendlabel=LABELHW;
   endif;
   if (PLOTHW=0 AND PLOTEP=1)
      bandplot LimitUpper=EP_UCL LimitLower=EP_LCL x=TIME /
         displayTail=false modelname="Survival" fillattrs=GRAPHCONFIDENCE
         name="EP" legendlabel=LABELEP;
   endif;
   if (PLOTHW=1 AND PLOTEP=1)
      bandplot LimitUpper=HW_UCL LimitLower=HW_LCL x=TIME /
         displayTail=false modelname="Survival" fillattrs=GRAPHDATA1
         datatransparency=.55 name="HW" legendlabel=LABELHW;
      bandplot LimitUpper=EP_UCL LimitLower=EP_LCL x=TIME /
         displayTail=false modelname="Survival" fillattrs=GRAPHDATA2
         datatransparency=.55 name="EP" legendlabel=LABELEP;
   endif;
   if (PLOTCL=1)
      if (PLOTHW=1 OR PLOTEP=1)
         bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME /
            displayTail=false modelname="Survival" display=(outline)
            outlineattrs=GRAPHPREDICTIONLIMITS name="CL" legendlabel=LABELCL;
      else
         bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME /
            displayTail=false modelname="Survival"
            fillattrs=GRAPHCONFIDENCE
            name="CL" legendlabel=LABELCL;
      endif;
   endif;

   stepplot y=SURVIVAL x=TIME / name="Survival" &tips legendlabel="Survival"
         &stepopts;

   if (PLOTCENSORED=1)
      scatterplot y=CENSORED x=TIME / &censored &tiplabel
         name="Censored" legendlabel="Censored";
   endif;

   if (PLOTCL=1 OR PLOTHW=1 OR PLOTEP=1)
      discretelegend "Censored" "CL" "HW" "EP" / location=outside
         halign=center;
   else
      if (PLOTCENSORED=1)
         discretelegend "Censored" / location=inside
                                     autoalign=(topright bottomleft);
      endif;
   endif;
   %if not &outside %then %do;
      if (PLOTATRISK=1)
         innermargin / align=bottom;
            axistable x=TATRISK value=ATRISK / &atriskopts;
         endinnermargin;
      endif;
   %end;
```

```
      %mend;
```

The %MultipleStrata macro is as follows:

```
   %macro MultipleStrata;
     if (PLOTHW=1)
         bandplot LimitUpper=HW_UCL LimitLower=HW_LCL x=TIME / &bandopts
                 datatransparency=Transparency;
     endif;
     if (PLOTEP=1)
         bandplot LimitUpper=EP_UCL LimitLower=EP_LCL x=TIME / &bandopts
                 datatransparency=Transparency;
     endif;
     if (PLOTCL=1)
        if (PLOTHW=1 OR PLOTEP=1)
           bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME / &bandopts
                   display=(outline) outlineattrs=(pattern=ShortDash);
        else
           bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME / &bandopts
                   datatransparency=Transparency;
        endif;
     endif;

     stepplot y=SURVIVAL x=TIME / &groups name="Survival" &tips &stepopts;

     if (PLOTCENSORED=1)
        scatterplot y=CENSORED x=TIME / &groups &tiplabel &censored;
     endif;

     %if not &outside %then %do;
        if (PLOTATRISK=1)
           innermargin / align=bottom;
              axistable x=TATRISK value=ATRISK / &atriskopts &classopts;
           endinnermargin;
        endif;
     %end;

     %if %nrbquote(&legendopts) ne %then %do;
        DiscreteLegend "Survival" / &legendopts;
     %end;

     %if %nrbquote(&insetopts) ne %then %do;
        if (PLOTCENSORED=1)
           if (PLOTTEST=1)
              layout gridded / rows=2 &insetopts;
                 entry &censorstr;
                 %pValue
              endlayout;
           else
              layout gridded / rows=1 &insetopts;
                 entry &censorstr;
              endlayout;
           endif;
        else
```

```
            if (PLOTTEST=1)
               layout gridded / rows=1 &insetopts;
                  %pValue
               endlayout;
            endif;
         endif;
      %end;

   %mend;
```

Event Table Macros

All the macros and macro variables that have been described up to this point are used in defining the two survival plot graph templates. Some macros (%StmtsTop and %StmtsBottom) and macro variables (StepOpts and GraphOpts) are null and do not affect the generated template code, but all are resolved somewhere in the process of producing the templates. In contrast, the macros %SurvTabHeader, %SurvivalTable, and %SurvivalSummaryTable are not used by default. They are available for you to use to add more statements to the templates.

The %SurvTabHeader macro provides the headings for the event table:

```
%macro SurvTabHeader(multiple);
   %if &multiple %then %do; entry ""; %end;
   entry "";
   entry &r "Median";
   entry "";

   %if &multiple %then %do; entry ""; %end;
   entry &r "Subjects";
   entry &r "Event";
   entry &r "Censored";
   entry &r "Survival";
   entry &r PctMedianConfid;
   entry halign=left  "CL";
%mend;
```

This table is not displayed by default. There are two types of headings: one for multiple strata and one for a single stratum.

The %SurvivalTable macro provides the body of the event table:

```
%macro SurvivalTable;
   %local fmt r i t;
   %let fmt = bestd6.;
   %let r = halign = right;
   columnheaders;
      layout overlay / pad=(top=5);
         if(NSTRATA=1)
            layout gridded / columns=6 border=TRUE;
               dynamic PctMedianConfid NObs NEvent Median
                       LowerMedian UpperMedian;
               %SurvTabHeader(0)
```

```
                   entry &r NObs;
                   entry &r NEvent;
                   entry &r eval(NObs-NEvent);
                   entry &r eval(put(Median,&fmt));
                   entry &r eval(put(LowerMedian,&fmt));
                   entry &r eval(put(UpperMedian,&fmt));
                endlayout;
            else
                layout gridded / columns=7 border=TRUE;
                    dynamic PctMedianConfid;
                    %SurvTabHeader(1)
                    %do i = 1 %to 10;
                        %let t = / textattrs=GraphData&i;
                        dynamic StrVal&i NObs&i NEvent&i Median&i
                                LowerMedian&i UpperMedian&i;
                        if (&i <= nstrata)
                            entry &r StrVal&i &t;
                            entry &r NObs&i &t;
                            entry &r NEvent&i &t;
                            entry &r eval(NObs&i-NEvent&i) &t;
                            entry &r eval(put(Median&i,&fmt)) &t;
                            entry &r eval(put(LowerMedian&i,&fmt)) &t;
                            entry &r eval(put(UpperMedian&i,&fmt)) &t;
                        endif;
                    %end;
                endlayout;
            endif;
        endlayout;
    endcolumnheaders;
%mend;
```

The %SurvivalSummaryTable macro redefines the %AtRiskLatticeStart and %AtRiskLatticeEnd macros so that they provide the body of the event table:

```
%macro SurvivalSummaryTable;
    %macro AtRiskLatticeStart;
        layout lattice / columndatarange=union rowgutter=10
            rows=%if &outside %then 2 rowweights=ROWWEIGHTS;
                 %else             1;;
        %if &outside %then %do; cell; %end;
    %mend;

    %macro AtRiskLatticeEnd(useclassopts);
        %if &outside %then %do;
            endcell;
            cell;
                layout overlay / walldisplay=none xaxisopts=(display=none);
                    axistable x=TATRISK value=ATRISK / &atriskopts
                              %if &useclassopts ne %then &classopts;;
                endlayout;
            endcell;
        %end;
        %SurvivalTable
        endlayout;
    %mend;
%mend;
```

If you want to create an event table like the one displayed in Figure 23.30, you only need to call the %SurvivalSummaryTable macro. If you want to modify the table, then you need to modify the %SurvTabHeader and %SurvivalTable macros.

Dynamic Variables

Graph templates consist of instructions, written by SAS developers, in conjunction with SAS procedure code. However, SAS developers cannot fully provide some instructions when the template is written, because some elements of some graphs cannot be known until the procedure runs. For example, the legend title in a graph that has multiple strata corresponds to the label or name of the stratification variable, and the procedure calculates the *p*-value for the homogeneity test. SAS procedures create dynamic variables to provide some run-time information to graphs.[8] Some dynamic variables are set by the procedure and are declared in the template. Other dynamic variables are also set by the procedure, but you must declare them directly or through the template modification macros before you can use them.

Dynamic Variables That Are Automatically Declared

The primary dynamic variables are as follows:

ByFootNote   is a binary variable that, when true, displays the BY-group BY line as a footnote.

ByLine   is a character variable that provides the BY-group BY line.

ByTitle   is a binary variable that, when true, displays the BY-group BY line as a title.

ClassAtRisk   is a character variable that names the data object column that contains the classification (stratification) values.

GroupName   is a character variable that contains the stratification legend title.

LabelCL   is a character variable that contains the label for the confidence limits legend entry (including the percent sign).

LabelEP   is a character variable that contains the label for the equal-precision band legend entry (including the percent sign).

LabelHW   is a character variable that contains the label for the Hall-Wellner band legend entry (including the percent sign).

MaxTime   is a numeric variable that contains the maximum value to display on the X axis.

Method   is a character variable that contains the method for the plot title.

NStrata   is an integer variable that contains the number of strata.

PValue   is a numeric variable that contains the *p*-value for the homogeneity test.

PlotAtRisk   is a binary variable that, when true, is used to display the at-risk table.

PlotCensored   is a binary variable that, when true, displays the censored values on the step functions.

---

[8]Axis labels can be set directly in the template or at run time through dynamic variables or through data object column labels.

PlotCL	is a binary variable that, when true, displays the pointwise confidence limits.
PlotEP	is a binary variable that, when true, displays the equal-precision band.
PlotHW	is a binary variable that, when true, displays the Hall-Wellner confidence band.
PlotTest	is a binary variable that, when true, displays the *p*-value for the homogeneity test.
RowWeights	is a pair of relative heights of the plot and the external at-risk table.
SecondTitle	is a character variable that provides the second title line.
StratumID	is a character variable that provides the value of the stratification variable for the single stratum case.
TestName	is a character variable that provides the name of the homogeneity test (for example, 'logrank').
Transparency	is a numeric variable that provides the transparency for the confidence bands in the multiple strata case.
XName	is a character variable that contains a short label for the X axis, which might be used in place of the ordinary X-axis label when the ordinary label is long or the plot is small.
XtickValFitPol	is a character variable that contains the option for handling dense tick values on the X axis.
XtickVals	is a list of X-axis tick values.

Additional Dynamic Variables

PROC LIFETEST passes to the survival plots a number of dynamic variables that contain summary statistics. Some of these dynamic variables are used when you call the %SurvivalSummaryTable macro. Table 23.1 and Table 23.2 list these additional dynamic variables for the Kaplan-Meier curves and the life-table curves, respectively. These dynamic variables are not declared in the templates for the survival curves, but you can declare them and use them to enhance the default plots.[9] The names of the dynamic variables depend on the STRATA= suboption of the PLOTS=SURVIVAL option: STRATA=INDIVIDUAL produces a separate plot for each stratum, and STRATA=OVERALL produces one plot that has overlaid curves.

Table 23.1 Additional Dynamic Variables for
`Stat.Graphics.ProductLimitSurvival`

STRATA=	Dynamic	Description
OVERLAY	StrVal*j*	Label for the *j*th stratum
	NObs*j*	Number of observations in the *j*th stratum
	NEvent*j*	Number of events in the *j*th stratum
	Median*j*	Median survival time of the *j*th stratum
	LowerMedian*j*	Lower median survival time of the *j*th stratum
	UpperMedian*j*	Upper median survival time of the *j*th stratum
	PctMedianConfid	Confidence of the median intervals, in percentage

---

[9]Because the number of dynamic variables is a function of the number of strata, the template definition cannot automatically contain the correct number of dynamic variables.

Table 23.1 *continued*

STRATA=	Dynamic	Description
INDIVIDUAL	NObs	Number of observations
	NEvent	Number of events
	Median	Median survival time
	LowerMedian	Lower median survival time
	UpperMedian	Upper median survival time
	PctMedianConfid	Confidence of the median intervals, in percentage

Table 23.2 Additional Dynamic Variables for
`Stat.Graphics.LifetableSurvival`

STRATA=	Dynamic	Description
OVERLAY	StrVal*j*	Label for the *j*th stratum
	NObs*j*	Number of observations in the *j*th stratum
	NEvent*j*	Number of events in the *j*th stratum
INDIVIDUAL	NObs	Number of observations
	NEvent	Number of events

Style Templates

Graphs that are produced by ODS Graphics are controlled by the data object (the matrix of information that is graphed), the graph template (the program that controls how a specific graph is constructed), and a style template (a program that controls the overall appearance of graphs, including colors, line and marker styles, sizes, fonts, and so on). Although it is rarely necessary, you can use different styles or modify styles to change the appearance of all graphs, including the survival plot. In the past, you could make certain Kaplan-Meier plot modifications only through style modifications. However, with the addition of the DATACOLORS=, DATACONTRASTCOLORS=, and DATALINEPATTERNS= options in the GTL, you no longer have to modify styles in order to modify how groups of observations are displayed. This section shows you how to change styles, extract group color and other information from styles, and modify styles.

Changing the Style

The graphs that have been displayed up to this point were all produced by using the HTMLBlue style, which is the default style for the HTML destination in the SAS windowing environment. This is an all-color style (because of the ATTRPRIORITY='Color' option); it does not rely on line style or marker changes to differentiate groups. You can switch to a style that varies colors, markers, and lines by specifying the STYLE= option in an ODS destination statement. You can use the HTMLBlueCML style as follows to make a graph whose line patterns differ:

```
ods html style=htmlbluecml image_dpi=300;
proc lifetest data=sashelp.BMT
            plots=survival(cb=hw test atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
ods html close;
```

The results are displayed in Figure 23.33. This example also illustrates specifying the IMAGE_DPI= option to control the resolution (measured in dots per inch, or DPI) of the image. All images in this chapter are created at 300 DPI. The default setting for the HTML destination is 100 DPI. Images that are created at 300 DPI are clearer than images created at 100 DPI, but they require about nine times as much disk space.

Figure 23.33 Line and Color Group Differentiation

Color Priority Styles

You can use the HTMLBlue or Pearl style when you want to distinguish groups only by color. Alternatively, you can easily modify any other style to be an all-color style like HTMLBlue or Pearl by using the ATTRPRIORITY='Color' option:[10]

```
proc template;
   define style styles.ListingColor;
      parent = styles.Listing;
      style Graph from Graph / attrpriority = "Color";
   end;
run;
```

You need to specify the new style name in an ODS destination statement, as in the following:

```
ods html style=ListingColor image_dpi=300;
proc lifetest data=sashelp.BMT
              plots=survival(cb=hw test atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
ods html close;
```

The results are displayed in Figure 23.34.

Figure 23.34 ATTRPRIORITY='Color' Style

---

[10]The style option is ATTRPRIORITY=*quoted-string*, whereas the GTL option is ATTRPRIORITY=*keyword*.

Displaying a Style and Extracting Color Lists

You can use PROC TEMPLATE with the SOURCE statement to display a style as follows:

```
proc template;
   source styles.htmlblue;
run;
```

The results of this step, which are not shown, include the option PARENT=STYLES.STATISTICAL and do not include definitions of the colors (**gData1**, **gData2**, ..., **gData12**) and contrast colors (**gcData1**, **gcData2**, ..., **gcData12**). These are the color definitions that are used in the style elements **GraphData1**, **GraphData2**, ..., **GraphData12**. You can examine the parent Statistical style as follows:

```
proc template;
   source styles.statistical;
run;
```

The results of this step are not shown because they are hard to interpret in their raw form, but the desired color definitions are included. You can submit the following statements to display the colors for the Statistical (and hence HTMLBlue) style in a more understandable form:

```
proc template;
   source styles.statistical / file='style.tmp';
run;

data colors;
   length element Color $ 20;
   infile 'style.tmp';
   input;
   if index(_infile_, 'data') then do;
      element = scan(_infile_, 1, ' ');
      Color = scan(_infile_, 3, ' ;');
      Type = ifc(index(element, 'gc'), 'Line', 'Fill') || ' Colors';
      i = input(compress(element, 'gcdat'';'), ?? 2.);
      if i then output;
   end;
run;

proc sort; by descending type i; run;

proc print; id type; by descending type; var color; run;
```

The results are displayed in Figure 23.35. All colors are specified in values of the form CX*rrggbb*, where the last six characters specify RGB (red, green, blue) values on the hexadecimal scale of 00 to FF (0 to 255, base 10).

Figure 23.35 HMTLBlue Style Colors List

Type=Line Colors

Type	Color
Line Colors	cx445694
	cxA23A2E
	cx01665E
	cx543005
	cx9D3CDB
	cx7F8E1F
	cx2597FA
	cxB26084
	cxD17800
	cx47A82A
	cxB38EF3
	cxF9DA04

Type	Color
Fill Colors	cx6F7EB3
	cxD05B5B
	cx66A5A0
	cxA9865B
	cxB689CD
	cxBABC5C
	cx94BDE1
	cxCD7BA1
	cxCF974B
	cx87C873
	cxB7AEF1
	cxDDD17E

You can use the following steps to display the `GraphData1` – `GraphData12` line and fill colors (contrast colors and colors, respectively):

```
data display;
   array y[12] y1 - y12;
   do i = 1 to 12; y[i] = i;       end;
   do x = 1 to 10; output;         end;
   do i = 1 to 12; y[i] = i + .5; end;
   do x = 1 to 10; output;         end;
run;

data _null_;
```

```
   set colors;
   call symputx(compress(type || put(i, 2.)), color);
run;


proc sgplot noautolegend data=display;
   %macro reg;
      title 'Line and Fill Colors, Respectively';
      %do i = 1 %to 12;
         reg y=y%eval(13-&i) x=x / lineattrs=GraphData&i clmattrs=GraphData&i
                      nomarkers clm  curvelabelpos=max
                      curvelabel="  GraphData&i &&LineColors&i &&FillColors&i";
      %end;
   %mend;
   %reg
   xaxis display=none;
   yaxis display=none;
run;


   title;
```

The results are displayed in Figure 23.36. The colors in Figure 23.36 are richer than the colors in the bands in the survival plots because of the DATATRANSPARENCY= options in the BANDPLOT statements.

Figure 23.36 HTMLBlue Style Colors Display

Modifying Color Lists

You can use the information in Figure 23.36 to specify the desired colors in the graph template. You can copy the third, second, and first colors from each list and switch the colors as follows:

```
%ProvideSurvivalMacros

%let GraphOpts = DataContrastColors=(cx01665E cxA23A2E cx445694)
                 DataColors=(cx66A5A0 cxD05B5B cx6F7EB3);

%CompileSurvivalTemplates

proc lifetest data=sashelp.BMT
              plots=survival(cb=hw test atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.37. The familiar colors are used, but they are now in a different order. The next section shows you how to modify a style template to change the color order without having to extract the original color names.

Figure 23.37 Swapping Colors from the Style

You can use the information in Figure 23.36 to modify the style template, but the next example shows an easier way.

Swapping Colors among Style Elements

You can modify the colors in a style as follows:

```
%macro reorder(from, to, list);

   proc template;
      define style styles.&to;
         parent=styles.&from;
         %do i = 1 %to 12;
            %let s = %scan(&list, &i);
            %if &s ne %then %do;
               style GraphData&i from GraphData&i /
                     contrastcolor = GraphColors("gcdata&s")
                     color = GraphColors("gdata&s");
            %end;
         %end;
      end;
   run;

%mend;

%reorder(htmlblue,   /* Parent style.                              */
         MyStyle,    /* New style to create.  Specify it in an ODS */
                     /* destination statement.                     */
         3 2 1)      /* Replace the first few GraphData colors     */
                     /* (1 2 3) with the colors from the specified */
                     /* GraphData style elements (3 2 1).          */
                     /* You can specify up to 12 integers in the   */
                     /* range 1 - 12.                              */
```

The %Reorder macro creates a new style called MyStyle that inherits most of its attributes from the HTMLBlue style. However, in the new style, the colors for groups 1, 2, and 3 have been replaced by the colors for groups 3, 2, and 1. In other words, the colors for `GraphData1` and `GraphData3` have been switched.

The following step creates the plot:

```
ods html style=mystyle;
proc lifetest data=sashelp.BMT
              plots=survival(cb=hw test atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
ods html close;
```

The survival plot is not shown, but it matches the plot in Figure 23.37.

The rest of this section is optional. It explains how you can directly modify colors in a style template when the %Reorder macro or the technique illustrated in the section "Changing the Group Color" on page 832 is not sufficient.

The source code for the MyStyle style (as generated by the %Reorder macro) is as follows:

```
proc template;
   define style Styles.MyStyle;
      parent = styles.htmlblue;
      style GraphData1 from GraphData1 /
         color = GraphColors('gdata3')
         contrastcolor = GraphColors('gcdata3');
      style GraphData2 from GraphData2 /
         color = GraphColors('gdata2')
         contrastcolor = GraphColors('gcdata2');
      style GraphData3 from GraphData3 /
         color = GraphColors('gdata1')
         contrastcolor = GraphColors('gcdata1');
   end;
run;
```

You can create a modified style that has direct color specifications by using the colors in Figure 23.35 as follows:

```
proc template;
   define style Styles.MyStyle;
      parent = styles.htmlblue;
      style GraphData1 from GraphData1 /
         color = cx66A5A0
         contrastcolor = cx01665E;
      style GraphData2 from GraphData2 /
         color = cxD05B5B
         contrastcolor = cxA23A2E;
      style GraphData3 from GraphData3 /
         color = cx6F7EB3
         contrastcolor = cx445694;
   end;
run;
```

You can define additional GraphData*N* style elements as well. For more information about how to define style elements, see the section "Displaying Other Style Elements" on page 876.

You can delete the new style template as follows:

```
proc template;
   delete Styles.MyStyle / store=sasuser.templat;
run;
```

Displaying a Style and Extracting Font Information

You can use PROC TEMPLATE with the SOURCE statement to display a style and its parent styles in the SAS log:

```
proc template;
    source styles.htmlblue / expand;
run;
```

The results of this step are long and are not shown. You can write a copy of the style templates to a file as follows:

```
proc template;
    source styles.htmlblue / expand file='style.tmp';
run;
```

The EXPAND option writes the specified style (HTMLBlue), followed by its parent (Statistical), and followed by the parent's parent (DEFAULT) to the file. The following step extracts and displays the first place in the file that the graph fonts are defined (which make the final decision in the style template):

```
data _null_;
    infile 'style.tmp' pad;
    input line $char80.;
    file print;
    if index(lowcase(line), ' graphfonts ') then y + 1;
    if y then put line $char80.;
    if y and index(line, ';') then stop;
run;
```

The results are displayed in Figure 23.38.

Figure 23.38 Graph Font Definition

```
class GraphFonts /
    'NodeDetailFont' = ("<sans-serif>, <MTsans-serif>",7pt)
    'NodeInputLabelFont' = ("<sans-serif>, <MTsans-serif>",9pt)
    'NodeLabelFont' = ("<sans-serif>, <MTsans-serif>",9pt)
    'NodeTitleFont' = ("<sans-serif>, <MTsans-serif>",9pt)
    'GraphDataFont' = ("<sans-serif>, <MTsans-serif>",7pt)
    'GraphUnicodeFont' = ("<MTsans-serif-unicode>",9pt)
    'GraphValueFont' = ("<sans-serif>, <MTsans-serif>",9pt)
    'GraphLabel2Font' = ("<sans-serif>, <MTsans-serif>",10pt)
    'GraphLabelFont' = ("<sans-serif>, <MTsans-serif>",10pt)
    'GraphFootnoteFont' = ("<sans-serif>, <MTsans-serif>",10pt)
    'GraphTitleFont' = ("<sans-serif>, <MTsans-serif>",11pt,bold)
    'GraphTitle1Font' = ("<sans-serif>, <MTsans-serif>",14pt,bold)
    'GraphAnnoFont' = ("<sans-serif>, <MTsans-serif>",10pt);
```

If the **GraphFonts** style element is defined in the HTMLBlue style, then it will appear first in the file, followed by the definitions from the Statistical style and then the DEFAULT style. In this case, the **GraphFonts** style element is defined in the DEFAULT style (last in the file), which is overridden by a definition in the Statistical style (closer to the top of the file); that is the definition that is inherited by the HTMLBlue style and displayed in Figure 23.38.
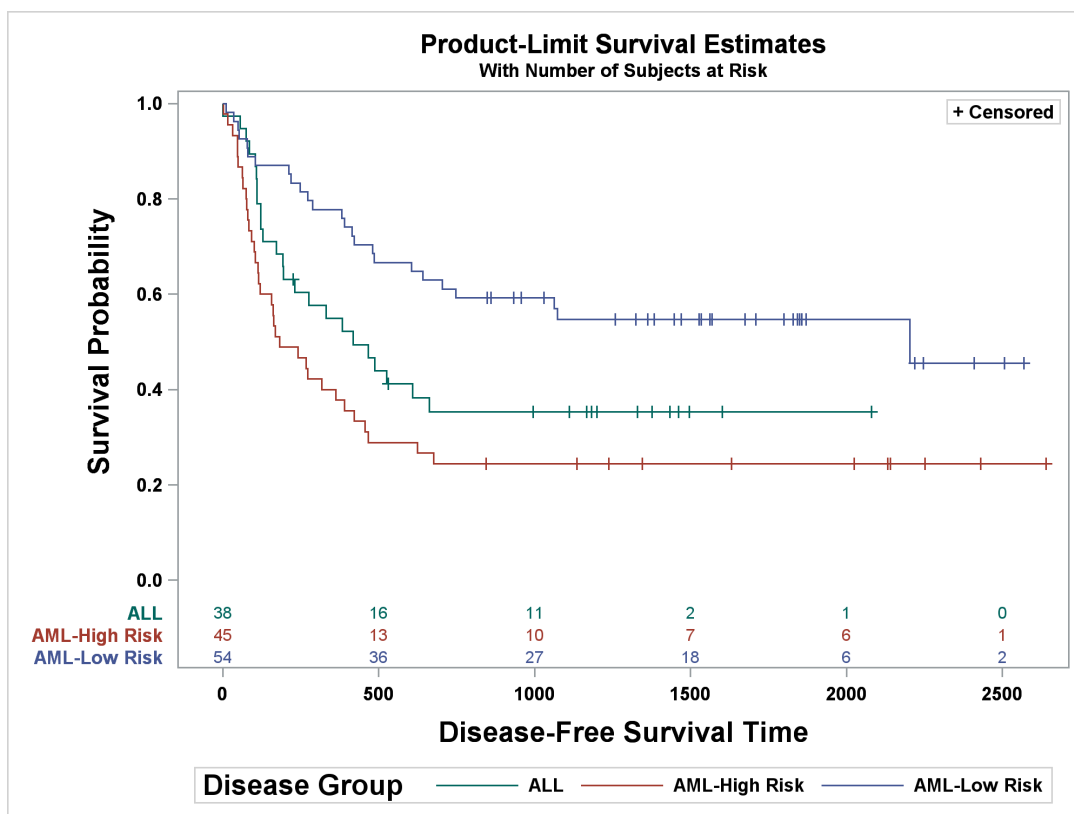
The following step creates a new style, BigFont, that changes the `GraphLabelFont` style element from a regular 10-point font to a bold 12-point font and changes the `GraphValueFont` style element from a regular 9-point font to a bold 8-point font:

```
proc template;
   define style Styles.BigFont;
      parent = Styles.HTMLBlue;
      style graphfonts from graphfonts /
         'GraphLabelFont' = ("<sans-serif>, <MTsans-serif>",12pt,bold)
         'GraphValueFont' = ("<sans-serif>, <MTsans-serif>",8pt,bold);
   end;
run;
```

The following step creates the plot that is displayed in Output 23.39:

```
ods html style=BigFont;
proc lifetest data=sashelp.BMT plots=survival(maxlen=13 atrisk);
   time T * Status(0);
   strata Group;
run;
ods html close;
```

Figure 23.39 Font Modifications

You can delete the new style template as follows:

```
proc template;
   delete Styles.BigFont / store=sasuser.templat;
run;
```

For information about making ad hoc font changes in the graph templates rather than making more global font changes in style templates, see the section "Changing the Font" on page 834.

Displaying Other Style Elements

You can use the approach from the previous example to display other style elements:

```
proc template;
   source styles.htmlblue / expand file='style.tmp';
run;

data _null_;
   infile 'style.tmp' pad;
   input line $char80.;
   file print;
   if index(lowcase(line), ' graphdata1 ') then y + 1;
   if y then put line $char80.;
   if y and index(line, ';') then stop;
run;
```

This example displays the **GraphData1** style element. The results are displayed in Figure 23.40.

Figure 23.40 GraphData1 Style Element

```
class GraphData1 /
   markersymbol = "circle"
   linestyle = 1
   contrastcolor = GraphColors('gcdata1')
   color = GraphColors('gdata1');
```

The following steps display all the **GraphFonts** style elements from all the styles:

```
proc template;
   source styles / file='style.tmp';
run;

data _null_;
   infile 'style.tmp' pad;
   length style $ 80;
   retain style;
   input line $char80.;
   file print;
   if index(lowcase(line), 'define style') then style = line;
   if index(lowcase(line), ' graphfonts ') then do;
      y + 1;
      put style $char80.;
   end;
   if y then put line $char80.;
   if index(line, ';') then y = 0;;
run;
```

The results of this step are not displayed. You can use this approach to help you better understand the options that are available for modifying styles. The SOURCE statement specifies a single-level value of STYLES rather than a specific style name such as STYLES.HTMLBLUE, so all templates that begin with STYLES as the first level (all style templates) are written to the file. The DATA step displays all definitions of **GraphFonts** and the names of all styles that define the **GraphFonts** style element.

You can insert the name of another style element (in lowercase with a leading and trailing blank) in the preceding programs in place of 'graphdata1' or 'graphfonts'. After you display a style element, you can modify the definition and create a new style that uses the modified definition, as in the example in the section "Displaying a Style and Extracting Font Information" on page 874. Some of the style elements that you might want to display and modify are listed in Table 23.3.

Table 23.3 Style Elements

AfterCaption	GraphAnnoText	GraphHeaderBackground	List3
Batch	GraphAxisLines	GraphHistogram	ListItem
Body	GraphBackground	GraphInitial	Note
BodyDate	GraphBand	GraphLabel2Text	NoteBanner
ByContentFolder	GraphBar	GraphLabelText	NoteContentFixed
Byline	GraphBlock	GraphLegendBackground	Output
BylineContainer	GraphBlockHeader	GraphMissing	PageNo
Caption	GraphBorderLines	GraphOther	Pages
Color_list	GraphBox	GraphOutlier	PagesProcLabel
Colors	GraphBoxMean	GraphOutlines	PagesTitle
Container	GraphBoxMedian	GraphOverflow	Paragraph
ContentFolder	GraphBoxWhisker	GraphPhaseBox	Parskip
ContentProcLabel	GraphClipping	GraphPrediction	PrePage
ContentTitle	GraphColors	GraphPredictionLimits	ProcTitle
Contents	GraphConfidence	GraphReference	ProcTitleFixed
Continued	GraphConfidence2	GraphRunTest	RowFooter
Data	GraphConnectLine	GraphSelection	RowFooterEmphasis
DataEmphasis	GraphContour	GraphStars	RowFooterEmphasisFixed
DataEmphasisFixed	GraphControlLimits	GraphTitle1Text	RowFooterFixed
DataFixed	GraphData1	GraphTitleText	RowFooterStrong
DataStrong	GraphData2	GraphUnderflow	RowFooterStrongFixed
DataStrongFixed	GraphData3	GraphUnicodeText	RowHeader
Date	GraphData4	GraphValueText	RowHeaderEmphasis
Document	GraphData5	GraphWalls	RowHeaderEmphasisFixed
DropShadowStyle	GraphData6	GraphZoneA	RowHeaderFixed
ErrorBanner	GraphData7	GraphZoneB	RowHeaderStrong
ErrorContentFixed	GraphData8	GraphZoneC	RowHeaderStrongFixed
ExtendedPage	GraphData9	Header	SysTitleAndFooterContainer
FatalBanner	GraphData10	HeaderEmphasis	SystemFooter
FatalContentFixed	GraphData11	HeaderEmphasisFixed	SystemTitle
Fonts	GraphData12	HeaderFixed	Table
Footer	GraphDataDefault	HeaderStrong	ThreeColorAltRamp
FooterEmphasis	GraphDataText	HeaderStrongFixed	ThreeColorRamp
FooterEmphasisFixed	GraphEllipse	HeadersAndFooters	TitleAndNoteContainer
FooterFixed	GraphError	Index	TitlesAndFooters
FooterStrong	GraphFinal	IndexItem	TwoColorAltRamp
FooterStrongFixed	GraphFit	IndexProcName	TwoColorRamp
Frame	GraphFit2	IndexTitle	UserText
Graph	GraphFloor	LayoutContainer	WarnBanner
GraphAltBlock	GraphFonts	LineContent	WarnContentFixed
GraphAnnoLine	GraphFootnoteText	List	
GraphAnnoShape	GraphGridLines	List2	

SAS Item Stores

In other sections of this chapter, you submit PROC TEMPLATE statements (either directly or through macros) to compile and save graph and style templates. Compiled templates are stored in special SAS files called item stores. Assuming that you have not modified your ODS path with an ODS PATH statement, the templates that you compile are stored in an item store in the Sasuser library. By default, all templates that SAS provides are stored in an item store in the Sashelp library. By default, the Sashelp item store has read access only; you cannot write to it. By default, the Sasuser item store has update access; you can both read and write to it. **CAUTION:** Never set the Sashelp item store to update or write access. If you do, and if you have administrator privileges on your computer, you could corrupt the Sashelp item store.

Assuming that the default ODS path is used, ODS first looks for a template in the Sasuser item store. If it does not find the template there, ODS next looks for the template in the Sashelp item store. Files in the Sasuser library persist across SAS sessions until you delete them. You can run the following step to delete the entire Sasuser item store (including all compiled graph and style templates that you added or modified) so that ODS uses only the templates the SAS System supplies:

```
ods path sashelp.tmplmst(read);
proc datasets library=sasuser nolist;
   delete templat(memtype=itemstor);
run;
ods path reset;
```

For more information about the ODS path and SAS item stores, see Chapter 21, "Statistical Graphics Using ODS."

References

Hall, W. J., and Wellner, J. A. (1980). "Confidence Bands for a Survival Curve from Censored Data." *Biometrika* 67:133–143.

Kaplan, E. L., and Meier, P. (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association* 53:457–481.

Klein, J. P., and Moeschberger, M. L. (1997). *Survival Analysis: Techniques for Censored and Truncated Data*. New York: Springer-Verlag.

Ready to take your SAS® and JMP® skills up a notch?

Be among the first to know about new books,
special events, and exclusive discounts.
support.sas.com/newbooks

Share your expertise. Write a book with SAS.
support.sas.com/publish

sas.com/books
for additional books and resources.

§sas

THE POWER TO KNOW®