SAS® GLOBAL FORUM 2021

Paper 1126-2021

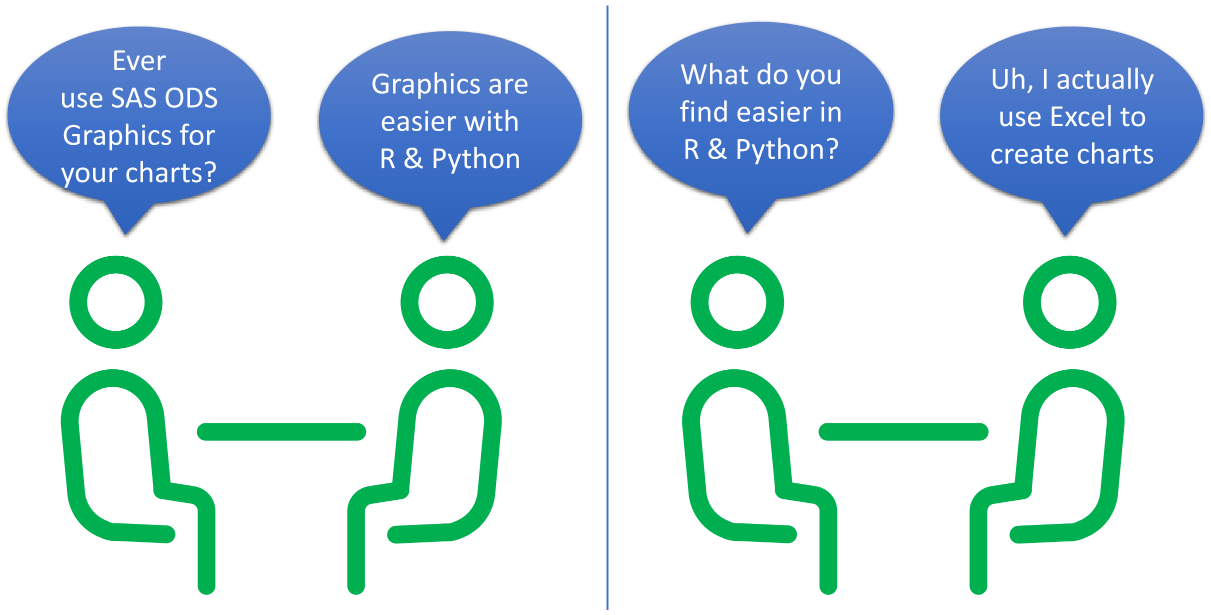
Hello Graphics World: Basic Charts in SAS® , R and Python

Ted Conway, Chicago, IL

ABSTRACT

In this presentation, we'll take a look at what it takes to generate fundamental chart types using SAS (ODS Graphics), R (ggplot2) and Python (Matplotlib). Attend this session to get a feel for some of the pluses and minuses of each language's approach. You'll see that there's often no need to leave the friendly confines of Base SAS to address your charting needs!

INTRODUCTION


Ever use SAS ODS Graphics for your charts?
Graphics are easier with R & Python
What do you find easier in R & Python?
Uh, I actually use Excel to create charts

Credit: Interview icon designed by visual language on the @NounProject

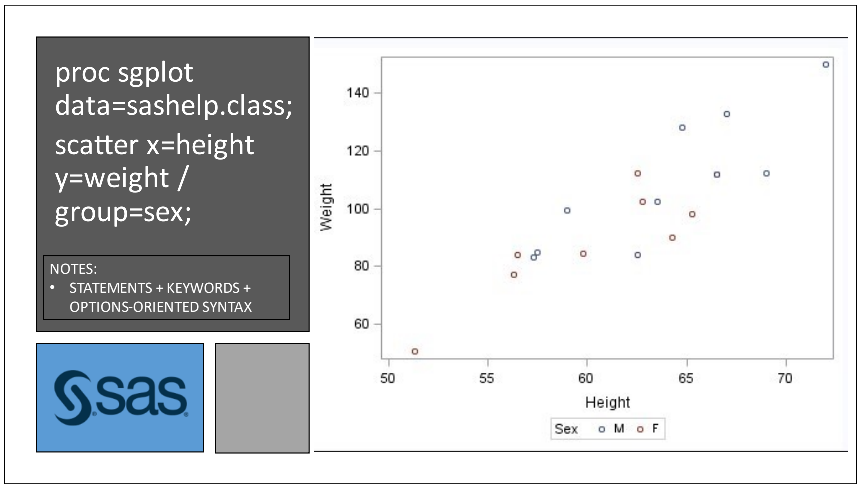
Some Data Science folks – even SAS users who may be aware that ODS Graphics is behind much of the graphical procedure output they use – have the perception that SAS doesn't do custom chart output a la R and Python. Let's dispel that notion and show why SAS ODS Graphics should be in your analytical toolbox if it isn't already!

HELLO GRAPHICS WORLD

You may be familiar with "Hello World" programs, small programs that simply display the message "Hello World" in different programming languages (e.g., **%put Hello World;** in the SAS macro language). With just a few lines of code, "Hello World" programs help illustrate the basic syntax of various coding languages.

In the spirt of "Hello World" programs, we'll take a look at some "Hello Graphics World" programs that illustrate the basic syntax used to create fundamental chart types in SAS, R, and Python. Each of these languages has a number of graphics frameworks. Here we'll be using core plotting libraries – ODS graphics for SAS, ggplot2 for R and Matplotlib for Python.

HELLO GRAPHICS WORLD (CONT.)

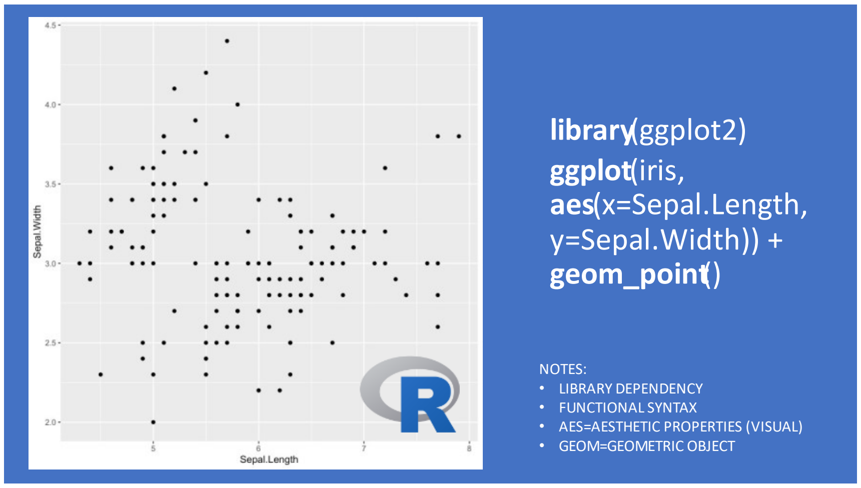


```
proc sgplot
data=sashelp.class;
scatter x=height
y=weight /
group=sex;
```

NOTES:
- STATEMENTS + KEYWORDS + OPTIONS-ORIENTED SYNTAX

SAS Scatter Plot

Here's a SAS ODS Graphics code snippet that produces a simple scatter plot from a SAS dataset with just a few lines of code using PROC SGPLOT. The syntax will be very familiar to any SAS programmer.

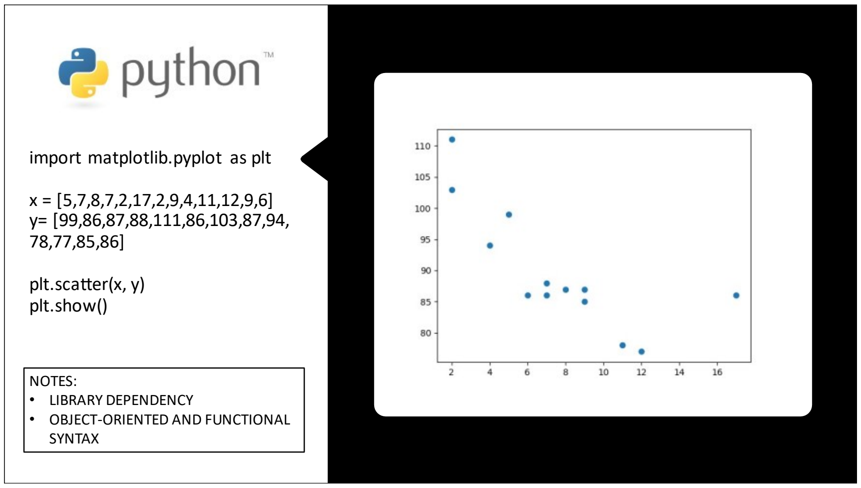


```
library(ggplot2)
ggplot(iris,
aes(x=Sepal.Length,
y=Sepal.Width)) +
geom_point()
```

NOTES:
- LIBRARY DEPENDENCY
- FUNCTIONAL SYNTAX
- AES=AESTHETIC PROPERTIES (VISUAL)
- GEOM=GEOMETRIC OBJECT

R Scatter Plot

And here's a simple scatter plot in R created from a dataframe using ggplot2. The gridded grey background is from a default R style template. You'll note a more functional syntax than the above SAS example. ggplot is a separate R library, so you have to specify a library reference.



```
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y= [99,86,87,88,111,86,103,87,94,
78,77,85,86]

plt.scatter(x, y)
plt.show()
```

NOTES:
- LIBRARY DEPENDENCY
- OBJECT-ORIENTED AND FUNCTIONAL SYNTAX

Python Scatter Plot

Finally, here's some Python code to generate a scatter plot from arrays using Matplotlib. Here, you'll note the syntax is a mix of functional and object-oriented.

WHY CREATE CHARTS WITH CODING?

Credit: **Harris & Ewing**

There are other alternatives for producing graphics, such as Excel and interactive data visualization software like Power BI, Tableau and SAS Visual Analytics. So why bother with SAS ODS Graphics, ggplot2, or Matplotlib?

Well, the graphics coding frameworks associated with SAS, R, and Python share common goals, offering features that enable the automated "hands-off" production of charts from within a programming language. Among other things, the coding frameworks:

- ✓ Provide an easy-to-use, flexible, rich feature set for data visualization
- ✓ Make a wealth of single and multiple-panel chart types available
- ✓ Facilitate automated chart creation
- ✓ Promote reproducible, repeatable results
- ✓ Scale to produce dozens, hundreds, even thousands of precisely-sized and formatted charts with a consistent "look and feel"
- ✓ Fully-integrate with the programming language, giving one access to powerful analytic and data transformation features to make data chart-ready in no time
- ✓ Enable viewers to quickly gain an understanding of data
- ✓ Can be readily reviewed for walkthroughs, audits, compliance

HELLO GRAPHICS WORLD – BASIC CHART TYPES

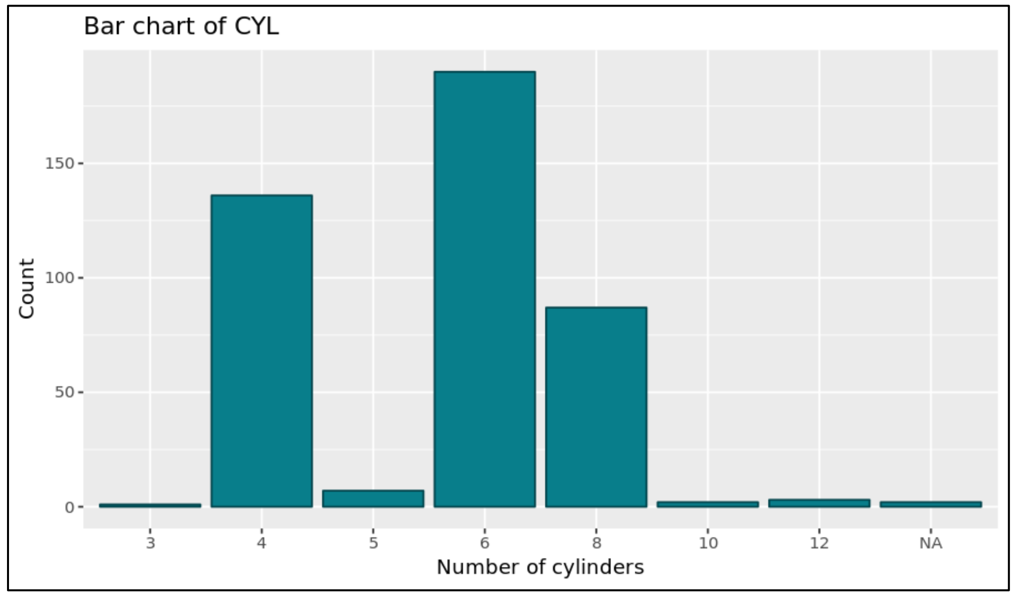
Even though they're all trying to achieve similar goals and results, there are some big differences in how the three languages' graphics frameworks accomplish their magic.

The devil is in the detail, as they say, so in the next few pages we're going to take a quick look at how to produce some fundamental chart types – 1. Bar Chart, 2. Scatter/Bubble Plot, 3. Histogram, 4. Boxplot, 5. Series/Line Plot – using SAS, R and Python to illustrate some differences in library requirements, data sources and prep, grammar/syntax, etc.

Because my R and Python knowledge stems largely from books and online courses, to be fair to those languages I've tapped into R/Python practitioner Dario Radečić's benchmark code from Matplotlib vs. ggplot2: Which to Choose for 2020 and Beyond? for examples comparing Matplotlib and ggplot2, making some changes to the code to get it running using my datasources (the SASHELP CARS and AIR sample datasets – see Appendix for info) and on my laptop. I then added my own SAS ODS graphics versions of the code for comparison.

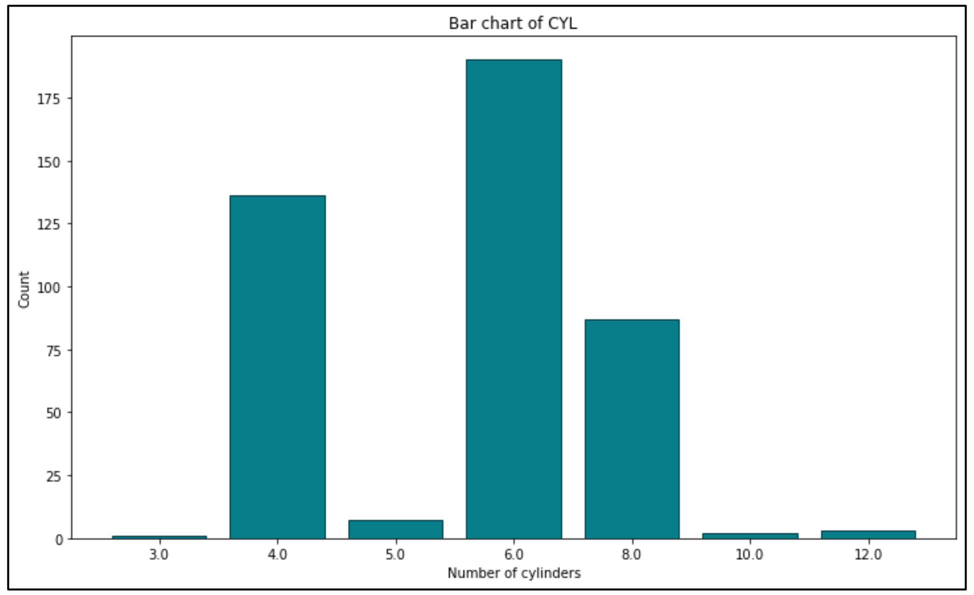
Now, let's take a look at how things compare!

HELLO GRAPHICS WORLD – BAR CHART



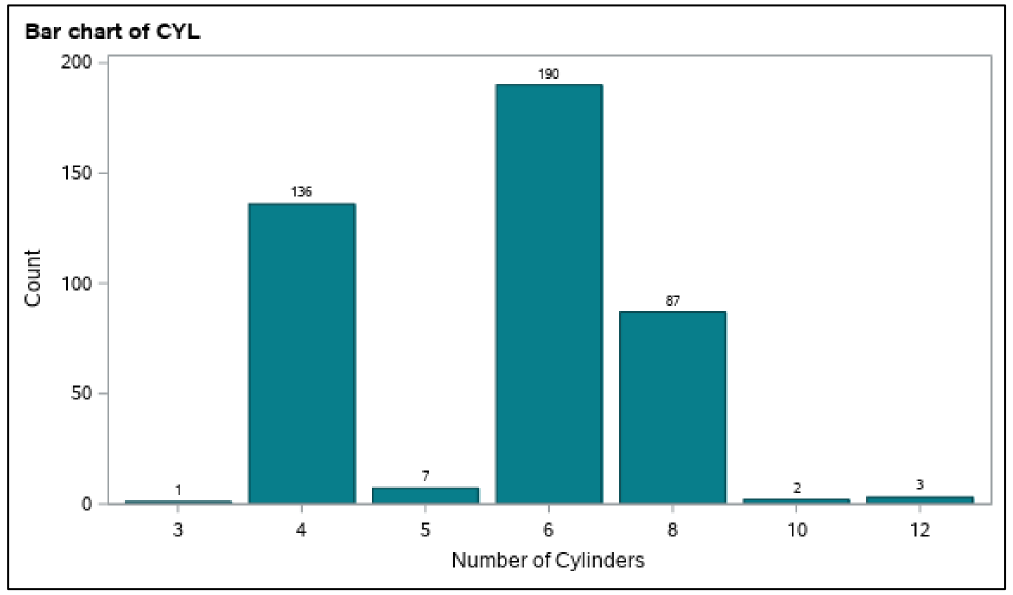
```
library(ggplot2, plyr)
options(repr.plot.width=7.5, repr.plot.height=4.375)
df <-
read.csv('https://raw.githubusercontent.com/tedconway/sasgf/master/cars.csv')
df$Cylinders <- factor(df$Cylinders)
ggplot(df, aes(x=Cylinders)) +
  geom_bar(fill='#087E8B', color='#02454d') +
  ggtitle('Bar chart of CYL') +
  xlab('Number of cylinders') + ylab('Count')
```

- Plus signs ("+") used to separate various chart components
- Custom colors specified as hex RGB constants with a # prefix



```
import pandas as pd, matplotlib.pyplot as plt, matplotlib.dates as mdates
df=pd.read_csv('https://raw.githubusercontent.com/tedconway/sasgf/master/car
s.csv')
bar_counts = df['Cylinders'].value_counts().sort_index()
bar_x = df['Cylinders'].value_counts().sort_index().index
bar_height = df['Cylinders'].value_counts().sort_index().values
plt.figure(figsize=(12, 7))
plt.bar(x=bar_x.astype(str), height=bar_height, color='#087E8B', ec='#02454d')
plt.title('Bar chart of CYL')
plt.xlabel('Number of cylinders')
plt.ylabel('Count');
```

- Additional statements needed to summarize counts & resequence values/labels to display the x-axis ticks in the desired order

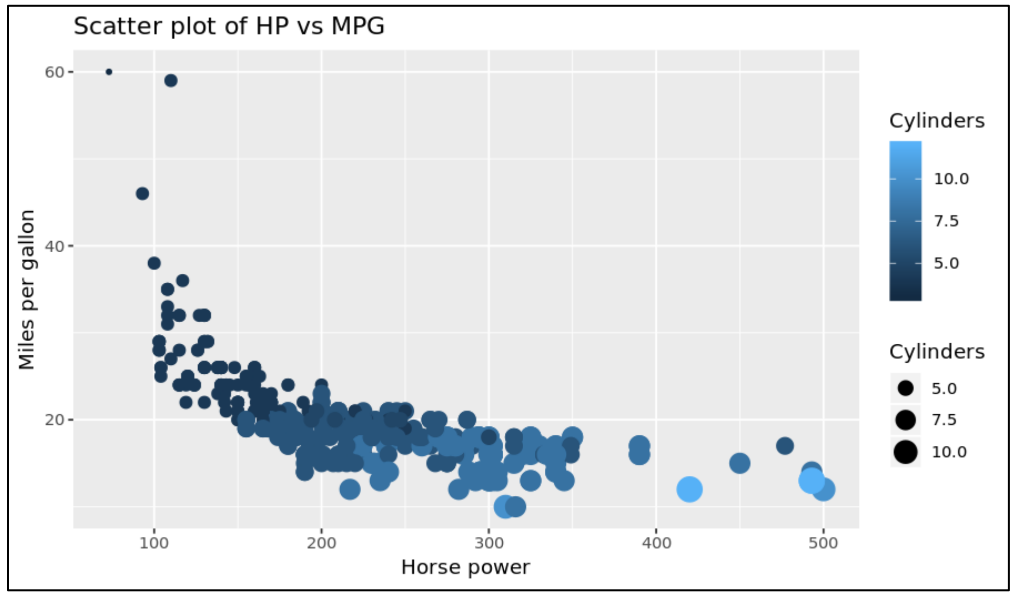


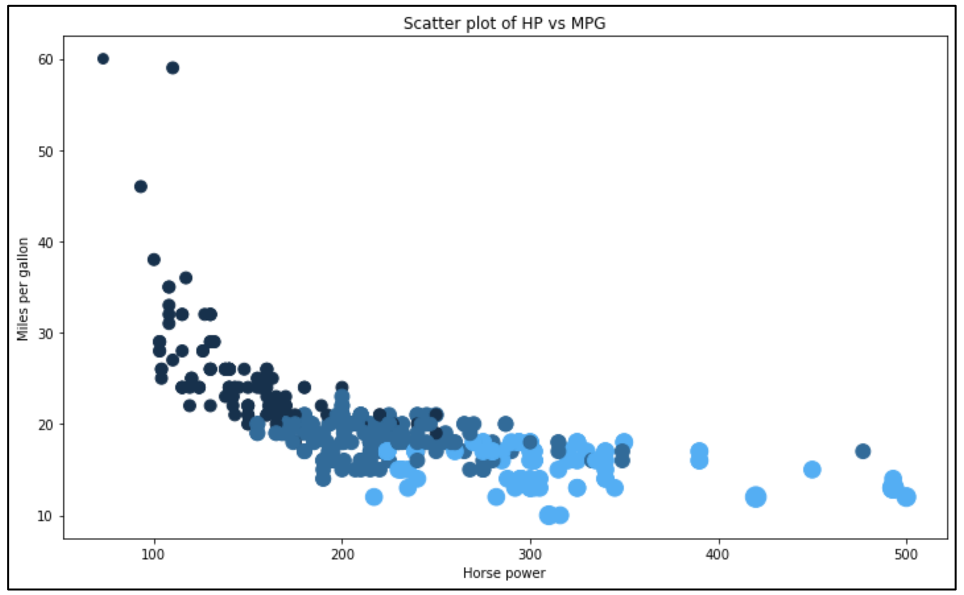
```
ods graphics / height=4.375in width=7.5in noborder antialias;
proc sgplot data=sashelp.cars;
title justify=left 'Bar chart of CYL';
vbar cylinders / fillattrs=(color=cX087E8B) outlineattrs=(color=cx02454d)
datalabel;
label cylinders='Number of Cylinders'; yaxis label='Count';
```

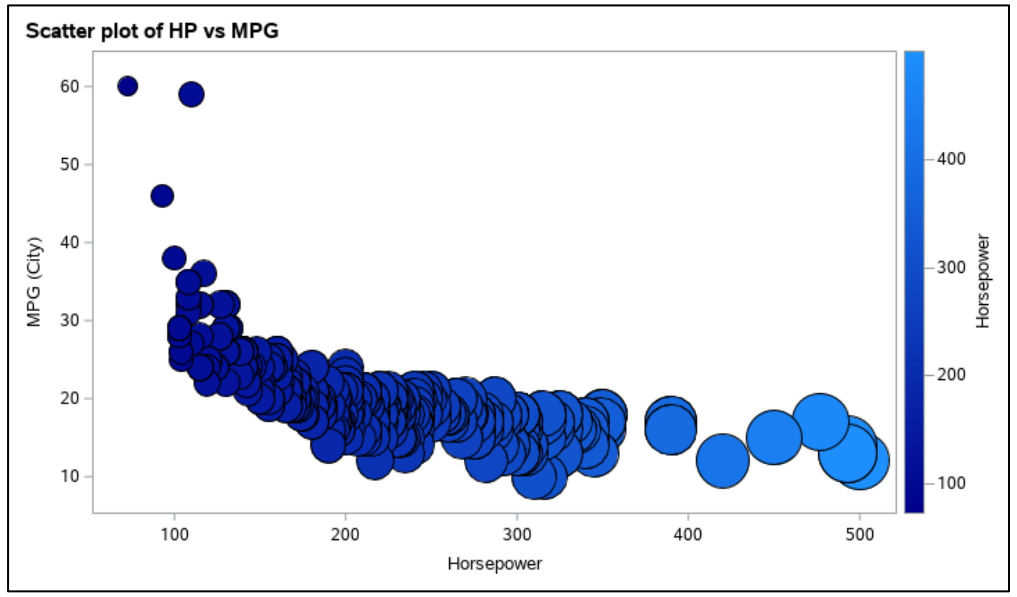
- Bars labeled with the addition of a simple keyword (DATALABEL)
- Like R/Python examples, custom width/height specified

BAR CHART

HELLO GRAPHICS WORLD – SCATTER/BUBBLE PLOT







```
library(ggplot2, plyr)
options(repr.plot.width=7.5, repr.plot.height=4.375)
df <-
read.csv('https://raw.githubusercontent.com/tedconway/sasgf/master/cars.csv')
ggplot(df, aes(x=Horsepower, y=MPG_City)) +
  geom_point(aes(size=Cylinders, color=Cylinders)) +
  ggtitle('Scatter plot of HP vs MPG') +
  xlab('Horse power') + ylab('Miles per gallon')
```

- Marker size and color based on number of cylinders
- Automatically generates legends for both colors and bubble sizes

```
import pandas as pd, matplotlib.pyplot as plt, matplotlib.dates as mdates
df=pd.read_csv('https://raw.githubusercontent.com/tedconway/sasgf/master/car
s.csv')
colors = []
for val in df['Cylinders']:
    if val <= 4: colors.append('#17314c')
    elif val <= 6: colors.append('#326b99')
    else: colors.append('#54aef3')
plt.figure(figsize=(12, 7))
plt.scatter(x=df['Horsepower'], y=df['MPG_City'], s=df['Cylinders'] * 20,
c=colors)
plt.title('Scatter plot of HP vs MPG')
plt.xlabel('Horse power')
plt.ylabel('Miles per gallon');
```

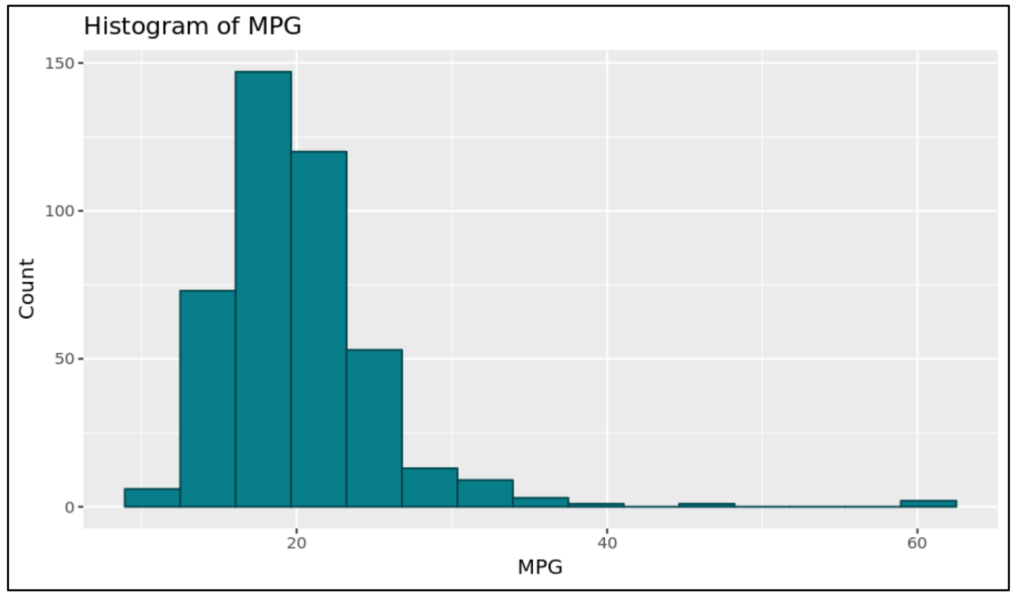
- Need to specify hex RGB values in a Python list to get color range
- No legends are automatically generated

```
ods graphics / height=4.375in width=7.5in noborder antialias;
proc sgplot data=sashelp.cars;
title justify=left 'Scatter plot of HP vs MPG';
bubble x=horsepower y=mpg_city size=horsepower /
       colorresponse=horsepower colormodel=(darkblue dodgerblue);
```

- Generates range of colors automatically from beginning/end values
- Creates a legend for colors, but not for bubble sizes

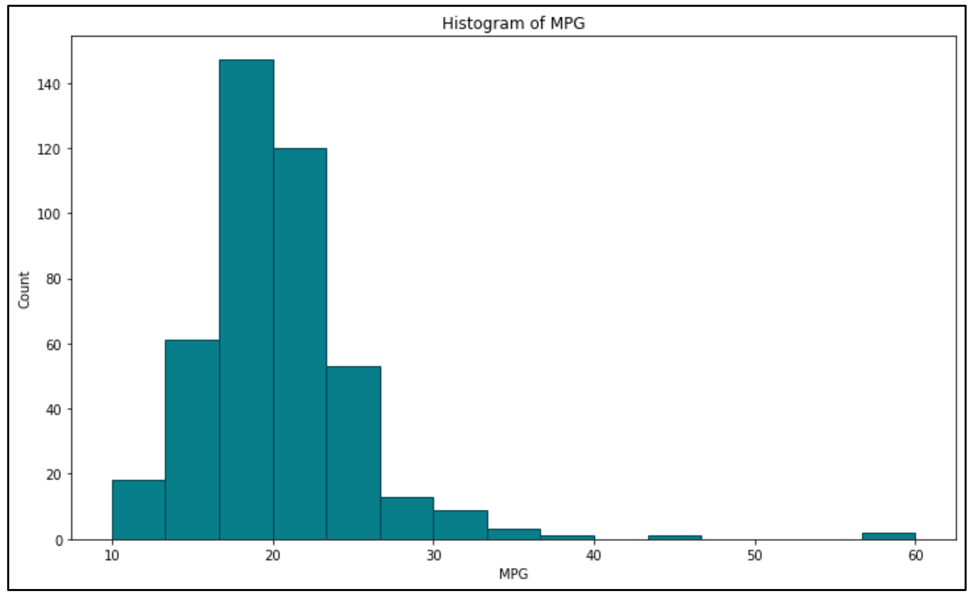
SCATTER/BUBBLE PLOT

HELLO GRAPHICS WORLD – HISTOGRAM



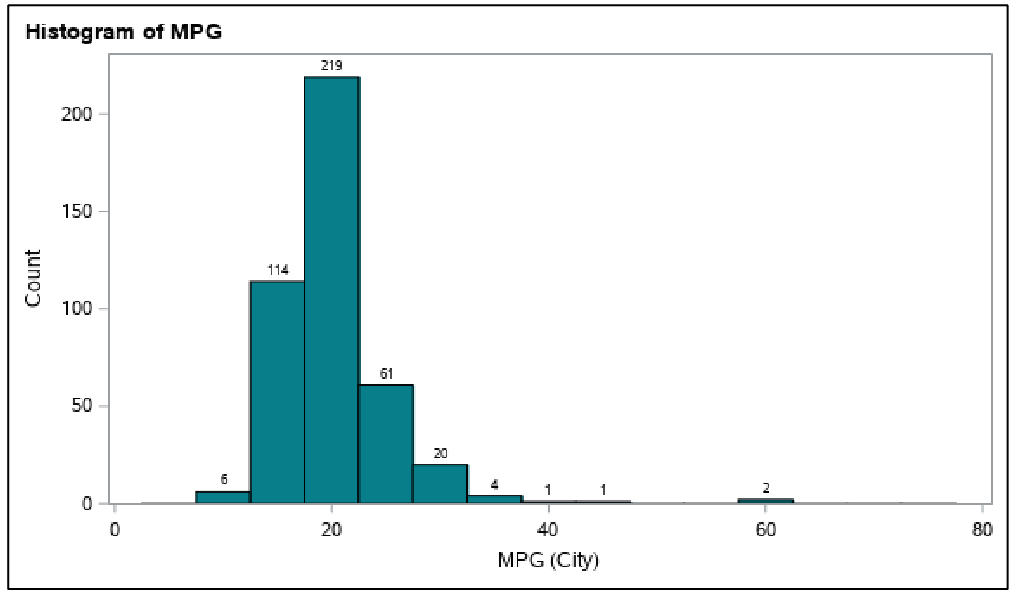
```
library(ggplot2, plyr)
options(repr.plot.width=7.5, repr.plot.height=4.375)
df <-
read.csv('https://raw.githubusercontent.com/tedconway/sasgf/master/cars.csv')
ggplot(df, aes(x=MPG_City)) +
  geom_histogram(bins=15, fill='#087E8B', color='#02454d') +
  ggtitle('Histogram of MPG') + xlab('MPG') + ylab('Count')
```

- Code needed to generate a 15-bin histogram is straightforward



```
import pandas as pd, matplotlib.pyplot as plt, matplotlib.dates as mdates
df=pd.read_csv('https://raw.githubusercontent.com/tedconway/sasgf/master/car
s.csv')
plt.figure(figsize=(12, 7))
plt.hist(df['MPG_City'], bins=15, color='#087E8B', ec='#02454d')
plt.title('Histogram of MPG')
plt.xlabel('MPG')
plt.ylabel('Count');
```

- Matplotlib also shines when it comes to histograms

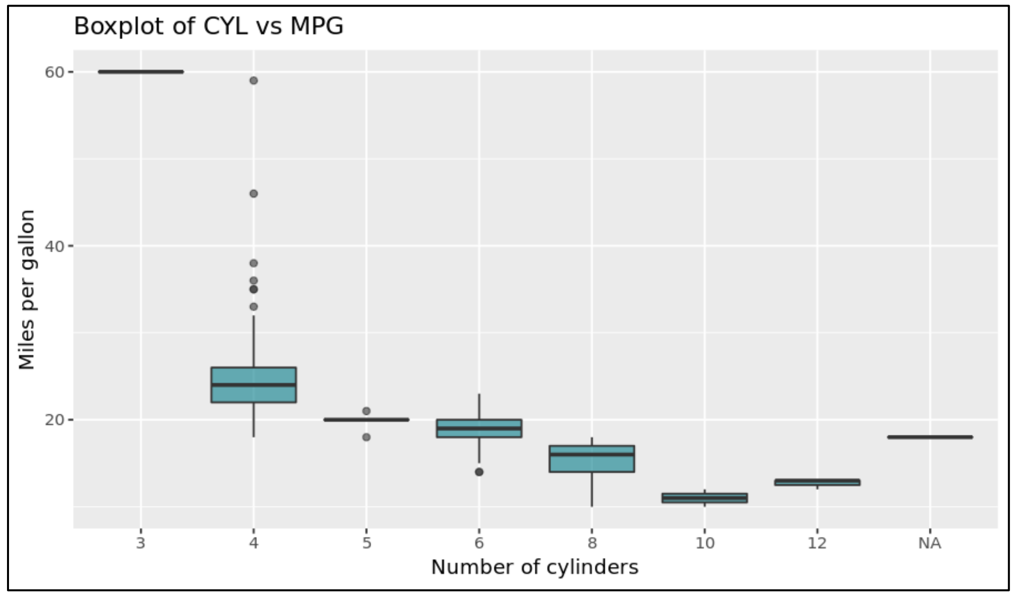


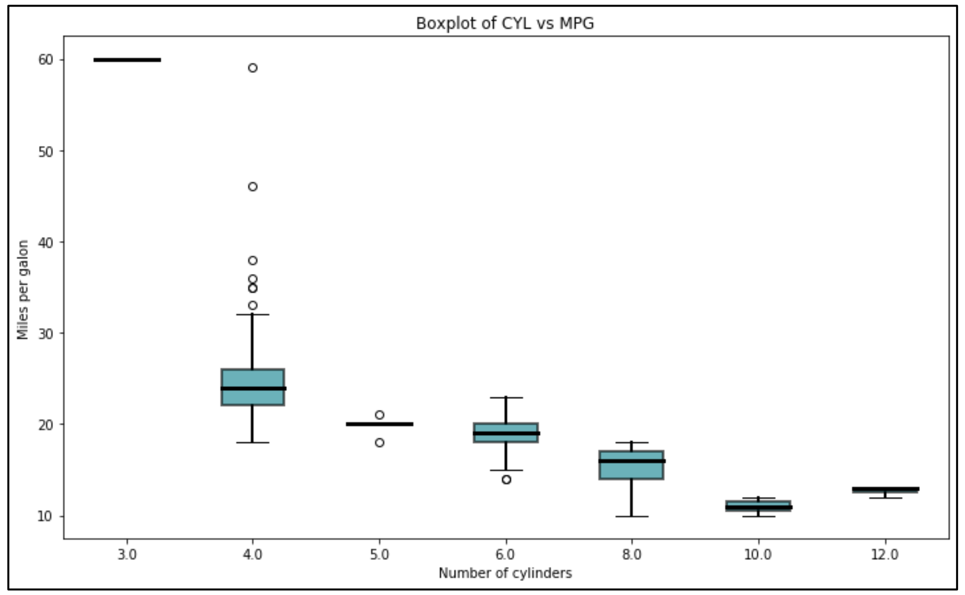
```
ods graphics / height=4.375in width=7.5in noborder antialias;
proc sgplot data=sashelp.cars;
title justify=left 'Histogram of MPG';
histogram mpg_city / nbins=15 scale=count fillattrs=(color=cX087E8B)
datalabel=count;
```

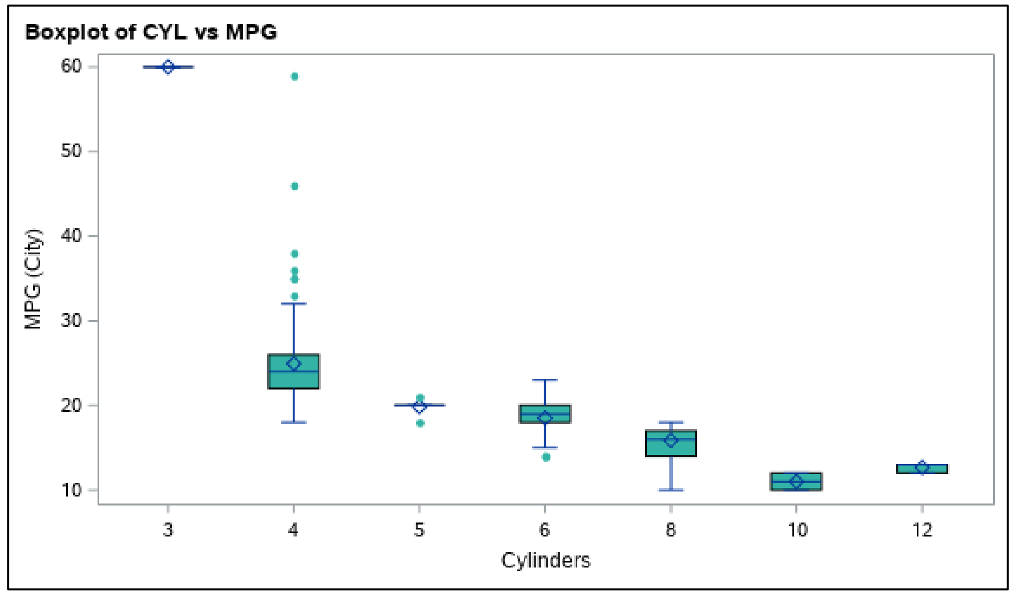
- SAS ODS Graphics also handles histograms with aplomb
- DATALABEL option used to add counts to bars

HISTOGRAM

HELLO GRAPHICS WORLD – BOXPLOT







```
library(ggplot2, plyr)
options(repr.plot.width=7.5, repr.plot.height=4.375)
df <-
read.csv('https://raw.githubusercontent.com/tedconway/sasgf/master/cars.csv')
ggplot(df, aes(x=as.factor(Cylinders), y=MPG_City)) +
  geom_boxplot(fill='#087E8B', alpha=0.6) +
  ggtitle('Boxplot of CYL vs MPG') +
  xlab('Number of cylinders') + ylab('Miles per gallon')
```

- Alpha parameter makes more transparent turquoise, so lines won't be as obscured

```
import pandas as pd, matplotlib.pyplot as plt, matplotlib.dates as mdates
df=pd.read_csv('https://raw.githubusercontent.com/tedconway/sasgf/master/car
s.csv')
dfCylMPG = []
unqCyl=df[df.Cylinders.notnull()]['Cylinders'].unique().tolist()
unqCyl.sort()
for x in unqCyl:
  dfCylMPG.append(df[df['Cylinders'] == x]['MPG_City'].tolist())
fig = plt.figure(1, figsize=(12, 7))
ax = fig.add_subplot(111)
bp = ax.boxplot(dfCylMPG, patch_artist=True)
for box in bp['boxes']:
    box.set(facecolor='#087E8B', alpha=0.6, linewidth=2)
for whisker in bp['whiskers']:
    whisker.set(linewidth=2)
for median in bp['medians']:
    median.set(color='black', linewidth=3)
ax.set_title('Boxplot of CYL vs MPG')
ax.set_xlabel('Number of cylinders')
ax.set_ylabel('Miles per galon')
ax.set_xticklabels(unqCyl)
```

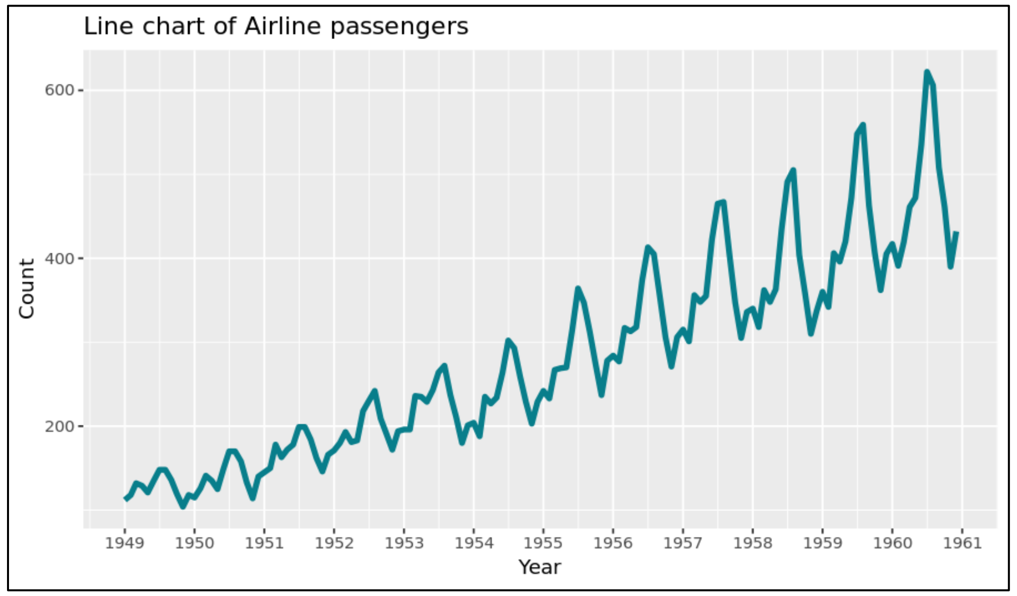
- Trying to make Matplotlib match R styling is surprising involved

```
ods graphics / height=4.375in width=7.5in noborder antialias;
proc sgplot data=sashelp.cars;
title justify=left 'Boxplot of CYL vs MPG';
vbox mpg_city / category=cylinders fillattrs=(color=cX33b3a6)
                outlierattrs=(color=cX33b3a6 symbol=circlefilled);
```

- PROC SGPLOT doesn't support an alpha value for boxplot fill color, so an RGB value for a lighter shade of turquoise was specified

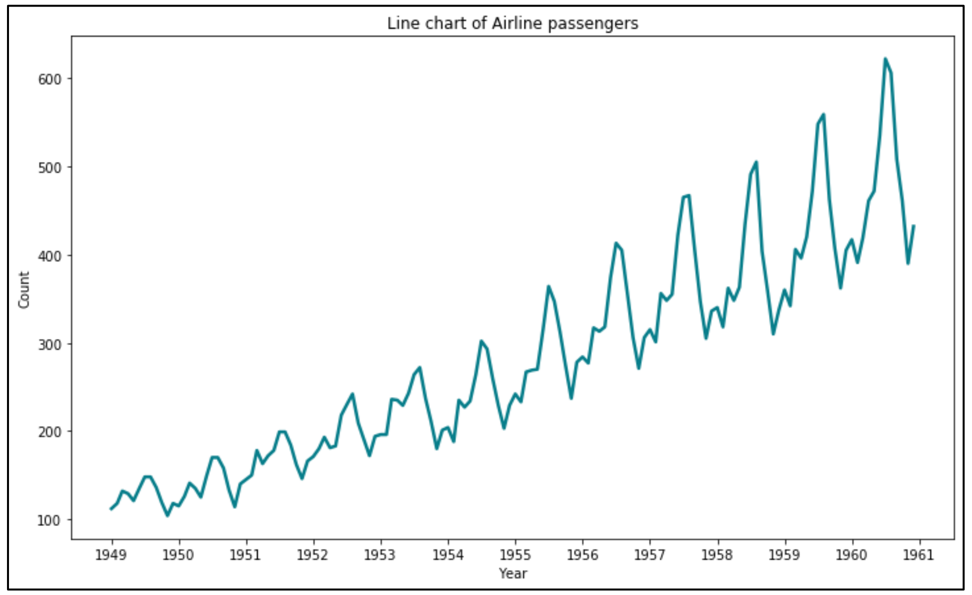
BOXPLOT

HELLO GRAPHICS WORLD – SERIES/LINE PLOT



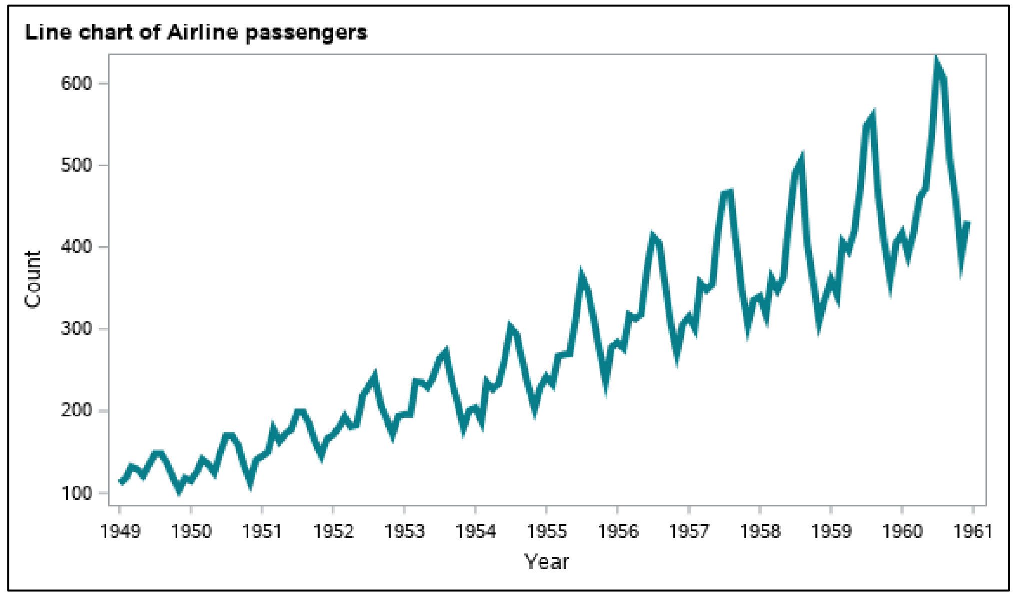
```
library(ggplot2, plyr)
options(repr.plot.width=7.5, repr.plot.height=4.375)
df <-
read.csv('https://raw.githubusercontent.com/tedconway/sasgf/master/air.csv')
df$MONTH <- as.Date(df$DATE)
ggplot(df, aes(x=MONTH, y=AIR)) +
     geom_line(size=1.5, color='#087E8B') +
     scale_x_date(date_breaks='1 year', date_labels='%Y') +
     ggtitle('Line chart of Airline passengers') +
     xlab('Year') + ylab('Count')
```

- Line thickness increased from default using SIZE parameter
- Formatting features used to display 4-digit year values for x-axis



```
import pandas as pd, matplotlib.pyplot as plt, matplotlib.dates as mdates
df=pd.read_csv('https://raw.githubusercontent.com/tedconway/sasgf/master/air.
csv')
df['DATE'] = df['DATE'].apply(lambda x: pd.to_datetime(x))
fig = plt.figure(1, figsize=(12, 7))
ax = fig.add_subplot(111)
line = ax.plot(df['DATE'], df['AIR'], lw=2.5, color='#087E8B')
formatter = mdates.DateFormatter('%Y')
ax.xaxis.set_major_formatter(formatter)
locator = mdates.YearLocator()
ax.xaxis.set_major_locator(locator)
ax.set_title('Line chart of Airline passengers')
ax.set_xlabel('Year')
ax.set_ylabel('Count');
```

- Matplotlib also has to jump through a few hoops to produce x-axis ticks with each year

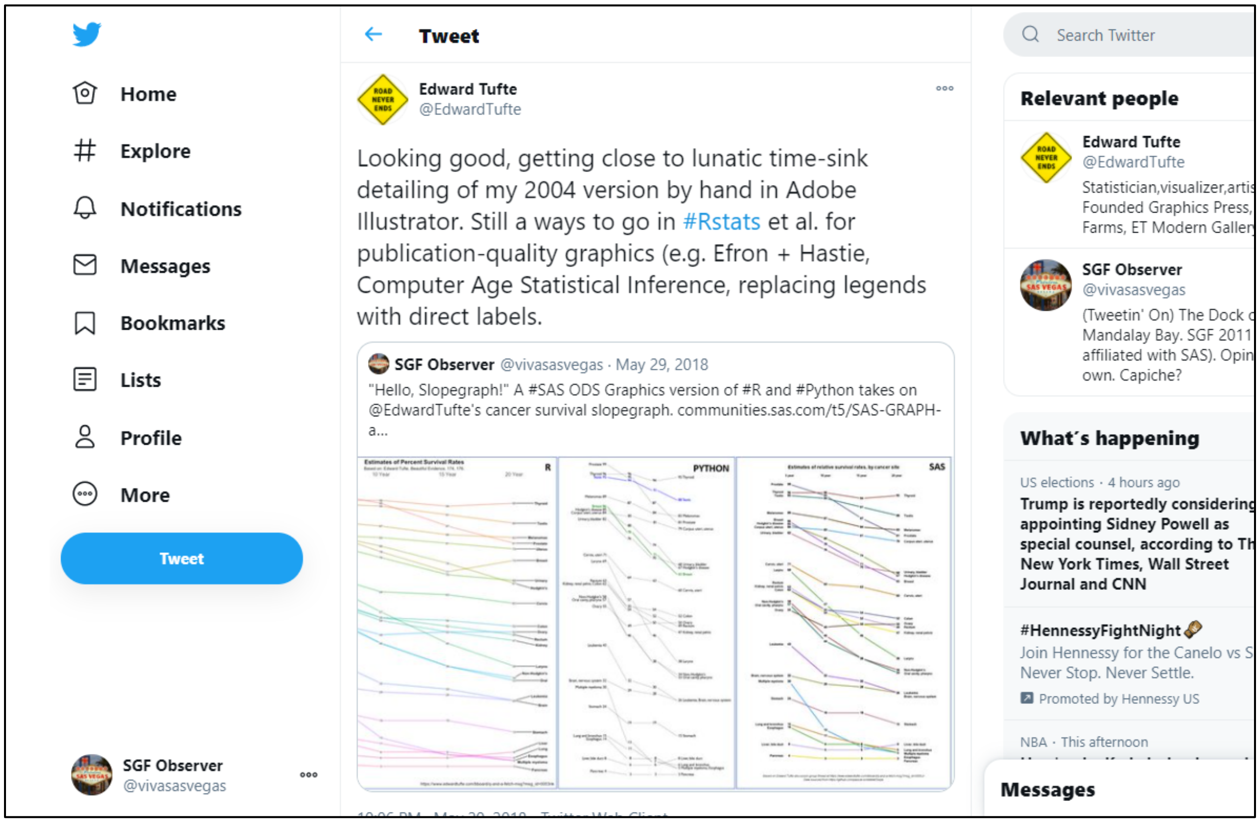


```
ods graphics / height=4.375in width=7.5in noborder antialias;
proc sgplot data=sashelp.air;
title justify=left 'Line chart of Airline passengers';
series x=date y=air / lineattrs=(thickness=3pt color=cX087E8B);
label date='Year' air='Count';
```

- SAS ODS Graphics generates desired x-ticks by default

SERIES/LINE PLOT

TRADEOFFS



Edward Tufte
@EdwardTufte

Looking good, getting close to lunatic time-sink detailing of my 2004 version by hand in Adobe Illustrator. Still a ways to go in #Rstats et al. for publication-quality graphics (e.g. Efron + Hastie, Computer Age Statistical Inference, replacing legends with direct labels.

SGF Observer @vivasasvegas · May 29, 2018
"Hello, Slopegraph!" A #SAS ODS Graphics version of #R and #Python takes on @EdwardTufte's cancer survival slopegraph. communities.sas.com/t5/SAS-GRAPH-a...

10:06 PM · May 30, 2019 · Twitter Web Client

Relevant people

Edward Tufte
@EdwardTufte
Statistician,visualizer,artis
Founded Graphics Press,
Farms, ET Modern Galler

SGF Observer
@vivasasvegas
(Tweetin' On) The Dock c
Mandalay Bay. SGF 2011
affiliated with SAS). Opin
own. Capiche?

What's happening

US elections · 4 hours ago
Trump is reportedly considering appointing Sidney Powell as special counsel, according to Th New York Times, Wall Street Journal and CNN

#HennessyFightNight
Join Hennessy for the Canelo vs S
Never Stop. Never Settle.
Promoted by Hennessy US

NBA · This afternoon

Messages

Even Edward Tufte – the "Father of Data Visualization" – is intrigued by the time-savings possible with SAS/R/Python data visualization frameworks, while cautioning that some tradeoffs are involved.

Indeed, there are a number of factors to balance when considering which programming language and graphics framework to use for data visualization, as well as which features to use within a given framework. This includes:

- ✓ Time, effort and cost
- ✓ Presentation quality and "Wow!" factor
- ✓ Complexity, maintenance, and auditability
- ✓ Interactivity
- ✓ Target audience and medium

While custom charts produced using the latest-and-greatest graphics libraries may indeed dazzle, the additional complexity may result in errors and increased maintenance costs later.

Similarly, while some may delight in exploring interactive visualizations all day in search of "aha!" moments, such is not always going to be the case for the recipients of FDA, FIDC, and academic paper submissions, who are likely to also want you to get straight to your points with a set of nicely-prepared static charts that pre-anticipate questions and save them time.

Not to stifle anyone's creativity, but sometimes you have to consider your audience – and your budget!

CONCLUSION

So, what should you use for graphics?

Well, the answer to that depends on the situation of course, but hopefully we've at least dispelled the notion that SAS can't create graphics as easily as R and Python.

Indeed, with SAS ODS Graphics and Graph Template Language (GTL), you'll find that there's often no need to leave the friendly confines of Base SAS for your charting needs!

EASY IS EASY • HARD IS POSSIBLE • ELIMINATES COSTLY CONTEXT SWITCHING • APPEALING, TERSE SYNTAX • RICH FEATURE SET • WEALTH OF SINGLE AND MULTIPLE-PANEL CHART TYPES • AUTOMATION • REPRODUCIBLE, REPEATABLE RESULTS • PRECISE POSITIONING, SIZING, AND FORMATTING • QUALITY DOCUMENTATION • BUILT FOR DATA SCIENCE • SCALES • MINIMAL DEPENDENCY WORRIES • CONSISTENT LOOK-AND-FEEL • FULL SAS LANGUAGE INTEGRATION • SUPPORT COMMUNITIES • EASY TO WALKTHROUGH, AUDIT, MAINTAIN, ENHANCE

REFERENCES AND RECOMMENDED READING

Matplotlib. "Matplotlib: Visualization with Python." Available at https://matplotlib.org/

SAS. "SAS ODS Graphics: Procedures Guide." Available at http://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatproc/titlepage.htm

SAS. "SAS Graph Template Language: Reference." Available at http://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatgraph/titlepage.htm

Tidyverse. "ggplot2." Available at https://ggplot2.tidyverse.org/

ACKNOWLEDGMENTS

Thanks to Dario Radečić, whose benchmark code in Matplotlib vs. ggplot2: Which to Choose for 2020 and Beyond? (Medium) was tapped to provide the Matplotlib and ggplot2 fundamental chart examples for comparison to SAS ODS Graphics. Any errors made in translation are mine!

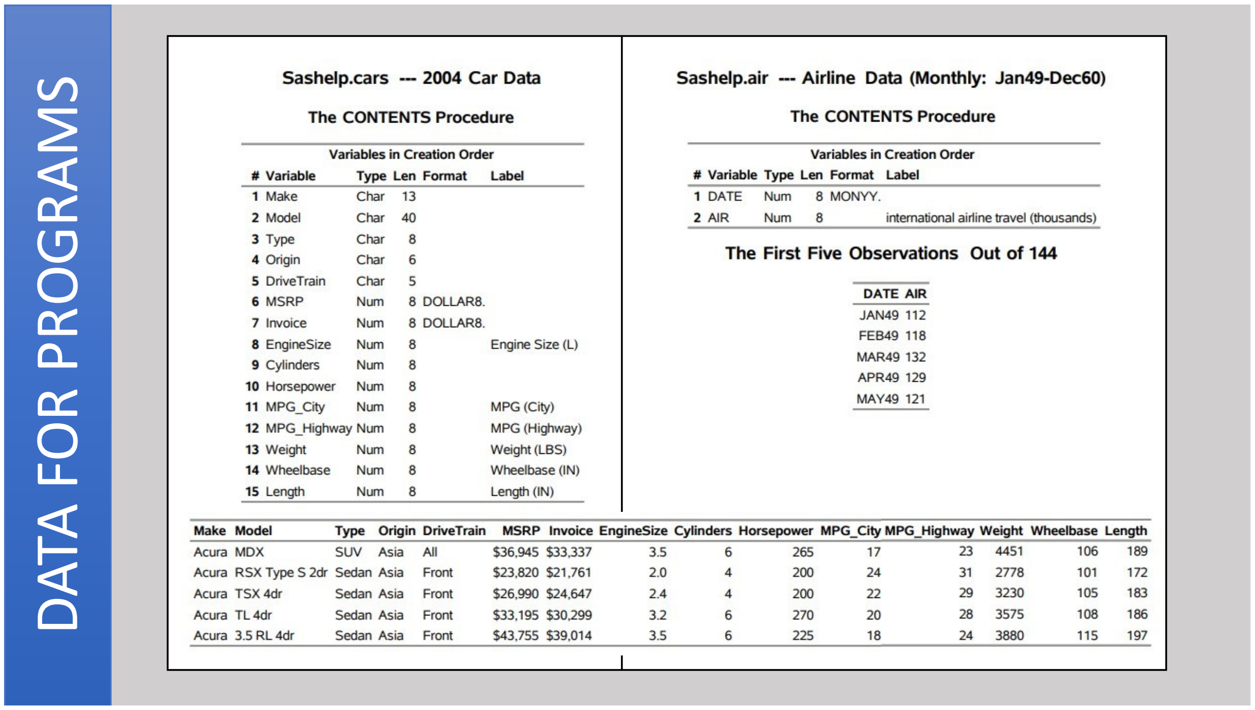
CONTACT INFORMATION

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

Ted Conway
ted.j.conway@gmail.com
@vivasasvegas (Twitter)

APPENDIX – DATA USED IN PROGRAMS

The SAS code used the **CARS** and **AIR** sample SAS datasets provided with SAS in the SASHELP library.


DATA FOR PROGRAMS

Sashelp.cars --- 2004 Car Data

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Format	Label
1	Make	Char	13		
2	Model	Char	40		
3	Type	Char	8		
4	Origin	Char	6		
5	DriveTrain	Char	5		
6	MSRP	Num	8	DOLLAR8.	
7	Invoice	Num	8	DOLLAR8.	
8	EngineSize	Num	8		Engine Size (L)
9	Cylinders	Num	8		
10	Horsepower	Num	8		
11	MPG_City	Num	8		MPG (City)
12	MPG_Highway	Num	8		MPG (Highway)
13	Weight	Num	8		Weight (LBS)
14	Wheelbase	Num	8		Wheelbase (IN)
15	Length	Num	8		Length (IN)

Sashelp.air --- Airline Data (Monthly: Jan49-Dec60)

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Format	Label
1	DATE	Num	8	MONYY.	
2	AIR	Num	8		international airline travel (thousands)

The First Five Observations Out of 144

DATE	AIR
JAN49	112
FEB49	118
MAR49	132
APR49	129
MAY49	121

Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase	Length
Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	265	17	23	4451	106	189
Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2.0	4	200	24	31	2778	101	172
Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200	22	29	3230	105	183
Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270	20	28	3575	108	186
Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225	18	24	3880	115	197

The R and Python code used ,csv versions of the same data, created using PROC EXPORT.

```
data air;
set sashelp.air;
format date yymmddd10.;
proc export data=air dbms=csv outfile="…/air.csv" replace;
proc export data=sashelp.cars dbms=csv outfile="…/cars.csv" replace;
```