

Toe to Toe: Comparing ODS LAYOUT and the ODS Report Writing Interface

Daniel Kummer, SAS Institute Inc., Cary, NC

ABSTRACT

Two new production features offered in the Output Delivery System (ODS) in SAS[®] 9.4 are ODS LAYOUT and the ODS Report Writing Interface. This one-two punch gives you power and flexibility in structuring your SAS[®] output. What are the strengths for each? How do they differ? How do they interact? This paper highlights the similarities and differences between the two and illustrates the advantages of using them together. Why go twelve rounds? Make your report a knockout with ODS LAYOUT and the Report Writing Interface.

INTRODUCTION

Reporting in general, and arranging SAS output on a page in particular, has always been challenging and somewhat limited within SAS. For the longest time you could display your output in one column only or use PROC GSLIDE/GREPLAY or PROC DOCUMENT to somehow arrange it. Even the addition of the COLUMNS= option for the ODS PDF statement left the user with limited control about the placement of output. With the introduction of ODS LAYOUT and the ODS Report Writing Interface (RWI), these challenges and limitations are now a thing of the past.

In short, ODS LAYOUT provides a set of new ODS statements to arrange output objects that come from various sources (for example SAS procedures) with a never before experienced flexibility and control. The ODS Report Writing Interface lives within the DATA step, and displays individual, computed, or aggregated values from an input data set with the same flexibility and control as ODS LAYOUT allows it for procedure output. The next few pages first describe the concept behind each feature, followed by examples that visualize what all of it actually means.

Both ODS LAYOUT and the ODS Report Writing Interface are independent features that can be used stand-alone. Considering the flexibility and control they offer, there are situations where it is useful to use them as a partner-in-crime so to speak. By first understanding the different concepts behind each feature, you will get a clearer understanding of the possibilities each of them has to offer. While advancing through this paper, it will get pretty obvious where each of these features can be useful in creating reports. Both features support the HTML and the PRINTER ODS output destinations for all output that is generated. The only exception is the use of an absolute layout. This type of layout is supported only for the PRINTER destinations.

Within the context of this paper, the term output objects refers to output from SAS procedures (for example graphs and tables), the DATA step, and also to plain text and images.

All SAS code that generates the example reports is attached at the end of this paper. As a common theme, a fictitious real estate company doing business in the Pacific region of the United States is used. The data for all examples is randomly generated, so all examples provided can be reproduced by simply submitting the attached code within SAS. Notice that all images used in the example reports are not provided and will be missing when executing the code.

ODS LAYOUT

As mentioned before, ODS LAYOUT consists of a set of new ODS statements: ODS LAYOUT and ODS REGION. These statements do not produce any output by themselves. They merely define a grid or area in which the output objects are placed. Following ODS standards, those statements wrap around the SAS code that produces the various output objects.

There are two types of layouts available within ODS LAYOUT: the absolute layout and the gridded layout. But in general, starting a layout (no matter the type) with the ODS LAYOUT statement creates what we call a layout container. This container defines the space where the different kinds of output objects can be arranged in. The ODS LAYOUT statement controls the type, size, and placement of the container, but also the behavior within it. Depending on the type of layout, this container can have a fixed size and position, or it can be sized and positioned dynamically.

Individual region containers are specified inside this layout container. These region containers are controlled with the ODS REGION statement. Again, depending on the type of layout, these region containers can have a fixed size and position or they can be sized and positioned dynamically.

The actual output objects are displayed inside the individual region containers. A single region can display one-to-many output objects.

The code structure of a layout follows an easy pattern.

- Start a layout with an ODS LAYOUT statement.
- Define a region with an ODS REGION statement, followed by the code to produce the output object or objects that are supposed to be displayed in this region. This step is repeated as often as needed to populate the report.
- Close the layout with the ODS LAYOUT END statement.

The following code snippet is an example of the basic structure of the program. It also indicates the wrapping of the ODS LAYOUT statements around the SAS code.

```
ods pdf file="...";
ods layout ...;
ods region ...;

    SAS code to produce output objects

ods region ...;

    SAS code to produce output objects

ods layout end;
ods pdf close;
```

To provide near endless customization, layouts can be nested, too. This means that a new layout grid can be defined within an existing region container of a parent layout. The scope of this nested layout is limited in size to the region container in which it is defined. Also, the user is not limited to a single layout type. The two types of layouts can coexist in the same report either by following each other or by being nested (for example a gridded layout can contain an absolute one and vice versa).

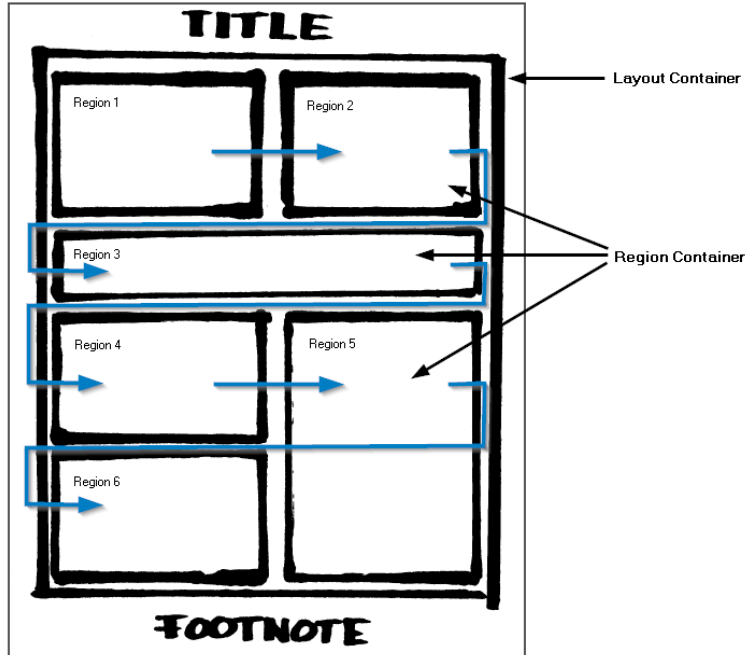
GRIDDED LAYOUT

A gridded layout arranges your output objects in a very dynamic fashion and reflects a two-dimensional grid structure similar to a spreadsheet or table. The regions of a gridded layout are arranged in columns and rows. Every single item in this grid can be customized: From column and row spanning, to the vertical and horizontal space between the regions, but also the size and position of the regions themselves. The big strength of a gridded layout is the automatic alignment and sizing of the respective regions. Regions in a gridded layout will always be centered within the layout container and are by default just as high and wide as needed to display the output object or objects.

The integrity of the grid is always honored. Meaning, that if the report has multiple rows, then the widest region in a column controls the width of the whole column. The same concept applies to the height of a row.

Another feature that makes a gridded layout very dynamic is the possibility to automatically advance to new regions based on certain criteria like BY-groups among others. For example, each table or graph result from a BY-group is displayed in its own region without specifically using an ODS REGION statement for each BY-group.

The layout grid is generally populated in a row wise direction (Display 1). All regions in the first row are populated first before moving on to the next row. This pattern is repeated until the last region is reached. As a single gridded layout can span multiple pages, there really is no restriction on how many regions are specified or dynamically generated.



Display 1: Concept of the Row Wise Population of a Gridded Layout

All these characteristics allow for a lot of flexibility in a report with ever changing data. The syntax is fairly simple and usually does not need to be altered because the data changes over time. Even if those data changes potentially alter the dimensions of individual output objects by adding new variables or BY-group values. The layout regions will adapt accordingly as the layout grid as well as the regions alignment and sizes are newly calculated each time the program runs. It is always important to remember page constraints in the PRINTER destinations. While an HTML page has no true limitations in its width and height, there is only so much space on a piece of paper. When the dynamically defined grid gets too wide to fit on a page, the result will panel and in most cases, not look pretty at all. In those cases, it is necessary to change the syntax of the gridded layout to adapt to the changes in the data.

The syntax for a gridded layout can be very simple. With just a few extra statements in the code, a report can be upgraded to look much more professional than just one graph on a page. The first example is a bar chart that displays the number of units sold for a set of property types. The procedure code for the bar chart uses the BY statement to display a separate chart for each property group (commercial or residential) in each of the five states in which the company does business. Running just the procedure by itself simply wrapped with the ODS PDF statement generates a 10 page document with one chart on each page. Adding the STARTPAGE=NO option to the ODS PDF statement displays up to three charts on one page instead of just one. Still, all the charts are displayed only in one column.

Using a gridded layout, each separate chart can be displayed in its own region of a defined grid. As there are two charts per state, they could be displayed side-by-side so that each state consumes only one row in the report. The following ODS LAYOUT GRIDDED statement defines such a grid. The COLUMNS= option indicates the number of columns in the grid. The COLUMN_GUTTER= option specifies the horizontal space between the columns. The ADVANCE= option controls for which output type to move to the next region in the grid.

```
ods layout gridded columns=2 column_gutter=0in advance=bygroup;
```

To make the report look more professional, an extra region is defined to create something like a header before displaying all the charts. This region is not dynamically populated. It gets a specific height, spans both columns of the grid, has a blue background color, and all the contents inside the region are right aligned. The only output objects that are displayed inside the region are two lines of text mimicking a company logo.

```
ods region height=1in column_span=2 style={background=&backBlue. just=right};
ods text="^S={background=&backBlue. foreground=white font_size=50pt}ACME";
ods text="^S={background=&backBlue. foreground=white font_size=10pt}Sales Report
Pacific Region";
```

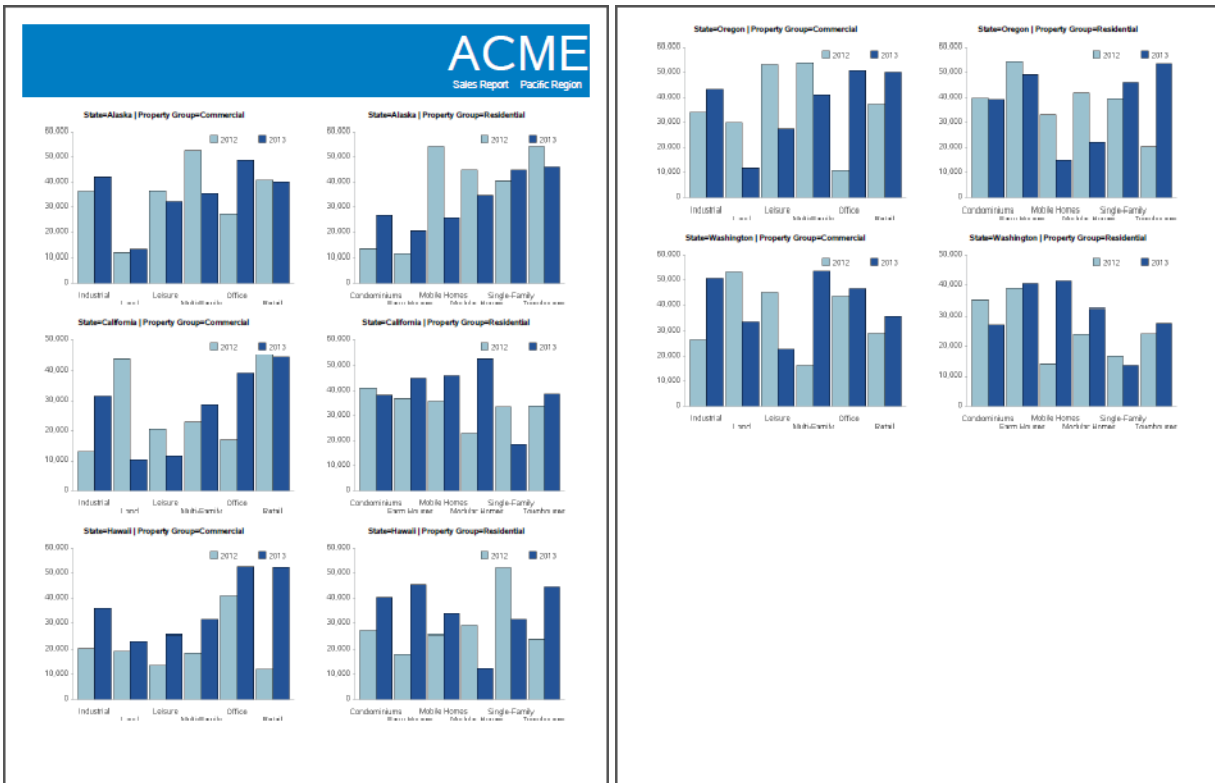
Another ODS REGION statement creates the second region container. As the first region spans two columns, this one is in the first column of the second row. No additional options are specified to maintain maximum flexibility when aligning and sizing the region container. As this region is supposed to display the first chart, all code to generate the charts follows next.

```
ods region;
  goptions ...;
  axis pattern & label statements;
  proc gchart code to generate bar charts;
```

The first chart that is produced is displayed in this region. All other charts that are generated by the procedure are placed in new regions with the same characteristics. Each new region is aligned within the defined grid. As this is all the output that is displayed in this example, an ODS LAYOUT END statement is used to close the gridded layout.

```
ods layout end;
```

Display 2 shows the final report. Also, notice that individual regions can display titles and footnotes. In this case, the BY-line displays the BY-group values for each chart.



Display 2: Result of the Gridded Layout Example

ABSOLUTE LAYOUT

While a gridded layout is very dynamic in populating a report, an absolute layout is more static in nature. Static when talking about the placement of the region containers. An absolute layout is not restricted to columns and rows, but rather follows the concept of layers. The regions are generally defined by an X and Y coordinate and a width and height. When not specifying values for those options, they resolve to their default values (X and Y = 0, width and height = max). The X and Y coordinate's zero position is the upper left corner of the layout container.

Correct use and order of specifying the region position and dimensions are crucial with this type of layout, as regions can overlap each other. Therefore, some type of design phase is beneficial before starting to write code. This can help to avoid resizing one or more regions whenever changing the position and dimension of preceding ones.

While a region in a gridded layout can dynamically adjust to the size of the output object or objects that are displayed, an absolute layout region is empty if it is not sized properly. In addition, a warning is displayed in the log.

The scope of an absolute layout is one single page. However, a new layout can always be started on the next page to continue the report.

The structure of the statements for an absolute layout is similar to the one for a gridded layout. It is usually a lot more code as the regions are not dynamically populated. Hence, each region needs to be addressed specifically. The following absolute layout example is a one page report that compares last year's sales numbers with the ones from previous years (Display 3 below). Multiple output objects are displayed, including a bar chart, pie charts, tables, an image, and text.

The absolute layout is started with a single ODS LAYOUT ABSOLUTE statement. No additional options are used on the statement itself. Hence, the default values for X (0), Y (0), WIDTH (max), and HEIGHT (max) are used. As no titles and footnotes are used the rendered layout container takes up the whole available space that is defined by the page margins.

```
ods layout absolute;
```

Next, each region and its contents are defined. Individual regions are rendered in the order in which they appear in the code. The options used to define a region or the contents displayed inside this region can have an influence on the order it appears in the code. The first two regions generate the company header. The first one generates a blue box (a region container with a blue background color). As this region is supposed to start at the top left corner of the report and be as wide as the whole layout container, no X, Y, and WIDTH options are needed. The default values take care of this. However, the header is supposed to have a specific height of 1 inch. An absolute layout region needs to contain at least one output object. Otherwise, the region is not rendered. In this case, the ODS TEXT statement displays a blank, making sure the region is rendered.

```
ods region height=1in style={background=&backBlue.};  
ods text=" ";
```

The second region does not specify a Y option and starts at the top of the layout container as well. However, as it uses the X option, the region starts 5.5 inches to the right of the left border of the layout container. This region is rendered after the first one and sits on top of it. It uses the same height and background color as the previous region to make it look as one unit. The reason for the second region is to offset the text a little bit from the right border.

```
ods region x=5.5in width=2.5in height=1in style={background=&backBlue.};  
ods text=" ^S={background=&backBlue. foreground=white font_size=50pt}ACME";  
ods text=" ^S={background=&backBlue. foreground=white font_size=10pt}Sales Report  
Pacific Region";
```

The other regions in this layout follow the same pattern. The next region displays text on the left side right under the header.

```
ods region y=1.25in width=4.75in height=1.25in;  
ods text=" ^S={font_size=8pt}Commercial property sales accelerated in 2013, ...";  
...
```

Right below the text, another region with the same width is rendered to display a bar chart.

```
ods region y=2.75in width=4.75in height=2.75in;
  goptions ...;
  axis pattern & label statements;
  proc gchart code to generate bar chart;
```

The next two regions populate the space to the right of the text and bar chart. The first region displays a description for the image that is displayed in the second region right below it. Notice that the values for the X and WIDTH option are the same in both regions to align them in the report. The ODS Report Writing Interface is used to display the image in the second region.

```
ods region x=5in y=1.25in width=3in height=0.18in style={background=&backBlue.};
  ods text="^S={just=center background=&backBlue. foreground=white
font_size=10pt}States of the Pacific Region";

ods region x=5in y=1.6in width=3in;
  data _NULL_;
  dcl odsout dak();
  dak.image(file: "pacificMed.jpg");
run;
```

Looking at the finished report (Display 3), someone could guess that the next region addresses the blue box with the description for the pie chart and table below it. However, when generating the pie chart with outside labels, the result has a rather big area of white space around it. For this reason, the region that contains the pie chart is rendered first, and the other regions are put on top of it to maximize the space for the pie chart.

```
ods region y=5.5in height=2.5in width=3in;
  goptions ...;
  proc gchart code to generate pie chart;

ods region y=5.5in width=8in height=0.18in style={background=&backBlue.};
  ods text="^S={just=center background=&backBlue. foreground=white
font_size=10pt}2013 Numbers for Commercial Properties";

ods region x=3.5in y=5.8in height=2.2in;
  ODS RWI code to generate table;
```

The same logic is applied to the last section of regions. Again, the region for the pie chart is defined first, followed by the region for the description, and finally the one for the table. To show the flexibility in positioning the output objects on the page, the table and the pie chart are displayed on the opposite side compared to the section before.

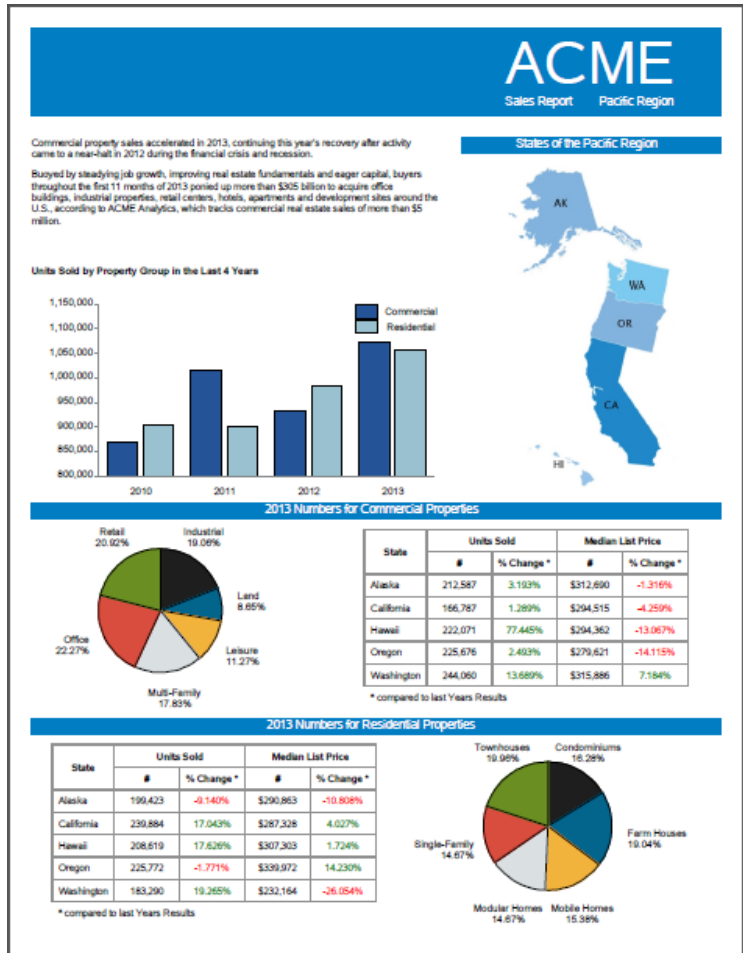
```
ods region x=4in y=8in height=2.5in width=4in;
  goptions ...;
  proc gchart code to generate pie chart;

ods region y=8in width=8in height=0.18in style={background=&backBlue.};
  ods text="^S={just=center background=&backBlue. foreground=white
font_size=10pt}2013 Numbers for Residential Properties";

ods region y=8.3in width =4.25in height=2.2in;
  ODS RWI code to generate table;
```

As all the regions for the report are defined, the ODS LAYOUT END statement ends the absolute layout.

```
ods layout end;
```



Display 3: Result of the Absolute Layout Example

ODS REPORT WRITING INTERFACE

The ODS Report Writing Interface could be described as the DATA step's PUT statement on steroids. It uses the ODSOUT object in combination with a set of methods to create reports within a DATA _NULL_ step. It is designed to display individual, computed, or aggregated values from an input data set in the shape of a table, a gridded layout, or an absolute layout. In addition, titles, footnotes, and page breaks can be controlled within the DATA step.

The syntax for a DATA step using the ODS Report Writing Interface can be very simple. But as soon as an input data set is involved, the complexity increases quite a bit. A basic understanding of the DATA step is beneficial and recommended. The length and complexity of the DATA step highly depends on how sophisticated the final result is supposed to be, and more importantly, on the structure of the data used.

Sometimes it can be easier and faster to use the established procedures like PRINT, TABULATE, and REPORT; however, often those reports need a final modification or more customization than the procedures can provide. These customized reports are the strength of the ODS Report Writing Interface. The combination of all the statements of the DATA step, including conditional processing and by group processing, and the integration of ODS through the methods used in the ODS Report Writing Interface can create reports you previously only dreamed about.

The result of a DATA step using the ODS Report Writing Interface is also considered an output object. Pretty much the same as the result of any procedure is an output object. It can stand by itself like any PROC PRINT for example. It can also be used in combination with ODS LAYOUT to populate regions within a gridded or absolute layout. The table structure that is displayed in the previous example for the absolute layout has been generated with the ODS Report Writing Interface.

The data set that is used to generate the table structure is pre-aggregated and has two observations for each state. The first of those two observations stores the numbers of sold units and the median house price for the year 2012. The second one stores the same information for the year 2013. The resulting table should display only the 2013 numbers, but should also contain an additional column that displays the percentage of change of the respective variable compared to the previous year.

The first few lines of the DATA step are standard statements. The SET statement specifies the data set (pacificSum). A WHERE= option reads only certain rows from the input data set and the END= option flags the last observation of the data set. BY-group processing is used to retain the values of two variables for the 2012 observation of each state and have them available when processing the observation for the year 2013.

```
data _NULL_;
  set pacificSum (where=(propGroup="Commercial")) end=done;
  by state year;
  retain units12 price12;
```

As the result contains only one table for the whole data set, the table structure is started at the beginning of the DATA step and for the first observation only. Without this condition, a new table would be started for each observation. The next two rows in the table act as the header section for the table.

```
if _N_ = 1 then do;
  dcl odsout dak();
  dak.table_start();
  dak.row_start();
  dak.format_cell(data: "State",
                 row_span: 2,
                 vjust: "center",
                 style_attr: "font_weight=bold");
  dak.format_cell(data: "Units Sold",
                 column_span: 2,
                 style_attr: "font_weight=bold");
  dak.format_cell(data: "Median List Price",
                 column_span: 2,
                 style_attr: "font_weight=bold");
  dak.row_end();
  dak.row_start();
  dak.format_cell(data: "#",
                 style_attr: "font_weight=bold",
                 width: ".75in");
  dak.format_cell(data: "% Change *",
                 style_attr: "font_weight=bold",
                 width: ".75in");
  dak.format_cell(data: "#",
                 style_attr: "font_weight=bold",
                 width: ".75in");
  dak.format_cell(data: "% Change *",
                 style_attr: "font_weight=bold",
                 width: ".75in");
  dak.row_end();
end;
```

For the first observation of the BY-groups (a specific state and the year 2012), the only action that is required is to update the values for the variables that need to be retained for later processing in another observation.

```
if first.state and first.year then do;
  units12 = sUnits;
  price12 = mPrice;
end;
```


For the last observation of the BY-groups (a specific state and the year 2013), the percentage of change variables are computed first. Then a table row is displayed populating all five cells of the row with their respective values. Traffic lighting is used for the percentage of change values to put a stronger emphasis on the results.

```
if last.state and last.year then do;
  changeUnits = ( ( sUnits - units12 ) / units12 );
  changePrice = ( ( mPrice - price12 ) / price12 );

  dak.row_start();
  dak.format_cell(data: state, just: "left");
  dak.format_cell(data: sUnits, format: "comma12.");
  if changeUnits GE 0 then dak.format_cell(data: changeUnits,
                                          format: "percentn10.3",
                                          style_attr: "foreground=green");
  else dak.format_cell(data: changeUnits,
                      format: "percentn10.3",
                      style_attr: "foreground=red");
  dak.format_cell(data: mPrice, format: "dollar12.");
  if changePrice GE 0 then dak.format_cell(data: changePrice,
                                          format: "percentn10.3",
                                          style_attr: "foreground=green");
  else dak.format_cell(data: changePrice,
                      format: "percentn10.3",
                      style_attr: "foreground=red");
  dak.row_end();
end;
```

To add a footnote under the table row for the last observation and to end the table, the following section of the code has to be executed last and only for the last observation of the data set.

```
if done then do;
  dak.row_start();
  dak.format_cell(data: "* compared to last Years Results",
                 just: "left",
                 column_span: 5,
                 inhibit: "LBR");
  dak.row_end();
  dak.table_end();
end;
```

End the DATA step with a RUN statement.

```
run;
```

Submitting this DATA step by itself creates the table structure that is pictured in Display 4. It features a defined header section that uses row and column spanning, a body that displays the values for the year 2013 that includes computed columns, and a footer that displays additional text but no table borders.

State	Units Sold		Median List Price	
	#	% Change *	#	% Change *
Alaska	212,587	3.193%	\$312,690	-1.316%
California	166,787	1.289%	\$294,515	-4.259%
Hawaii	222,071	77.445%	\$294,362	-13.067%
Oregon	225,676	2.493%	\$279,621	-14.115%
Washington	244,060	13.689%	\$315,886	7.184%

* compared to last Years Results

Display 4: Table Created with the Above DATA Step Example Using the ODS Report Writing Interface


The biggest strength of the ODS Report Writing Interface is the possibility to display any variable that is available within the DATA step anywhere inside the defined space of a report. Besides presenting the data in tabular form, the ODS Report Writing Interface can also use a gridded or absolute layout. The concept is exactly the same as for ODS LAYOUT. Instead of specifying ODS LAYOUT and ODS REGION statements, the respective methods of the Report Writing Interface are used. The biggest difference between the two is the type of output that is displayed within the regions of the layout. The ODS Report Writing Interface is confined to the information available within the DATA step. It can display only individual, computed, or aggregated values from variables available in the Program Data Vector (PDV). In addition, images can be displayed, too.

Display 5 shows the two page result of an absolute layout generated with the ODS Report Writing Interface. The whole report is data driven. The absolute layout defines the look of a single page. As each observation from the data set creates one page in the PDF document, the look of each page is the same. However, the data displayed on each page depends on the observation that populated the specific page.

Hill House
1328 Elm Street

ACME

Pacific Region



General Information

This finer upper is ready for you. This large plantation house with a low maintenance yard just needs a little TLC. Full Disclosure: A night spent here could turn into an eternity.

Bed Rooms:	1	Bath Rooms:	0
House Size (SqFt):	2,453	Lot Size (Acres):	25
Price:	\$275,643	Year Built:	1823
Property Type:	Single-Family	Style:	Traditional

Building Features

Exterior Finish:	Wood	# Garages:	0	Cooling:	Forced	Inside City Limits:	No
Roof:	Shingles	# Levels:	4	Heating:	Fireplace	Area:	Made Up Area
Floor:	Hardwood	Heated Area (SqFt):	2453	assocFees:	-	County:	Fake
Foundation:	Concrete	Additional Area (SqFt):	-	Deck:	Yes	Listing Number:	000001
Basement:	Yes	Fireplace:	Yes	Screened Porch:	Yes	Listing Area:	666

Room Dimensions

Master Bedroom:	20x20	Kitchen:	15x10	Sun Room:	Deck:
2nd Bedroom:	Family Room:	20x20	Entrance:	Basement:	45x30
3rd Bedroom:	Dining Room:	15x10			
4th Bedroom:	Office:				

School Information

Elementary School:	Voorhes Elementary	Middle School:	Myers Middle	High School:	Kuiper High
--------------------	--------------------	----------------	--------------	--------------	-------------


Contact

Laurie Strode
Office: (866)999-0000 ext. 2 Cell: (866)999-4866 Fax: (866)999-0008
email: laurie@smifsgroves.com www.smifsgroves.com/000001

The Farnsworth House
1800 Fancy Drive

ACME

Pacific Region



General Information

As the perfect family vacation retreat the Farnsworth House is lying in a cross-hair setting and seems to float weightlessly above the ground it occupies. It has become an iconic expression of 20th century modernist architecture.

Bed Rooms:	1	Bath Rooms:	1
House Size (SqFt):	1,554	Lot Size (Acres):	80
Price:	\$2,500,000	Year Built:	1951
Property Type:	Vacation Home	Style:	Modern

Building Features

Exterior Finish:	Steel & Glass	# Garages:	0	Cooling:	None	Inside City Limits:	No
Roof:	Metal	# Levels:	1	Heating:	Fireplace	Area:	Made Up Area
Floor:	Tile	Heated Area (SqFt):	-	assocFees:	-	County:	Fake
Foundation:	Steel Columns	Additional Area (SqFt):	-	Deck:	Yes	Listing Number:	000002
Basement:	No	Fireplace:	Yes	Screened Porch:	No	Listing Area:	535

Room Dimensions

Master Bedroom:	20x10	Kitchen:	20x4	Sun Room:	Deck:	15x40
2nd Bedroom:	Family Room:	30x20	Entrance:	Basement:	20x20	
3rd Bedroom:	Dining Room:					
4th Bedroom:	Office:					

School Information

Elementary School:	Peter Parker Elementary	Middle School:	Bruce Wayne Middle School	High School:	Han Solo High
--------------------	-------------------------	----------------	---------------------------	--------------	---------------

Contact

Wile E. Coyote
Office: (866)999-0000 ext. 1 Cell: (866)999-4861 Fax: (866)999-0008
email: wile.e.coyote@acmeincorp.com www.acmeincorp.com/000002

Display 5: Result of an Absolute Layout Generated with the ODS Report Writing Interface

10

At first glance, the DATA step code to generate the example shown in Display 5 seems long and complicated. But on a second glance, the code is pretty straightforward. Hardly any conditional processing is used, resulting in almost every line of the DATA step being executed for each observation of the input data set.

The fictitious real estate company has a data set that contains detailed information for each active property. The information for each individual property is stored in one observation at a time. The idea is to create a one page report for each active property that displays all the information available.

The DATA step starts with a SET statement that defines the input data set and flags the last observation for conditional processing. Next, one of only two IF statements in the whole DATA step declares the ODSOUT object. The other IF statement is used at the end of the DATA step to add a page break for all observations but the last one.

```
data _NULL_;  
  
    set acmeHouses end=done;  
  
    if _N_ = 1 then do;  
        dcl odsout dak();  
    end;
```

Next, the absolute layout is started. The first few regions of the layout generate a header similar to the ones that were used in the previous examples. An extra addition to this header is the display of values from two variables (name and street1). The concept of overlapping regions is again used to define correct placement of the text within the header. The whole header appears as one unit as the background color of all regions and text displayed matches.

```
dak.layout_absolute();  
  
dak.region(width: "8in",height: "1in", style_attr: "background=&backBlue.");  
dak.format_text(data: "", style_attr: "background=&backBlue.");  
  
dak.region(x: ".5in", y: ".2in", width: "3in", height: "1in");  
dak.format_text(data: name,  
                style_attr: "background=&backBlue.  
                            foreground=white  
                            font_size=20pt");  
dak.format_text(data: street1,  
                style_attr: "background=&backBlue.  
                            foreground=white  
                            font_size=15pt");  
  
dak.region(x: "5.5in", width: "2in", height: "1in");  
dak.format_text(data: "ACME",  
                just: "right",  
                style_attr: "background=&backBlue.  
                            foreground=white  
                            font_size=50pt");  
  
dak.region(x: "5.5in", y: ".7in", width: "2in", height: ".5in");  
dak.format_text(data: "Pacific Region",  
                just: "right",  
                style_attr: "background=&backBlue.  
                            foreground=white  
                            font_size=10pt");
```

The region that follows the header section uses only the left side of the page to display an image.

```
dak.region(y: "1.25in", width: "4in", height: "3in");  
dak.image(file: "&imageLoc." || strip(listNum) || ".jpg");
```

The next region addresses the space to the right side of the image. Using ODS style definitions, the appearance of the first text output is altered to achieve the impression of a section header. An empty line separates the following text from the section header.

```
dak.region(x: "4in", y: "1.25in", width: "4in", height: "3in");
  dak.format_text(data: "General Information",
                 style_attr: "font_size=12pt
                             font_weight=bold
                             width=100%
                             just=c
                             background=&backGray");

  dak.format_text(data: "");
  dak.format_text(data: desc);
```

Inside the same region, a table structure with no borders (frame=void rules=none) is created to display the labels and values of eight variables. The table has four rows that contain four cells. The first cell displays the variable's label and the second cell displays the actual value of the variable. This setup is repeated for cells three and four using a different variable. All other table rows are defined the same way just displaying different variables from the data set.

```
dak.table_start(style_attr:"frame=void rules=none width=4in");
  dak.row_start();
    dak.format_cell(data: vlabel(brooms) || ":",
                   just: "left",
                   style_attr: "font_weight=bold");
    dak.format_cell(data: brooms, just: "left");
    dak.format_cell(data: vlabel(bath) || ":",
                   just: "left",
                   style_attr: "font_weight=bold");
    dak.format_cell(data: bath, just: "left");
  dak.row_end();
  dak.row_start();
    dak.format_cell(data: vlabel(houseSize) || " (SqFt):",
                   just: "left",
                   style_attr: "font_weight=bold");
    dak.format_cell(data: houseSize, just: "left", format: "comma12.");
    dak.format_cell(data: vlabel(lotSize) || " (Acres):",
                   just: "left",
                   style_attr: "font_weight=bold");
    dak.format_cell(data: lotSize, just: "left");
  dak.row_end();
  dak.row_start();
    dak.format_cell(data: vlabel(price) || ":",
                   just: "left",
                   style_attr: "font_weight=bold");
    dak.format_cell(data: price, format: "dollar14.", just: "left");
    dak.format_cell(data: vlabel(ybuilt) || ":",
                   just: "left",
                   style_attr: "font_weight=bold");
    dak.format_cell(data: ybuilt, just: "left");
  dak.row_end();
  dak.row_start();
    dak.format_cell(data: vlabel(propType) || ":",
                   just: "left",
                   style_attr: "font_weight=bold");
    dak.format_cell(data: propType, just: "left");
    dak.format_cell(data: vlabel(style) || ":",
                   just: "left",
                   style_attr: "font_weight=bold");
    dak.format_cell(data: style, just: "left");
  dak.row_end();
dak.table_end();
```

To maintain readability, not all DATA step code is displayed here. Defining the other regions follows the same concept as for the region above. First, a REGION method defines the space for the content. Then a FORMAT_TEXT method displays a section header, followed by a table structure that displays the labels and values of variables from the input data set inside this space.

After all the regions for the one page report are defined, the absolute layout is ended with the LAYOUT_END method.

```
dak.layout_end();
```

The last step for each observation is to decide if a page break is necessary. The IF statement below adds a page break for each observation that is not flagged as the last one. Using this logic avoids an empty page after the last observation.

```
if not done then do;
  dak.page();
end;
```

End the DATA step with a RUN statement.

```
run;
```

CONCLUSION

ODS LAYOUT provides an easy way to arrange output objects from multiple sources in a report. The ODS Report Writing Interface provides methods inside the DATA step to arrange data values from an input data set.

If and how you use ODS LAYOUT or the ODS Report Writing Interface depends on your reporting needs. Both provide new ways to arrange different types of output within a report with ease. Before ODS LAYOUT and the ODS Report Writing Interface were available, reports like the ones presented in this paper were just not possible. Perhaps the use of macros, data manipulation, and reporting procedures would get you close but the code necessary is large, complicated, and hard to maintain.

ODS LAYOUT and the ODS Report Writing Interface are great features by themselves. There are almost no constraints about using one but not the other or both together. When using them together, the ODS Report Writing Interface can be used to provide content for a region in an ODS LAYOUT. However, it is not possible to use ODS LAYOUT statements inside the DATA step to arrange data set values. For this reason, the ODS Report Writing Interface has its own set of methods to arrange the data in a gridded or absolute way.

The possibilities ODS LAYOUT and the ODS Report Writing Interface offer are countless and this paper literally only showed the tip of the iceberg. Both will elevate your reports to new heights were not the sky but your imagination is the limit.

REFERENCES

- O'Connor, Daniel. 2013. "Take Home the ODS Crown Jewels: Master the New Production Features of ODS LAYOUT and Report Writing Interface Techniques." *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings13/015-2013.pdf>.
- O'Connor, Daniel and Scott Huntley. 2009. "Breaking New Ground with SAS® 9.2 ODS Layout Enhancements." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings09/043-2009.pdf>.
- O'Connor, Daniel. 2009. "The Power to Show: Ad Hoc Reporting, Custom Invoices, and Form Letters." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/base/datastep/dsubject/Power_to_show_paper.pdf.

ACKNOWLEDGMENTS

The author would like to thank Allison Crutchfield and Susan Owenby for their contributions to this paper.

RECOMMENDED READING

- *SAS® 9.4 Output Delivery System: User's Guide*
 - Chapter 19: Report Writing Interface
 - Chapter 21: ODS LAYOUT
- *Step-By-Step Programming with Base SAS® 9.4*

CONTACT INFORMATION

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

Daniel Kummer
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
daniel.kummer@sas.com
<http://www.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.

APPENDIX

Please notice that copying multiple pages of code at once will include the page numbers, too. They need to be removed before submitting the code. Otherwise the code will generate errors and will not run successfully.

ODS LAYOUT EXAMPLES

Sample Data

```
/* custom formats are used to translate do loop numbers into actual data */
proc format;
value stateP
  1="Alaska" 2="Washington" 3="Oregon" 4="California" 5="Hawaii";
value propG
  1="Residential" 2="Commercial";
value resT
  1="Condominiums" 2="Modular Homes" 3="Single-Family"
  4="Townhouses" 5="Mobile Homes" 6="Farm Houses";
value comT
  1="Office" 2="Land" 3="Industrial"
  4="Retail" 5="Leisure" 6="Multi-Family";
run;

/* this data set is used to create all bar & pie charts */
data acmePacific;
length year 8 region division state propGroup propType $30.
       soldUnits medianListPrice 8;
format soldUnits commal2. medianListPrice dollar12.;
do year=2010 to 2013;
  regionN = 4;
  region = "West";
  divisionN = 9;
  division = "Pacific";
  do stateN = 1 to 5;
    state = put(stateN,stateP.);
    do propGroupN =1 to 2;
      propGroup = put(propGroupN,propG.);
      do propTypeN = 1 to 6;
        if propGroupN = 1 then propType = put(propTypeN,resT.);
        else propType = put(propTypeN,comT.);
        soldUnits = 45000*ranuni(10000)+10000;
        medianListPrice = 200000*ranuni(199000)+199000;
        output;
      end;
    end;
  end;
end;
run;

/* aggregated data from previous data step, this one is used for
   the table structure displayed in the absolute layout example */
proc sql;
create table pacificSum as
select state,
       year,
       propGroup,
       sum(soldUnits) as sUnits format=commal2.,
       avg(medianListPrice) as mPrice format=dollar12.
  from acmePacific
  where year in (2012, 2013)
  group by state, year, propGroup
  ;
quit;
```

Gridded Layout Example

```
/* define general options */
options nodate nonumber papersize=letter
      topmargin=.25in bottommargin=.25in leftmargin=.25in rightmargin=.25in;

ods escapechar="^";

title;
footnote;

/* define colors used for the background */
%let backBlue = cx007DC3;

/* sort data for by-group processing */
proc sort data=acmePacific out=acmeSorted;
  by state propGroup;
run;

/* close all ODAs and open the PDF one */
ods _ALL_ close;

ods pdf file="pacificGridded.pdf" notoc nogtitle;

/* start a gridded layout */
ods layout gridded columns=2 column_gutter=0in advance=bygroup;

  /* define header section of the report - blue bar with company logo */
  ods region height=1in column_span=2 style={background=&backBlue. just=right};
  ods text="^S={background=&backBlue. foreground=white font_size=50pt}ACME";
  ods text="^S={background=&backBlue. foreground=white font_size=10pt}Sales Report
Pacific Region";

  /* define generic region to display individual charts */
  ods region;
  /* define goptions, axis pattern & legend settings */
  goptions reset=goptions noborder device=SASPRTC target=SASPRTC
    ftext='Helvetica' htext=.9 ftitle='Helvetica/bold'
    hsize=4in vsize=2.5in;

  axis1 label=none value=none;
  axis2 label=none minor=none offset=(0,0);
  axis3 label=none offset=(1,1) stagger;

  pattern1 v=s color=cx9AC1CF; /* light blue */
  pattern2 v=s color=cx245397; /* medium blue */

  legend1 label=none across=2 position=(top right inside) mode=protect
    shape=bar(.1in,.1in) value=(j=1);

  /* gchart procedure to generate bar charts */
  proc gchart data=acmeSorted;

    vbar year / discrete type=sum
      group=propType
      sumvar=soldUnits
      subgroup=year /* this controls the coloring */
      maxis=axis1 /* midpoint axis */
      raxis=axis2 /* response/numeric axis */
      gaxis=axis3 /* group axis */
      legend=legend1 /* assign the legend statement to graph */
      noframe /* no frame around graph */
      outline=black /* outline color of bars */
```



```

                space=0          /* bar spacing */
                gspace=.5       /* group spacing */
            ;
            by state propGroup;
            label state="State" propGroup="| Property Group";
            where year in (2012, 2013);
        run;
        quit;

/* end the gridded layout */
ods layout end;

ods _ALL_ close;

```

Absolute Layout Example

```

/* define general options */
options nodate nonumber papersize=letter
        topmargin=.25in bottommargin=.25in leftmargin=.25in rightmargin=.25in;

ods escapechar="^";

title;
footnote;

/* define colors used for the background */
%let backBlue = cx007DC3;
/* define image path */
%let imagePath = c:\temp\sgf\pacificMed.jpg;

/* close all ODAs and open the PDF one */
ods _ALL_ close;
ods pdf file="pacificAbsolute.pdf" notoc;

/* start a absolute layout */
ods layout absolute;
/* define header section of the report - blue bar with company logo */
ods region height=1in style={background=&backBlue.};
ods text=" ";

ods region x=5.5in width=2.5in height=1in style={background=&backBlue.};
ods text="^S={background=&backBlue. foreground=white font_size=50pt}ACME";
ods text="^S={background=&backBlue. foreground=white font_size=10pt}Sales Report
Pacific Region";

/* region to display text above the bar chart */
ods region y=1.25in width=4.75in height=1.25in;
ods text="^S={font_size=8pt}Commercial property sales accelerated in 2013,
continuing this year's recovery after activity came to a near-halt in 2012 during the
financial crisis and recession.";
ods text=" ";
ods text="^S={font_size=8pt}Buoyed by steady job growth, improving real estate
fundamentals and eager capital, buyers throughout the first 11 months of 2013 ponied
up more than $305 billion to acquire office buildings, industrial properties, retail
centers, hotels, apartments and development sites around the U.S., according to ACME
Analytics, which tracks commercial real estate sales of more than $5 million.";

/* region for the bar chart */
ods region y=2.75in width=4.75in height=2.75in;
/* define goptions, axis pattern & legend settings for bar chart */
goptions reset=goptions noborder device=SASPRTC target=SASPRTC
        ftext='Helvetica' htext=.9 ftitle='Helvetica/bold';

```

```

axis1 label=none value=none;
axis2 label=none minor=none offset=(0,0) order=(800000 to 1150000 by 50000);
axis3 label=none offset=(1,1);

pattern1 v=s color=cx245397;   /* medium blue #245397;
pattern2 v=s color=cx9AC1CF;   /* light blue #9AC1CF;

legend1 label=none across=1 position=(top right outside) mode=protect
       shape=bar(.25in,.15in) value=(j=c);

/* define title - region containers can display titles & footnotes */
title1 j=left 'Units Sold by Property Group in the Last 4 Years';

/* gchart procedure to generate bar chart */
proc gchart data=acmePacific;

    vbar propGroup / discrete type=sum
        group=year
        sumvar=soldUnits /* */
        subgroup=propGroup/* this controls the coloring */
        maxis=axis1      /* midpoint axis */
        raxis=axis2      /* response/numeric axis */
        gaxis=axis3      /* group axis */
        legend=legend1   /* assign the legend statement to graph */
        noframe          /* no frame around graph */
        outline=black    /* outline color of bars */
        space=1          /* bar spacing */
        gspace=2         /* group spacing */

;
run;
quit;

/* region to display the header above the image */
ods region x=5in y=1.25in width=3in height=0.18in style={background=&backBlue.};
ods text="^S={just=center background=&backBlue. foreground=white
font_size=10pt}States of the Pacific Region";

/* region to display image of the pacific census region */
ods region x=5in y=1.6in width=3in;
data _NULL_;
dcl odsout dak();
dak.image(file: "&imagePath.");
run;

/* region to display first pie chart */
ods region y=5.5in height=2.5in width=3in;

/* define goptions & pattern statements for pie charts */
goptions reset=goptions noborder device=sasprtc
         ftext='Helvetica' htext=.9 ftitle='Helvetica/bold';

title;

pattern1 v=s color=cx1E1E1E;
pattern2 v=s color=cx03658C;
pattern3 v=s color=cxF1B33C;
pattern4 v=s color=cxDADFE2;
pattern5 v=s color=cxD94D3E;
pattern6 v=s color=cx6B8E23;

/* gchart procedure to generate pie chart */
proc gchart data=acmePacific;
    pie propType / sumvar=soldUnits

```

```

        clockwise
        slice=outside
        percent=outside
        value=none
        noheading;
    where year=2013 and propGroup = "Commercial";
run;
quit;

/* region to display the header for 2013 & commercial */
ods region y=5.5in width=8in height=0.18in style={background=&backBlue.};
ods text="^S={just=center background=&backBlue. foreground=white
font_size=10pt}2013 Numbers for Commercial Properties";

/* region to display the first custom RWI table */
ods region x=3.5in y=5.8in height=2.2in;
data kumma;
set pacificSum (where=(propGroup="Commercial")) end=done;
by state year;
retain units12 pricel2;

if _N_ = 1 then do;
dcl odsout dak();
dak.table_start();
dak.row_start();
    dak.format_cell(data: "State", row_span: 2, vjust: "center",
        style_attr: "font_weight=bold");
    dak.format_cell(data: "Units Sold", column_span: 2,
        style_attr: "font_weight=bold");
    dak.format_cell(data: "Median List Price", column_span: 2,
        style_attr: "font_weight=bold");
dak.row_end();
dak.row_start();
    dak.format_cell(data: "#", style_attr: "font_weight=bold",
        width: ".75in");
    dak.format_cell(data: "% Change *", style_attr: "font_weight=bold",
        width: ".75in");
    dak.format_cell(data: "#", style_attr: "font_weight=bold",
        width: ".75in");
    dak.format_cell(data: "% Change *", style_attr: "font_weight=bold",
        width: ".75in");
    dak.row_end();
end;

if first.state and first.year then do;
units12 = sUnits;
pricel2 = mPrice;
end;

if last.state and last.year then do;
changeUnits = ( ( sUnits - units12 ) / units12 );
changePrice = ( ( mPrice - pricel2 ) / pricel2 );

dak.row_start(style_attr: "just=left");
dak.format_cell(data: state, just: "left");
dak.format_cell(data: sUnits, format: "comma12.");
if changeUnits GE 0 then dak.format_cell(data: changeUnits,
        format: "percentn10.3",
        style_attr: "foreground=green");
else dak.format_cell(data: changeUnits,
        format: "percentn10.3",
        style_attr: "foreground=red");
dak.format_cell(data: mPrice, format: "dollar12.");

```

```

        if changePrice GE 0 then dak.format_cell(data: changePrice,
                                                format: "percentn10.3",
                                                style_attr: "foreground=green");
    else dak.format_cell(data: changePrice,
                        format: "percentn10.3",
                        style_attr: "foreground=red");
    dak.row_end();
end;

if done then do;
    dak.row_start();
    dak.format_cell(data: "* compared to last Years Results",
                    just: "left", column_span: 5, inhibit: "LBR");
    dak.row_end();
    dak.table_end();
end;

run;

/* region to display second pie chart */
ods region x=4in y=8in height=2.5in width=4in;
/* gchart procedure to generate pie chart */
proc gchart data=acmePacific;
    pie propType / sumvar=soldUnits
                  clockwise
                  slice=outside
                  percent=outside
                  value=none
                  noheading;
    where year=2013 and propGroup = "Residential";
run;
quit;

/* region to display the header for 2013 & residential */
ods region y=8in width=8in height=0.18in style={background=&backBlue.};
ods text="^S={just=center background=&backBlue. foreground=white
font_size=10pt}2013 Numbers for Residential Properties";

/* region to display the second custom RWI table */
ods region y=8.3in width=4.25in height=2.2in;
data kumma;
    set pacificSum (where=(propGroup="Residential")) end=done;
    by state year;
    retain units12 price12;

if _N_ = 1 then do;
    dcl odsout dak();
    dak.table_start();
    dak.row_start();
    dak.format_cell(data: "State", row_span: 2, vjust: "center",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: "Units Sold", column_span: 2,
                    style_attr: "font_weight=bold");
    dak.format_cell(data: "Median List Price", column_span: 2,
                    style_attr: "font_weight=bold");
    dak.row_end();
    dak.row_start();
    dak.format_cell(data: "#", style_attr: "font_weight=bold",
                    width: ".75in");
    dak.format_cell(data: "% Change *", style_attr: "font_weight=bold",
                    width: ".75in");
    dak.format_cell(data: "#", style_attr: "font_weight=bold",
                    width: ".75in");

```

```

        dak.format_cell(data: "% Change *", style_attr: "font_weight=bold",
            width: ".75in");
    dak.row_end();
end;

if first.state and first.year then do;
    units12 = sUnits;
    price12 = mPrice;
end;

if last.state and last.year then do;
    changeUnits = ( ( sUnits - units12 ) / units12 );
    changePrice = ( ( mPrice - price12 ) / price12 );

    dak.row_start(style_attr: "just=left");
    dak.format_cell(data: state, just: "left");
    dak.format_cell(data: sUnits, format: "commal2.");
    if changeUnits GE 0 then dak.format_cell(data: changeUnits,
        format: "percentn10.3",
        style_attr: "foreground=green");
    else dak.format_cell(data: changeUnits,
        format: "percentn10.3",
        style_attr: "foreground=red");
    dak.format_cell(data: mPrice, format: "dollar12.");
    if changePrice GE 0 then dak.format_cell(data: changePrice,
        format: "percentn10.3",
        style_attr: "foreground=green");
    else dak.format_cell(data: changePrice,
        format: "percentn10.3",
        style_attr: "foreground=red");

    dak.row_end();
end;

if done then do;
    dak.row_start();
    dak.format_cell(data: "* compared to last Years Results",
        just: "left", column_span: 5, inhibit: "LBR");
    dak.row_end();
    dak.table_end();
end;

run;

/* end the absolute layout */
ods layout end;

ods pdf close;

```

ODS REPORT WRITING INTERFACE EXAMPLE

Sample Data

```
data acmeHouses;
```

```
length state $2 basement porch inCity fire deck porch $3
       zip sizeMBed size2Bed size3Bed size4Bed sizeKitchen sizeFamRoom sizeDinRoom
       sizeOffice sizeSunRoom sizeBase sizeDeck sizePorch sizeEnt $5
       listNum listArea $6 extFinish roof floor foundation cool heat style area
       county $20 name street1 city propType elemSchool middleSchool highSchool
       contact email webUrl $50 desc $250;
```

```
listNum="000001"; listArea="666"; name="Hill House"; street1="1328 Elm Street";
city="Springwood"; zip="23487"; state="OH"; price=275643; propType="Single-Family";
yBuilt=1823; extFinish="Wood"; roof="Shingles"; floor="Hardwood";
foundation="Concrete"; numGarage=0; numLevel=4; heatArea=2453; addArea=.;
basement="Yes"; cool="Forced"; heat="Fireplace"; bRooms=1; bath=0; lotSize=25;
houseSize=2453; style="Traditional"; assocFee=.; inCity="No"; area="Made Up Area";
county="Fake"; fire="Yes"; deck="Yes"; porch="Yes"; sizeMBed="20X20"; size2Bed="";
size3Bed=""; size4Bed=""; sizeKitchen="15x10"; sizeFamRoom="20x20"; sizeDinRoom="";
sizeOffice="15x10"; sizeSunRoom=""; sizeBase="45x30"; sizeDeck=""; sizePorch="20x5";
sizeEnt=""; elemSchool="Voorhees Elementary"; middleSchool="Myers Middle";
highSchool="Krueger High"; contact="Laurie Strode"; officePhone="(666)999-0000 ext.
2"; officeFax="(666)999-0006"; cellPhone="(666)999-6666";
eMail="laurie@smithsgrove.com"; webUrl="www.smithsgrove.com";
desc="This fixer upper is ready for you. This large plantation house with a low
maintenance yard just needs a little TLC. Full Disclosure: A night spent here could
turn into an eternity";
output;
```

```
listNum="000002"; listArea="555"; name="The Farnsworth House"; street1="1800 Fancy
Drive"; city="Fancy Town"; zip="66666"; state="CA"; price=2500000; propType="Vacation
Home"; yBuilt=1951; extFinish="Steel & Glass"; roof="Metal"; floor="Tile";
foundation="Steel Columns"; numGarage=0; numLevel=1; heatArea=.; addArea=.;
basement="No"; cool="None"; heat="Fireplace"; bRooms=1; bath=1; lotSize=60;
houseSize=1554; style="Modern"; assocFee=.; inCity="No"; area="Made Up Area";
county="Fake"; fire="Yes"; deck="Yes"; porch="No"; sizeMBed="20X10"; size2Bed="";
size3Bed=""; size4Bed=""; sizeKitchen="20x4"; sizeFamRoom="30x20"; sizeDinRoom="";
sizeOffice=""; sizeSunRoom=""; sizeBase=""; sizeDeck="15x40"; sizePorch="20x20";
sizeEnt=""; elemSchool="Peter Parker Elementary"; middleSchool="Bruce Wayne Middle
School"; highSchool="Han Solo High"; contact="Wile E. Coyote"; officePhone="(666)999-
0000 ext. 1"; officeFax="(666)999-0006"; cellPhone="(666)999-6661";
eMail="wile.e.coyote@acmefakecorp.com"; webUrl="www.acmefakecorp.com";
desc="As the perfect family vacation retreat the Farnsworth House is lying in a once-
rural setting and seems to float weightlessly above the ground it occupies. It has
become an iconic expression of 20th century modernist architecture.";
output;
```

```
label listNum="Listing Number" listArea="Listing Area" propType="Property Type"
       bRooms="Bed Rooms" bath="Bath Rooms" houseSize="House Size" lotSize="Lot Size"
       style="Style" price="Price" yBuilt="Year Built" extFinish="Exterior Finish"
       roof="Roof" floor="Floor" foundation="Foundation" heatArea="Heated Area"
       numGarage="# Garages" numLevel="# Levels" addArea="Additional Areas" area="Area"
       basement="Basement" cool="Cooling" heat="Heating" inCity="Inside City Limits"
       county="County" fire="Fireplace" deck="Deck" porch="Screened Porch";
```

```
run;
```

ODS RWI Absolute Layout Example

```
/* define general options */
options nodate nonumber papersize=letter
      topmargin=.25in bottommargin=.25in leftmargin=.25in rightmargin=.25in;

ods escapechar="^";

title;
footnote;

/* define colors used for the background */
%let backBlue = cx007DC3;
%let backGray = cxA9A9A9;

/* define location of images */
%let imageLoc = c:\temp\sgf\;

/* close all ODAs and open the PDF one */
ods _ALL_ close;

ods pdf file="pacificRWI.pdf" notoc;

data _NULL_;

  set acmeHouses end=done;

  /* declare the odsout output object only once */
  if _N_ = 1 then do;
    dcl odsout dak();
  end;

  /* start a gridded layout */
  dak.layout_absolute();

  /* define header section of the report - blue bar with company logo */
  dak.region(width: "8in",height: "1in", style_attr: "background=&backBlue.");
  dak.format_text(data: "", style_attr: "background=&backBlue.");
  dak.region(x: ".5in", y: ".2in", width: "3in", height: "1in");
  dak.format_text(data: name, style_attr: "background=&backBlue.
      foreground=white
      font_size=20pt");
  dak.format_text(data: street1, style_attr: "background=&backBlue.
      foreground=white
      font_size=15pt");
  dak.region(x: "5.5in", width: "2in", height: "1in");
  dak.format_text(data: "ACME", just: "right", style_attr: "background=&backBlue.
      foreground=white
      font_size=50pt");
  dak.region(x: "5.5in", y: ".7in", width: "2in", height: ".5in");
  dak.format_text(data: "Pacific Region", just: "right",
      style_attr: "background=&backBlue.
      foreground=white
      font_size=10pt");

  /* region to display image */
  dak.region(y: "1.25in", width: "4in", height: "3in");
  dak.image(file: "&imageLoc." || strip(listNum) || ".jpg");

  /* region to display description & general information */
  dak.region(x: "4in", y: "1.25in", width: "4in", height: "3in");
  /* display text that is formatted to appear as section header */
  dak.format_text(data: "General Information",
```

```

                style_attr: "font_size=12pt font_weight=bold width=100%
                            just=c background=&backGray");
/* add empty row to create space between header text and description */
dak.format_text(data: "");
/* display description */
dak.format_text(data: desc);
/* arrange the data labels & values using a table */
dak.table_start(style_attr:"frame=void rules=none width=4in");
dak.row_start();
    dak.format_cell(data: vlabel(brooms) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: brooms, just: "left");
    dak.format_cell(data: vlabel(bath) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: bath, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: vlabel(houseSize) || " (SqFt):", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: houseSize, just: "left", format: "comma12.");
    dak.format_cell(data: vlabel(lotSize) || " (Acres):", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: lotSize, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: vlabel(price) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: price, format: "dollar14.", just: "left");
    dak.format_cell(data: vlabel(ybuilt) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: ybuilt, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: vlabel(propType) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: propType, just: "left");
    dak.format_cell(data: vlabel(style) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: style, just: "left");
dak.row_end();
dak.table_end();

/* region to display building features */
dak.region(y: "4.5in", width: "8in", height: "3in");
/* display text that is formatted to appear as section header */
dak.format_text(data: "Building Features",
                style_attr: "font_size=12pt font_weight=bold width=100%
                            just=c background=&backGray");
dak.format_text(data: "");
/* arrange the data labels & values using a table */
dak.table_start(style_attr:"frame=void rules=none width=8in");
dak.row_start();
    dak.format_cell(data: vlabel(extFinish) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: extFinish, just: "left");
    dak.format_cell(data: vlabel(numGarage) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: numGarage, just: "left");
    dak.format_cell(data: vlabel(cool) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: cool, just: "left");
    dak.format_cell(data: vlabel(inCity) || ":", just: "left",
                    style_attr: "font_weight=bold");

```



```

    dak.format_cell(data: inCity, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: vlabel(roof) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: roof, just: "left");
    dak.format_cell(data: vlabel(numLevel) || ":", just: "left",
                    just: "left", style_attr: "font_weight=bold");
    dak.format_cell(data: numLevel, just: "left");
    dak.format_cell(data: vlabel(heat) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: heat, just: "left");
    dak.format_cell(data: vlabel(area) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: area, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: vlabel(floor) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: floor, just: "left");
    dak.format_cell(data: vlabel(heatArea) || " (SqFt):", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: heatArea, just: "left");
    dak.format_cell(data: vlabel(assocFee) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: assoc, just: "left");
    dak.format_cell(data: vlabel(county) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: county, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: vlabel(foundation) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: foundation, just: "left");
    dak.format_cell(data: vlabel(addArea) || " (SqFt):", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: addArea, just: "left");
    dak.format_cell(data: vlabel(deck) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: deck, just: "left");
    dak.format_cell(data: vlabel(listNum) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: listNum, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: vlabel(basement) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: basement, just: "left");
    dak.format_cell(data: vlabel(fire) || ":", just: "left",
                    just: "left", style_attr: "font_weight=bold");
    dak.format_cell(data: fire, just: "left");
    dak.format_cell(data: vlabel(porch) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: porch, just: "left");
    dak.format_cell(data: vlabel(listArea) || ":", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: listArea, just: "left");
    dak.row_end();
dak.table_end();

/* region to display room dimensions */
dak.region(y: "6.5in", width: "8in", height: "3in");
/* display text that is formatted to appear as section header */

```

```

dak.format_text(data: "Room Dimensions",
                style_attr: "font_size=12pt font_weight=bold width=100%
                            just=c background=&backGray");
dak.format_text(data: "");
/* arrange the data labels & values using a table */
dak.table_start(style_attr:"frame=void rules=none width=8in");
dak.row_start();
    dak.format_cell(data: "Master Bedroom:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeMBed, just: "left");
    dak.format_cell(data: "Kitchen:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeKitchen, just: "left");
    dak.format_cell(data: "Sun Room:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeSunRoom, just: "left");
    dak.format_cell(data: "Deck:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeDeck, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: "2nd Bedroom:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: size2Bed, just: "left");
    dak.format_cell(data: "Family Room:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeFamRoom, just: "left");
    dak.format_cell(data: "Entrance:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeEnt, just: "left");
    dak.format_cell(data: "Porch:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizePorch, just: "left");
dak.row_end();
dak.row_start();
    dak.format_cell(data: "3rd Bedroom:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: size3Bed, just: "left");
    dak.format_cell(data: "Dining Room:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeDinRoom, just: "left");
    dak.format_cell(data: "Basement:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeBase, just: "left");
    dak.format_cell(data: "");
    dak.format_cell(data: "");
dak.row_end();
dak.row_start();
    dak.format_cell(data: "4th Bedroom:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: size4Bed, just: "left");
    dak.format_cell(data: "Office:", just: "left",
                    style_attr: "font_weight=bold");
    dak.format_cell(data: sizeOffice, just: "left");
    dak.format_cell(data: "");
    dak.format_cell(data: "");
    dak.format_cell(data: "");
    dak.format_cell(data: "");
    dak.row_end();
dak.table_end();

/* region to display school information */
dak.region(y: "8.25in", width: "8in", height: "1in");

```

```

/* display text that is formatted to appear as section header */
dak.format_text(data: "School Information",
                style_attr: "font_size=12pt font_weight=bold width=100%
                            just=c background=&backGray");
dak.format_text(data: "");
/* arrange the data labels & values using a table */
dak.table_start(style_attr:"frame=void rules=none width=8in");
dak.row_start();
  dak.format_cell(data: "Elementary School:", just: "left",
                 style_attr: "font_weight=bold");
  dak.format_cell(data: elemSchool, just: "left");
  dak.format_cell(data: "Middle School:", just: "left",
                 style_attr: "font_weight=bold");
  dak.format_cell(data: middleSchool, just: "left");
  dak.format_cell(data: "High School:", just: "left",
                 style_attr: "font_weight=bold");
  dak.format_cell(data: highSchool, just: "left");
dak.row_end();
dak.table_end();

/* region to display contact information */
dak.region(y: "9.2in", width: "8in", height: "1.25in");
/* display text that is formatted to appear as section header */
dak.format_text(data: "Contact",
                style_attr: "font_size=12pt font_weight=bold width=100%
                            just=c background=&backGray");
/* arrange the data labels & values using a table */
dak.table_start(style_attr:"frame=void rules=none width=5in");
dak.row_start();
  dak.format_cell(data: contact, just: "left",
                 style_attr: "font_size=10pt font_weight=bold",
                 column_span: 4);

dak.row_end();
dak.row_start();
  dak.format_cell(data: "Office: " || strip(officePhone), just: "left");
  dak.format_cell(data: "Cell: " || strip(cellPhone),
                 just: "center", column_span: 2);
  dak.format_cell(data: "Fax: " || strip(officeFax), just: "right");
dak.row_end();
dak.row_start();
  dak.format_cell(data: "eMail: " || strip(eMail),
                 just: "left", column_span: 2);
  dak.format_cell(data: strip(webUrl) || "/" || strip(listNum),
                 just: "right", column_span: 2);

  dak.row_end();
dak.table_end();

/* close the absolute layout */
dak.layout_end();

/* add a page break for each obs but the last */
if not done then do;
  dak.page();
end;
run;

ods _ALL_ close;

```