

Mastering Parameters in SAS® Visual Analytics

Stu Sztukowski, SAS Institute Inc.

ABSTRACT

SAS® Visual Analytics provides simple, straight-forward filtering mechanisms that work right out of the box with almost no configuration needed: drop a control (filter) into a report and assign a variable role to it. It's easy, quick, and satisfies most UI requirements for users. When used on their own, report controls only filter other objects and do not provide any additional information to you, the designer. For example, if a user selects a filter, it might be helpful to know:

- What filter did they select?
- What value or values does the filter hold?
- Can I use the value they selected in calculated variables?

Visual Analytics allows you to answer these questions through special variables called parameters.

Parameters are data set independent dynamic variables in Visual Analytics that can be added to report controls to store user-selected value(s) of lists, buttons, sliders, and other controls into a variable or group of variables. Often underused, parameters provide you with a powerful level of control over the report.

INTRODUCTION

There are three types of parameters available in Visual Analytics:

1. Character
2. Numeric
3. Date/datetime

The types of values that a parameter will accept are self-explanatory. Character parameters can only accept character values, numeric parameters can only accept numeric values, and date/datetime parameters can only accept date or datetime values.

Parameters are like categorical, measure, or datetime variables. They work with the same values, can be used in calculated variables, and can be assigned to controls. There are two differentiating factors that make them such powerful assets in a report:

1. They are data set agnostic: parameters are independent of report data sets
2. They are dynamic: they change values as users interact with them

When you create a parameter, it is tied solely to the report and not any one data set, unlike a calculated variable, which can only be used within the data set in which it was created. Due to this data set independence, parameters require some sort of input to give them a value. This is either a static value that you decide, or a dynamic value that is populated from a control in the report.

WHAT IS A PARAMETER, ANYWAY?

A parameter is defined as “a variable whose value can be changed and that can be referenced by other report objects.” What makes parameters unique from other variables is that they can be changed *by the user* using report controls. One way to think of a parameter is like a large drink jar. You can fill it with any liquids that you like: water, juice, wine, and so on. The jar will remain filled with your beverage of choice until you decide to change it, whether you empty it or switch it with another drink.

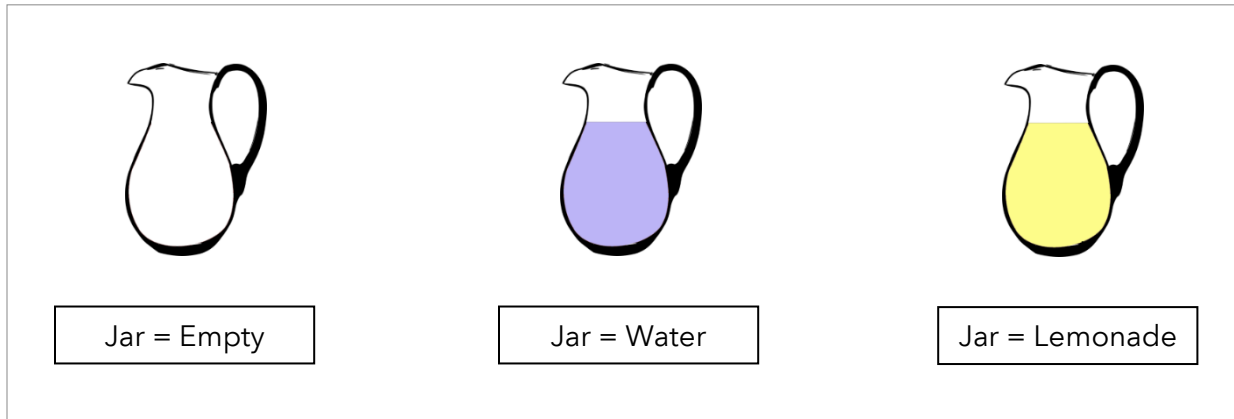


Figure 1. Parameters can be thought of as a container that holds something.

For those with SAS programming experience, think of a parameter like a macro variable. On its own, a macro variable has no value unless it is either given an initial value or changed at another time. For example, consider the statement below:

```
%global macvar;
```

The macro variable `macvar` is created and available for use. By default, its value is simply missing. `macvar` can hold a value by assigning one to it.

```
%let macvar = 1;
```

`macvar` now has a value of 1 and can be used in other calculations. It will always hold this value until you change it. Throughout your program, `macvar` is available to store values as you'd like. Like parameters, macro variables require something else to change its default value.

The best way to learn how to use parameters is by example. This paper will provide five examples of using parameters to enhance both Visual Analytics 7.4 and 8.2+ reports:

1. Dynamic ranks
2. Dynamic variables
3. Dynamic titles
4. Simple “What-If” calculators
5. Dynamic dates for drop-down list filters

All examples in this paper are accompanied by sample reports and data for Visual Analytics 7.4, 8.2, and 8.3.1.

DYNAMIC RANKS

For many business decisions, knowing the highest and lowest values in a category are important, especially if there are thousands of unique values. For example, a CEO might want to know the top 10 most profitable products that his or her company produces. Visual Analytics gives you the ability to apply a rank to a categorical variable for an object. Typically, this is a static value set by you, the designer. If you want to change it, you need to re-open the report in Designer Mode, change the value, and save it. With parameters, you can skip those steps and allow users to decide which top x or $x\%$ of a category to show.

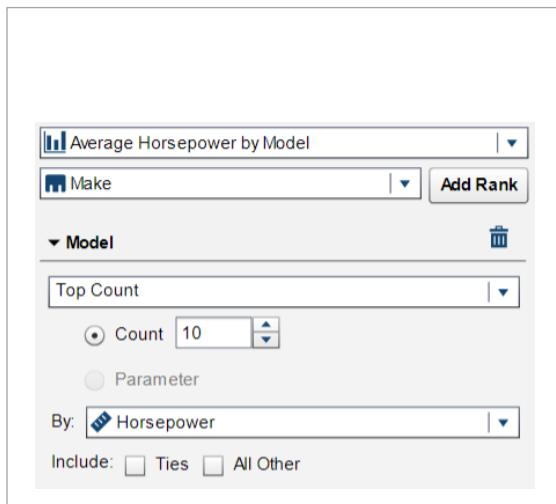


Figure 2. Applying a rank to a category (7.4).

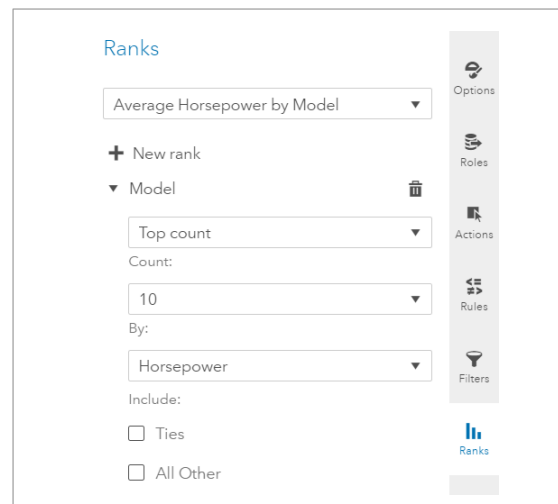


Figure 3. Applying a rank to a category (8.2+).

Suppose you are interested in buying a car. It needs to be fast – really fast. *sashelp.cars* is a data set conveniently located in your SASHELP directory that has vehicle statistics, and you also have access to Visual Analytics. You load *sashelp.cars* into Visual Analytics and create the bar chart in [Figure 4](#) that shows the average horsepower by model.

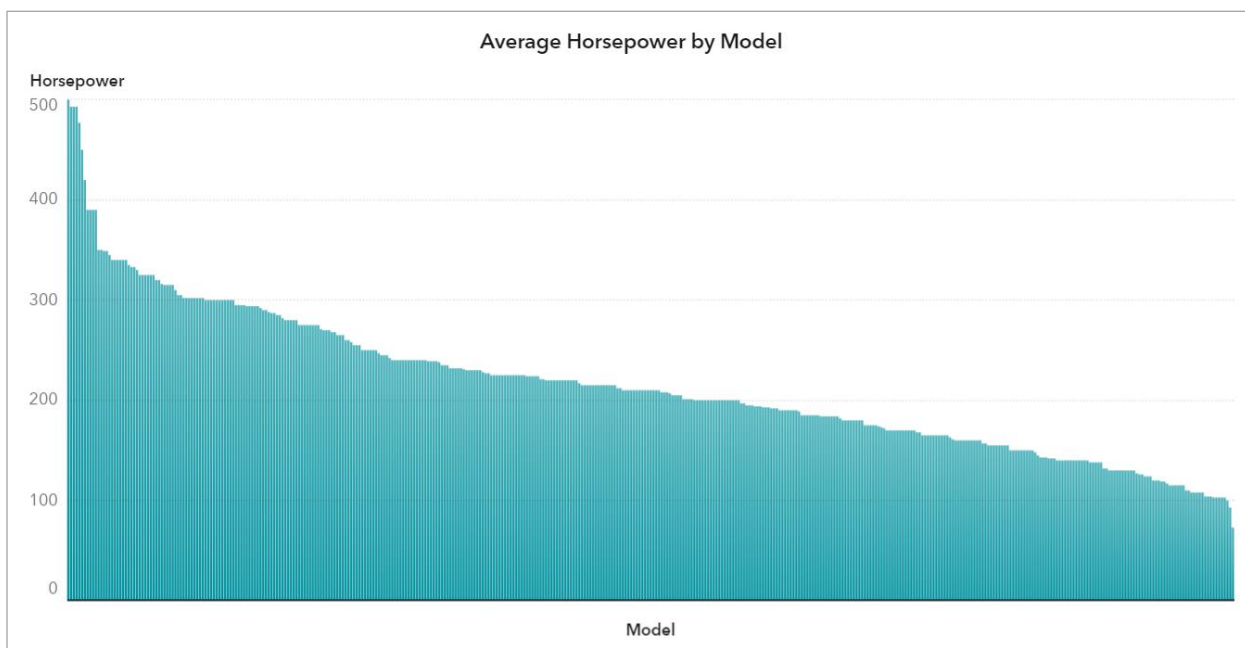




Figure 4. Bar chart of Horsepower by Model.

Yikes! That’s a lot of cars. To make this easier to see, you’d like to be able to rank the top x models by horsepower and give yourself the flexibility to change that value on the fly.

STEPS TO CREATE EXAMPLE REPORT

1. Load *sashelp.cars* as a promoted CAS table, or into LASR.
2. Drag a slider control anywhere into the report and leave it without any roles.
3. Create a bar chart and set the roles as follows:
 - a. **Category:** *Model*
 - b. **Measures:** *Horsepower* with an aggregation of *average*
4. Click on the bar chart and select the “Ranks” option. If this is not visible in Visual Analytics 7.4, click  on the right-hand corner of the screen and select  Ranks
5. In the drop-down list of the *Ranks* option menu, select *Model* and click *Add Rank*.
6. In the *By:* drop-down list, select *Horsepower*.
7. Uncheck *Ties* and *All Other*.

CREATING YOUR FIRST PARAMETER

There is an additional option to use a parameter under the *Count* option. Instead of forcing a static value of 10 for the *Top Makes by Horsepower*, you can take advantage of the *Parameter* option to let the user decide the top number of car makes by horsepower they would like to see. To use this, you first need to create a numeric parameter. This can be done in two different ways depending on your version of Visual Analytics.

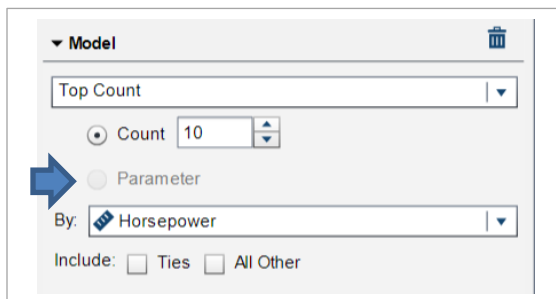


Figure 5. Parameter Option for Ranks (7.4).

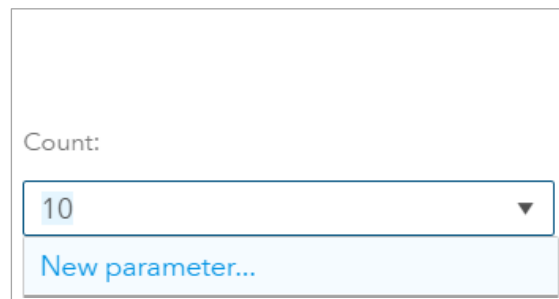





Figure 6. Parameter Option for Ranks (8.2+).

Visual Analytics 7.4

On the left side of the page, click , and create a new numeric parameter.

Visual Analytics 8.2+

You can create a new parameter by clicking  *New data item* in the *Data* tab on the left-hand side of the screen and selecting  *Parameter...*; however, Visual Analytics 8.2+ gives you a shortcut to create a parameter if it does not yet exist in the *Ranks* tab. Under *Count:*, select the drop-down menu and click *New parameter...*

Set the options in [Table 1](#) for the parameter:

Parameter Name	Min	Max	Current Value	Format	Decimals
User Rank	1	25	10	Float	0

Table 1. Options for User Rank parameter.

USING THE PARAMETER

Click on the slider you added to the report and add your new parameter to the *parameter* role. In the *Ranks* tab of your bar chart, ensure that your parameter is selected under the *Count* option.



Figure 7. Parameter Count (7.4).



Figure 8. Parameter Count (8.2+).

Save, and then switch back to the report viewer and try moving the slider like in [Figure 9](#).

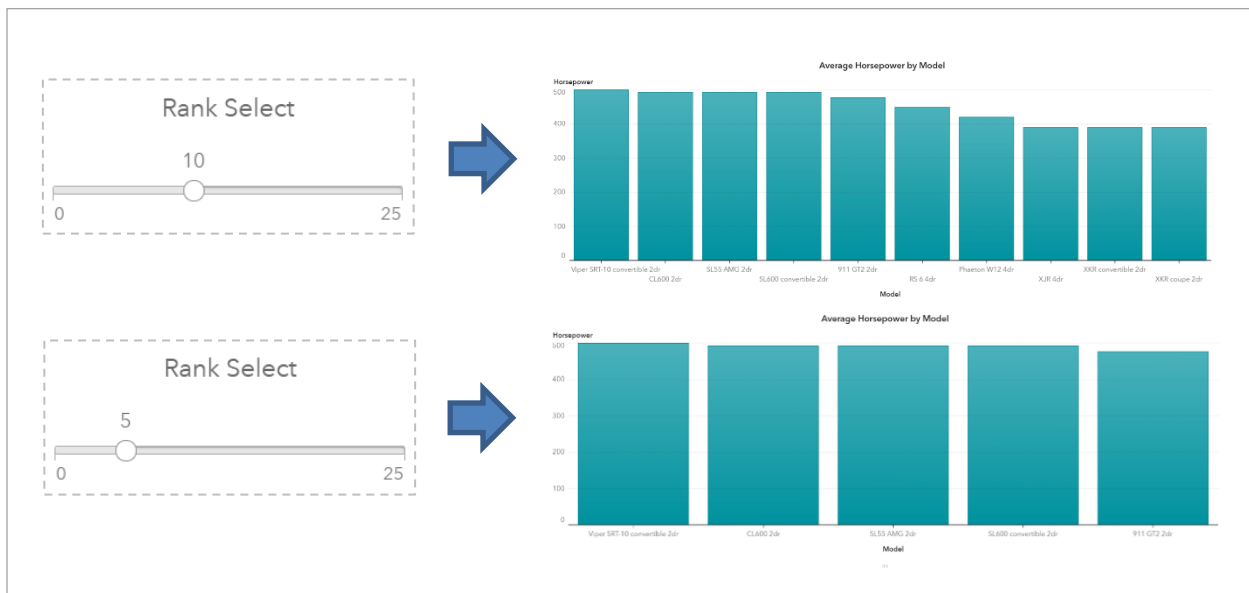


Figure 9. Using a slider with a parameter to control the top values displayed in a bar chart.

Thanks to your report, you've narrowed down the top 10 fastest vehicles on the market. Your significant other wanted to see the top 20, and you were able to show that easily thanks to your filter. After reviewing your options, you decide that a Dodge Viper might not be a good investment and settle on a gently used Nissan Sentra instead.

DYNAMIC VARIABLES

Within 500-1000ms of their first impression, people form their initial opinions of a person, and user interfaces are not much different. If the initial look of an interface is cluttered and full of graphs, it can quickly drive a user away. When users want to see multiple identical graphs for different variables in a single report, the report space can become overcrowded with stacked containers, tabs, and duplicated graphs whose only differences are the variables being displayed. If this happens, the user can feel lost or confused as to what they might be looking at or where to find the information that they want. This compounds further when interactions are added between various graphs in stacked containers.

A trick for giving users more flexibility to see data of interest without overcrowding a report is by allowing the user to select a variable to display in a graph using a button. If the button

were to dynamically change the variable being shown, then more information can be displayed on a page while introducing minimal clutter. Parameters give you this flexibility.

CREATING A UTILITY DATA SET

When you create a button, you can assign an optional character parameter to it. Unlike numeric or date parameters, character parameters cannot be assigned a set of static values. They can only hold the value of a pre-existing categorical variable that is assigned to the button. If you add a parameter to a button without adding a categorical data item, the button will not function. This same rule applies to drop-down menus and lists.

For example, using *sashelp.cars*, suppose you want to have a bar chart where users can choose to display the average City MPG or Highway MPG by Make via a button ([Figure 10](#)).



Figure 10. Variable selection button.

Buttons need a category to give them text since their primary function is for filtering. In this case, you don't want the button to filter anything. You need to do a trick to give the button text without letting it filter anything else in the report. A set of utility data sets will allow you to make any categories that you want without interacting with report objects or each other. Utility data sets are simple to create in SAS:

```
data va_dummy_data_body;
  do dummy_data_body = 1 to 25;
    output;
  end;
run;

data va_dummy_data_local;
  do dummy_data_local = 1 to 25;
    output;
  end;
run;

data va_dummy_data_global;
  do dummy_data_global = 1 to 25;
    output;
  end;
run;
```

Figure 11. Creating utility data sets using SAS.

Three data sets are created, all with different names and variables to prevent Visual Analytics from automatically assigning variable mappings. If Visual Analytics asks you to assign variable mappings for your utility data sets, do not assign them. Depending on where you place your controls, you'll want to use a data set in a specific location within the report. This keeps the controls independent of each other and prevents any unintended filtering.

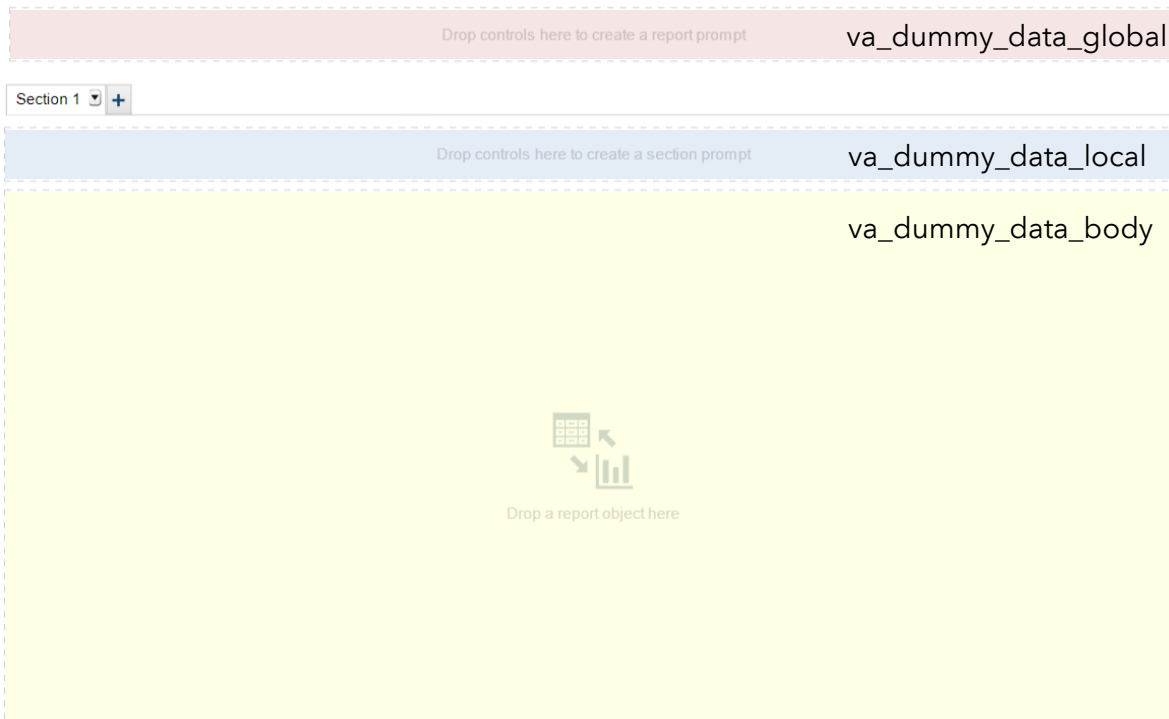


Figure 12. Where to use utility data sets in Visual Analytics report objects.

USING THE UTILITY DATA SET WITH A PARAMETER

For this example, drag a button to the body of the report, place it under an *Average MPG by Make* bar chart, and set it as *required*. Next, perform the following steps:

1. Add the utility data set *va_dummy_data_body* to your report.
2. Right-click the variable *dummy_data_body*, click *New Custom Category...*, and name it *Variable Select* ([Figure 13](#)).
3. Create a new group called *MPG (City)* and add a value to it.
4. Group the remaining values as *MPG (Highway)*.

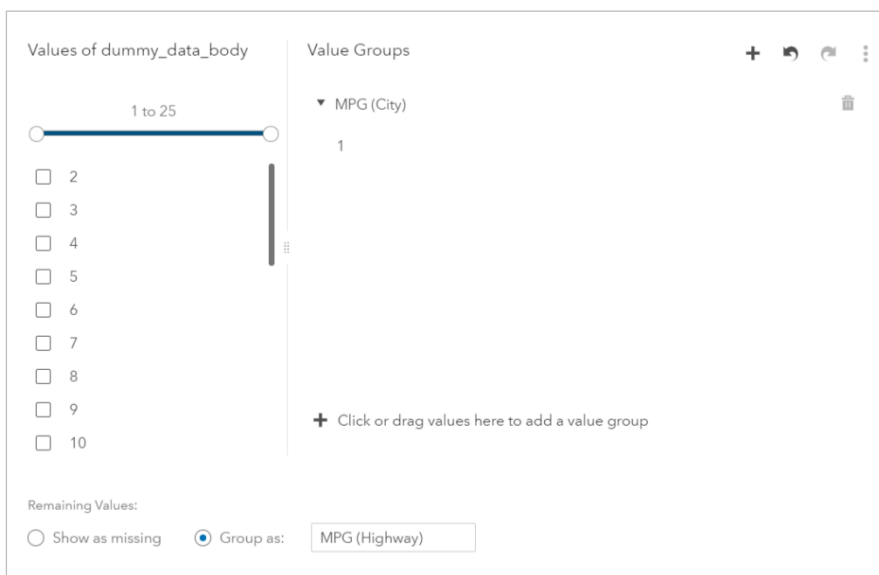


Figure 13. Creating button text using values of *dummy_data_body*.

5. Give *Variable Select* a role in the button.
6. Create a new character parameter and name it *Variable Select Parameter*.
7. Add *Variable Select Parameter* as a role in the button.

Great! You have a button with a parameter. Now you need to make the parameter do something. At the moment, it only takes on the value of whatever is selected on the button.

PUTTING THE PARAMETER TO WORK

You now have everything that you need to create a dynamically changing variable. The button logic is basic: if the user selects *MPG (City)*, you want to display values of *MPG (City)*. Otherwise, you want to display the *MPG (Highway)* variable. The value of the parameter tells you what the user has selected. Follow the next steps to finish the exercise.


1. Select the *cars* data set and create a new calculated variable named *MPG* with an aggregation of *average*.
2. Select the *text* tab and add the following code:

```
if(uppercase('Variable Select Parameter') = 'MPG (CITY)')
  return 'MPG (City)'
else 'MPG (Highway)'
```

3. Add your newly created *MPG* variable to the bar chart.
4. Click the button and observe the effects on the graph.

Note that the variable name *MPG* is always static because Visual Analytics does not support dynamic variable names as of this paper. The user only knows what is being displayed from the highlighted button. In reports where space is at a premium and controls are hidden in prompt containers, dynamic titles can help users know what is being shown.

DYNAMIC TITLES AND TEXT REPORTS

If you have dynamic variables or filters inside of a prompt container, users might be wondering which variables are being displayed or which filters are being applied. For example, selected values within prompt containers are hidden unless the user happens to click  on an individual object to see if any filters are applied.

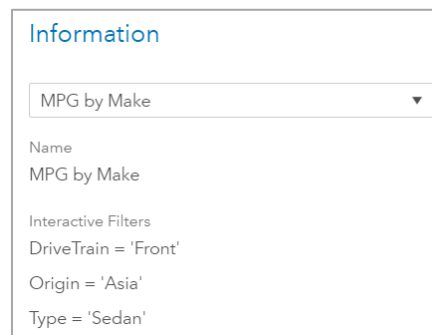


Figure 14. Object filter information displayed after clicking 

Instead, you can bring these filter values to the attention of the viewer at the top of the page by associating parameters with each report control and using them within a text box.

CREATING DYNAMIC TITLES FROM PARAMETERS

Expanding on the [previous example](#), add the following filters to the report:

1. Add three required list controls in a prompt container: *DriveTrain*, *Type*, and *Origin*.
2. Create three character parameters and add each parameter to a report control ([Figure 15](#)).

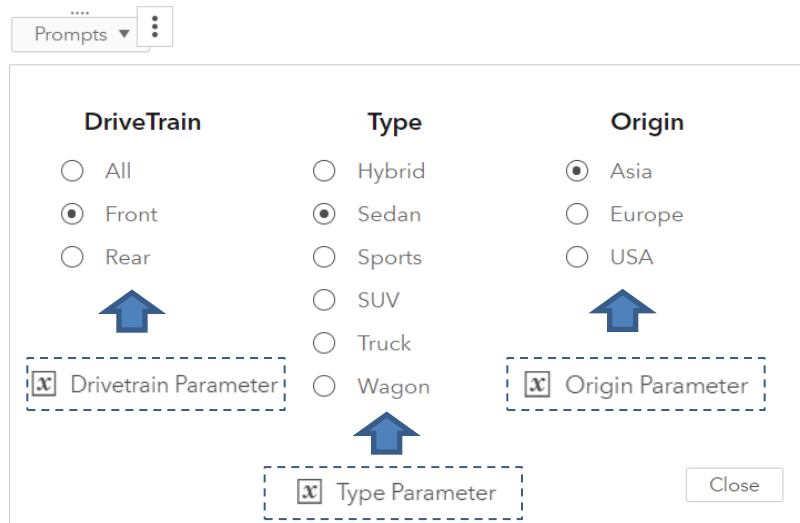


Figure 15. Adding three new parameters to report controls.

You want to create some descriptive text that tells the user about the filters selected. For example, if the user selects front wheel drive Asian sedans, you might want something at the top of the report that says:

Average **MPG (Highway)** of **Front Wheel Drive Sedans** made in **Asia**

To do this, create a new text box at the top of your report and begin entering the title. For each part of the title that is dynamic, add the desired parameter to the text box.

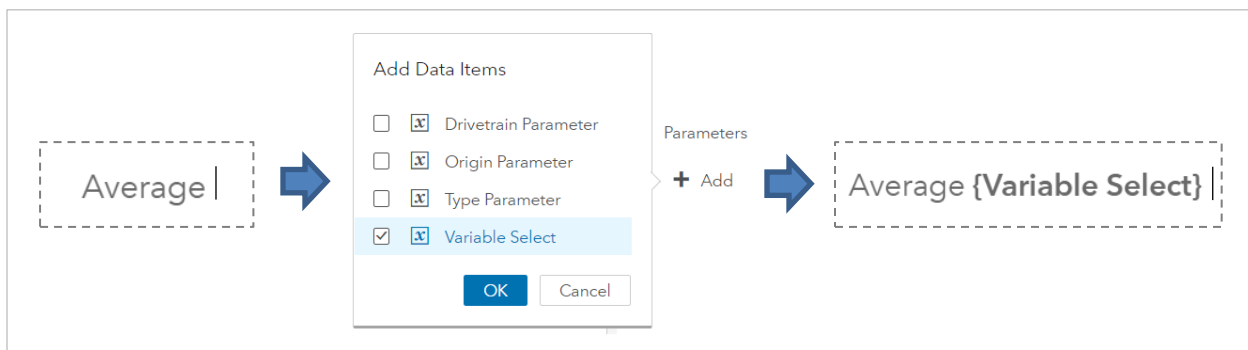


Figure 16. Incorporating parameters as dynamic text into a text box.

Continue entering your title and adding parameters as needed until your title is complete.

Average {Variable Select} of {Drivetrain Parameter} Wheel Drive {Type Parameter}s made in {Origin Parameter}

Figure 17. How the dynamic text box should look in Text Box Edit Mode.

Click through values in the controls within the prompt container and watch the title dynamically change.

DYNAMIC TEXT REPORTS

The dynamic title concept can be further expanded to full text reports that change dynamically based on filter values. Sometimes people like a good executive summary rather than graphs, and those can be tedious to enter on a regular basis. You can save yourself time by having Visual Analytics crunch the numbers and have the text already prepared for you. Parameters and measures can live together harmoniously in text boxes for this purpose.

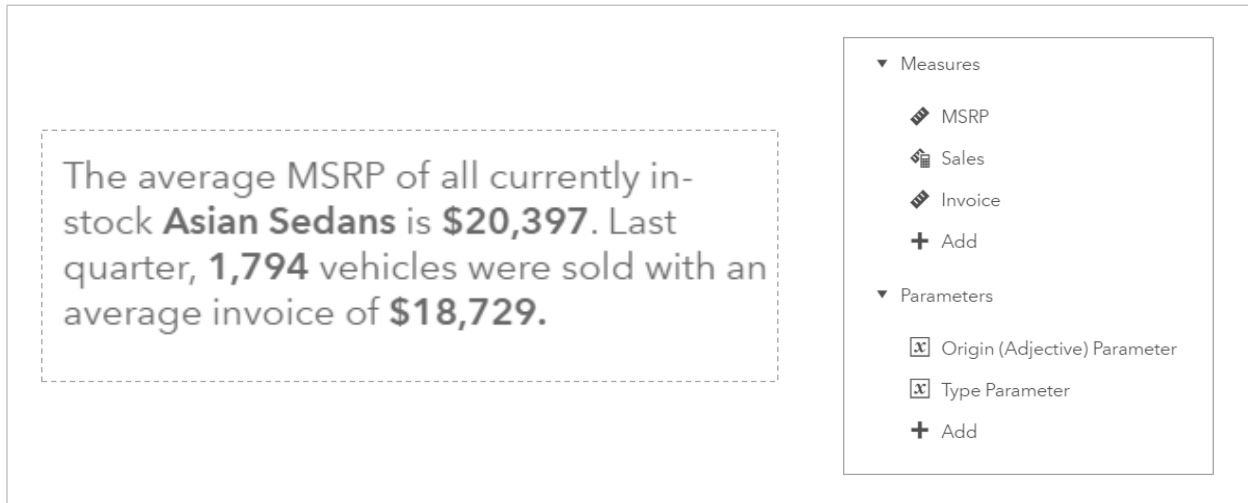


Figure 18. Combining measures and parameters to create a text summary.

SIMPLE WHAT-IF CALCULATORS

When reports are flexible and interactive, users become more interested and can get greater value out of them. One effective way is by creating simple calculators out of report controls that can let users make assumption-based scenarios. For example, a user can adjust values of a slider to see how a bank loan is affected by different interest rates. These types of calculators give users a wide range of assumption-based scenarios that can help them narrow down areas of focus and interest.

Continuing to expand on the report you made from [Dynamic Ranks](#), you've managed to convince your significant other to buy a high horsepower car, but only under the condition that you also get something fuel efficient to offset your carbon footprint. Recall that you settled with a used Nissan Sentra, but you *also* went out and leased a 2020 BMW M340i. You're about to go on your yearly three-week trip to visit your in-laws in Hays, Kansas. It's a long drive from Raleigh, North Carolina, and you'd like to estimate out how much it would cost you in fuel between the two vehicles.

You have a few different routes in mind that can make the trip more interesting. If you take the BMW, you've got nearly 400 horsepower at the command of your right foot, so you'll want to take a route that includes the Tail of the Dragon: one of the most famous curvy roads sought after by motorcyclists and car enthusiasts. If you take the Sentra, you'll want to drive on the straightest, flattest interstate you can find. To make it easier to review these different ideas, you want to add a tab to your Visual Analytics report with a fuel cost calculator to help estimate how much it will cost you to drive using various routes that you come up with.

To figure this out, you need to know a few things:

- Total distance
- % of total distance driven on the highway
- City MPG of your vehicle
- Highway MPG of your vehicle
- Average cost of fuel

Your desired metrics are:

- Gallons of fuel consumed
- Total cost of fuel

The two output metrics are based on formulas in [Table 2](#):

Gallons of Fuel Consumed	$\frac{Distance_{highway}}{MPG_{highway}} + \frac{Distance_{city}}{MPG_{city}}$
Fuel Cost	<i>Gallons of Fuel Consumed * Avg Fuel Cost per Gallon</i>

Table 2. Output formulas

The metrics won't tell you precisely how much you'll spend on fuel, but it's in the ballpark and is a starting point to decide which car to take. You want to be able to use some report controls to be able to vary these values and see how they affect your output metrics.

Remember that parameters can be used in calculated items, and only measures or categories can be displayed in most report objects. Since you can create your formulas as calculated items in any data set, the most efficient way to calculate them is by using the smallest data set possible. At 25 observations, *va_dummy_data_body* is a good option. Note that in Visual Analytics, a calculated item is applied to every row. Since you have 25 rows in *va_dummy_data_body*, your formula will be repeated 25 times. Changing the aggregation method from *sum* to *average* will return the correct value.

STEPS TO CREATE REPORT

1. Create five new numeric parameters with the options in [Table 3](#):

Parameter Name	Min	Max	Current Value	Format	Decimals
Total Distance	0	10,000	1	Comma	0
% Highway Miles	0	1	1	Percent	0
City MPG	0	75	1	Best	0
Highway MPG	0	75	1	Best	0
Avg Fuel Cost	0	5	1	Dollar	2

Table 3. Parameter options for calculator inputs.

2. Create two new calculated measures in *va_dummy_data_body* with an aggregation of *average*.

- a. **Gallons**

```

('Total Distance'p * '% Highway Miles Parameter'p) / 'Highway MPG
Parameter'p
+
('Total Distance Parameter'p * (1 - '% Highway Miles Parameter'p) ) /
'City MPG Parameter'p

```

- b. **Fuel Cost**

```
'Gallons'n * 'Price/Gallon Parameter'p
```

3. Add report controls to the body of your report such as those in [Figure 19](#) and [Figure 20](#), and attach the appropriate parameters to them; these will be the values you can adjust for your calculator.
4. Add objects to visualize the *Gallons* and *Fuel Cost* measures.

USING THE CALCULATOR

It's 1,334 miles one way from Raleigh, NC to Hays, Kansas. You estimate that you'll be on the highway about 75% of the time, and you *really* want to take the BMW. Your 2020 M340i gets an average City and Highway MPG of 20 and 30, respectively. Since it has a high-performance turbocharged engine, you're stuck with using premium 93 at an average of \$2.80 per gallon.

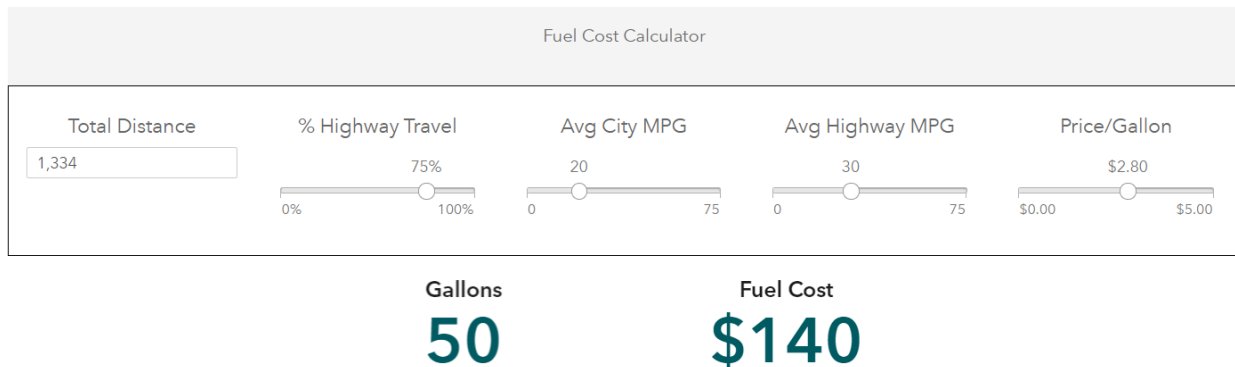


Figure 19. Estimated fuel cost to drive a 2020 BMW M340i from Raleigh, NC to Hays, Kansas.

Next, you look at the humble 2017 Nissan Sentra. While less fun, it is more economical. The Sentra gets 29 city/37 highway MPG and only needs to use 87 at the pump, costing an average of \$2.21/gallon.

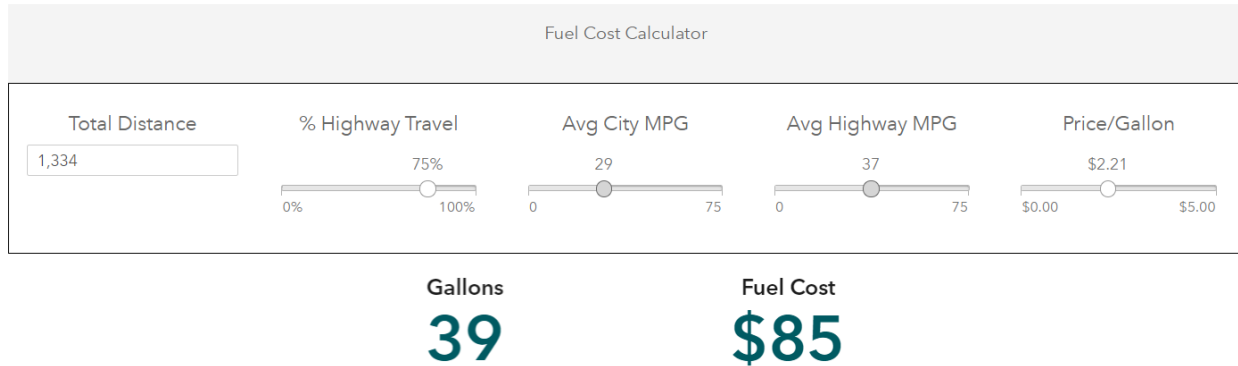


Figure 20. Estimated fuel cost to drive a 2017 Nissan Sentra from Raleigh, NC to Hays, Kansas.

Using the Sentra, you'll save around \$110 on fuel costs round-trip. In the end, you take the Sentra since the lease on your BMW only lets you drive 12,000 miles per year anyway.

DYNAMIC MAXIMUMS WITH DROP-DOWN LISTS

In forecasting, being able to review how forecasts change over time is paramount to ensuring quality forecasts. Visual Analytics makes it easy to review forecast start dates over time. With wide, regular time intervals, sliders are a great option to easily let users explore data. For infrequent or granular time intervals, drop-down lists are a better option. A drawback to using drop-down lists is the inability to have a dynamic maximum value (as of this report). Using parameters, you can add that functionality.

LIMITATIONS OF SLIDERS

In early versions of Visual Analytics, the value of a slider last saved in Designer mode was the default value every time the report was opened. For example, consider a forecast report that updates monthly. A user can use a slider to move back and forth between forecast start dates to review historical forecasts. When the report is first created, the forecast start date is set to its maximum by the designer. The next month the forecast updates, the forecast start date in the report remains at the previous value.

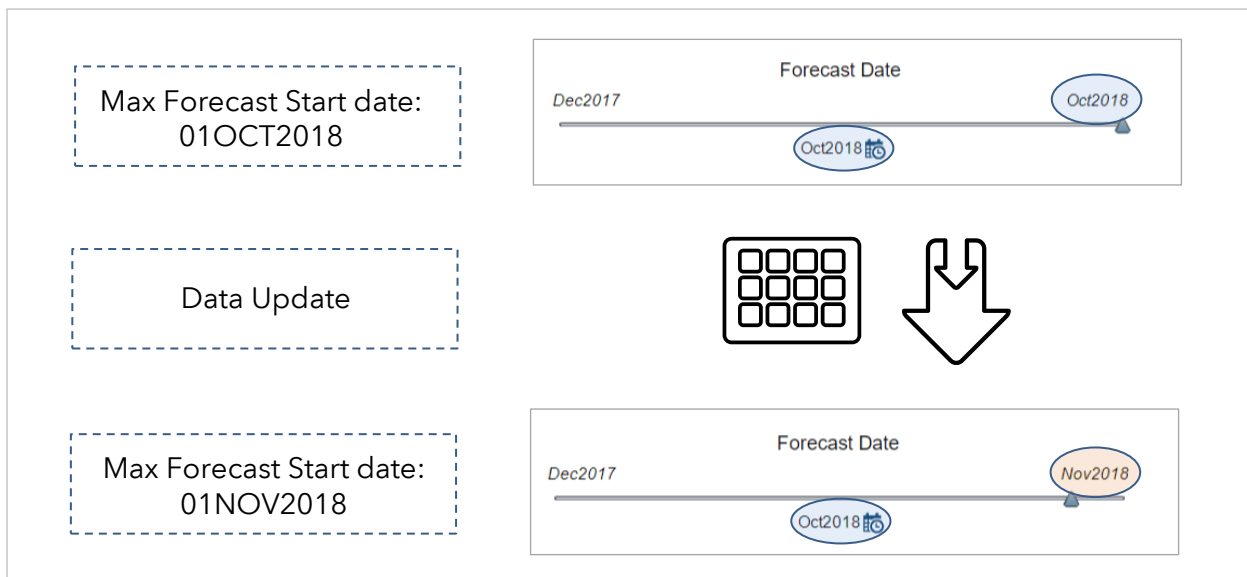


Figure 21. Static slider values in Visual Analytics 7.3 and earlier.

The solution was to open the report every time the data updated and manually change the slider, and then resave it. Versions 7.4 and 8.x introduced the "Set value to dynamic minimum" and "Set value to dynamic maximum" options for sliders to resolve this issue.

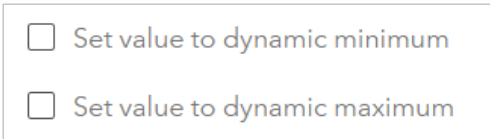


Figure 22. Dynamic slider minimums/maximums in Visual Analytics 7.4 and 8.x

Sliders increment in equal, standard time intervals, making them great for data that is updated on a regular wide time interval, such as weekly, monthly, yearly, and so on. Unfortunately, data is not always updated regularly, nor is it always at such a wide interval. Sliders can be difficult to use with dense data containing granular time periods such as day and hour. Also, any time gaps within the data are still able to be selected by the user, allowing users to select any invalid dates that exist. For example, if a forecast updates every *two* months rather than every month, selecting every other month on the slider will create errors in the report.

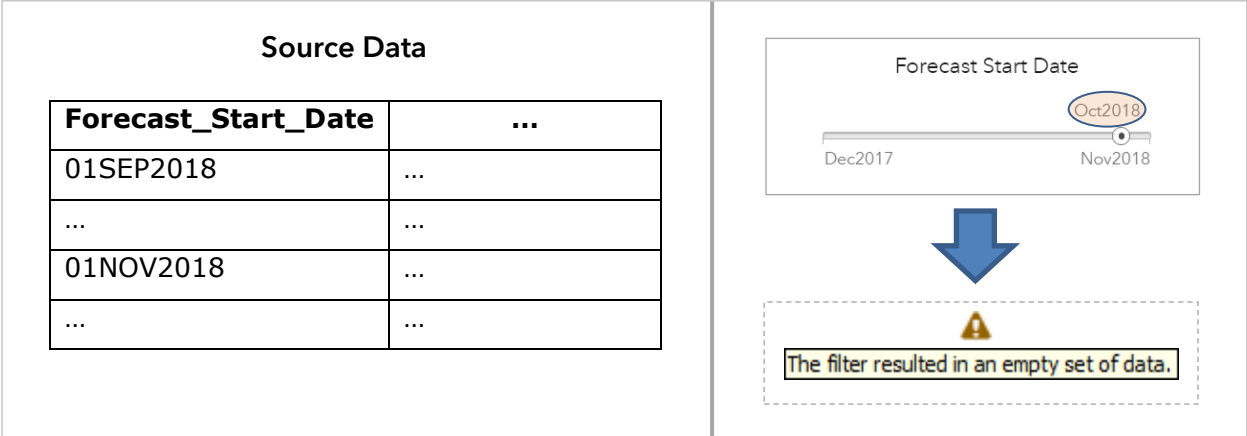


Figure 23. An invalid filter value can still be selected with a slider.

Sliders include all possible dates between a range, which can make it ambiguous as to which forecast dates are valid or invalid. One solution would be to use a drop-down menu, which will show the user only valid values; however, there is no built-in option for a dynamic minimum or maximum value, bringing you back to the original problem. With some minor source data changes and with parameters, you can add this functionality.

ADDING DYNAMIC MAXIMUM FUNCTIONALITY TO DROP-DOWN LISTS

There are three steps to add dynamic maximums to drop-down menus:

1. Create a binary flag in your data identifying the maximum forecast start date.
2. Add a date parameter to the drop-down menu.
3. Add logic to every report object on the page to display the maximum forecast date when nothing is selected in the drop-down list.

Step 1: Add a Forecast Start Date Flag

You will need to know which forecast start date is the most current. One way is to add a binary flag whenever the data is updated. An example of this is shown below.

```

/* Read in current forecast and add in a forecast start date. */
data current_forecast;
    set outfor;

    forecast_start_date = today();
run;

/* Append the current forecast to all previous forecasts.
Add a flag to tell us when the date of forecast is the most recent */
data all_forecasts;
    set previous_forecasts
        current_forecast
    ;

    flag_max_forecast_start_date = (forecast_start_date = today() );
run;

```

forecast_start_date	flag_max_forecast_start_date	...
01JAN2008	0	...
...
01OCT2018	0	...
01NOV2018	1	...

Figure 24. Creating a binary flag to identify the most recent forecast start date.

Step 2: Create a Date Parameter

Create a new date parameter and add it to your drop-down list. Give it a minimum value at least as low as the earliest forecast date, and a maximum value well above your maximum forecast date. Set the current value to any value.

The image shows two side-by-side screenshots from the Power BI interface. The left screenshot shows the configuration for a new date parameter. The 'Name' field is 'Forecast Start Date Parameter', the 'Type' is 'Date', and the 'Multiple values' checkbox is unchecked. The 'Minimum value' is set to '01Jan2008' and the 'Maximum value' is set to '01Jan2038'. The 'Format' is 'DATE9 (Date with Month Name)' and the 'Current value' is '01Nov2018'. The right screenshot shows the 'Data Roles' section with a dropdown menu set to 'Forecast Start Date'. Under the 'Parameter' category, 'Forecast Start Date Parameter' is listed with a blue arrow pointing to it, indicating it has been selected.

Figure 25. Creating a date parameter and adding it to a drop-down list.

Step 3: Add Object Filter Logic

For every object on the page affected by the drop-down list, add the filter in [Figure 26](#):

```
if(missing('Forecast Start Date Parameter') )  
    return 'flag_max_forecast_start_date'n = 1  
    else 1=1
```

Figure 26. Object filter to force the maximum forecast date to display when no filter is selected.

The normal behavior of clearing a filter is to show every value. Now if you clear all filters in your forecast start date, it will only show the most recent valid forecast date.

HOW IT WORKS

When the drop-down list has no selection, the value of *Forecast Start Date Parameter* is missing. This is the only time the parameter will have a missing value. When it has a missing value, you want all objects on the page to filter to the most recent forecast start date: that is, when `'flag_max_forecast_start_date'n = 1`.

You don't want the object filter to do anything else when a forecast start date is selected. In Visual Analytics, if/then logic requires an `else` condition. To get around this, you can return an always-true condition such as `1=1`. This makes the `else` statement do nothing, preventing any additional filtering at the object-level.

In other words, when you don't select a forecast start date, the object filter returns the most recent forecast start date. When you do select a forecast start date, the object filter does nothing. [Table 4](#) shows how this logic is applied.

Drop-down List	Parameter Value	Report Control	Object Filter
Forecast Start Date ▾	(missing)	None	<code>'flag_max_forecast_start_date'n = 1</code>
Oct2018 ▾	'01OCT2018'd	'01OCT2018'd	None

Table 4. Drop-down list filter and object filter behavior.

If you are using Visual Analytics 8.3+, the process of creating individual object filters is greatly simplified thanks to the Common Filters feature. For more information, see [Using Common Filters](#) in the Visual Analytics users guide.

FINALIZING YOUR REPORT

Adding a dynamic title showing the selected forecast date in big, bold text is a good way to improve your report. You might be tempted to use the value of the parameter, but that won't be helpful this time. If the parameter is put into a text box, it will show `<No item selected>` until a forecast start date is selected.

A workaround is to create two new calculated measures that get the month and year from the forecast start date and set their aggregation to average. An example report of everything put together is shown in [Figure 27](#).

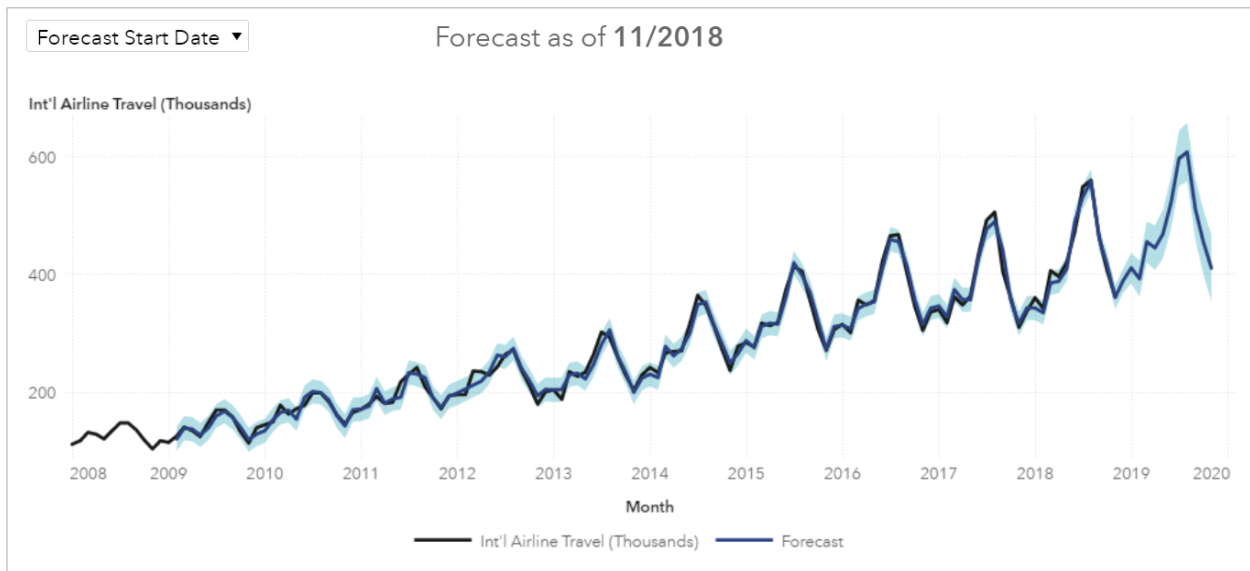


Figure 27. Example report displaying user-selectable forecast start dates.

A NOTE ON MULTI-VALUE PARAMETERS

Parameters can support holding multiple values, such as in multi-selection list controls. When using parameters in this way, the number of available functions that can act on them is limited. Only the `In`, `NotIn`, and `IsSet` operators can work with multi-valued parameters. For example, consider the multi-valued parameters in [Table 5](#):

'Char List'p DOG,CAT,BIRD	<p>✗ <code>if('Char List'p = 'DOG' OR 'Char List'p = 'CAT')</code></p> <p>✓ <code>if('DOG' In('Char List'p) OR 'CAT' In('Char List'p))</code></p>
'Num List'p 1,2,3	<p>✗ <code>if('Num List'p <> 1)</code></p> <p>✓ <code>if(1 NotIn('Num List'p))</code></p>

Table 5. Example of using multi-valued parameters.

For more information about multi-valued parameters, see the [SAS Help Documentation](#) on Parameters.

CONCLUSION

Parameters are powerful tools in Visual Analytics that vastly expand what is possible within a report. The five examples presented in this white paper were selected to help you understand effective ways in which parameters can interact with a report. There are many other ways parameters can be used outside the scope of this paper. For example, if you are savvy with custom polygons in Visual Analytics, you can create a special polygon data set that holds multiple layers that can be selected by using parameters for dynamic geographic variables. Another example would be using parameters to feed inputs into Visual Statistics objects to vary forecast inputs, regression models, decision trees, and more. By mastering parameters, your report will become more flexible and feature-rich, allowing you to reach a wider audience who can get the information they need from fewer, simpler tabs.

REFERENCES

SAS® Institute. "Working with Parameters in Reports." February 21, 2019. Available at <https://documentation.sas.com/?docsetId=vareportdata&docsetTarget=n1wv50n60ccq86n1nzp6zat1wj64.htm&docsetVersion=8.3&locale=en>

First Impressions: Making Up Your Mind After a 100-ms Exposure to a Face.
J Willis-A Todorov - <https://www.ncbi.nlm.nih.gov/pubmed/16866745>

ACKNOWLEDGMENTS

Thank you to my colleagues Kevin Baughman and Ryan Story for your inspiration to write this report. You are brilliant analysts who I am proud to work with.

Thank you to my mentor, Jared Peterson, who encouraged me to write my first white paper, and for always trying to get me to push myself further. You are a phenomenal leader and role model who I highly look up to.

Thank you to my manager, Ned Maran, who has supported my professional development and growth since day one. This report would not have been possible without your support.

RECOMMENDED READING

SAS® Visual Analytics 8.3 Documentation: Working with Report Data
<https://documentation.sas.com/?docsetId=vareportdata&docsetTarget=titlepage.htm&docsetVersion=8.3&locale=en>

Using Parameters in SAS® Visual Analytics
<https://blogs.sas.com/content/sgf/2015/01/29/using-parameters-in-sas-visual-analytics/>

CONTACT INFORMATION

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

Robert (Stu) Sztukowski
SAS Institute Inc
(919) 531-3238
Stu.Sztukowski@sas.com
<https://www.linkedin.com/in/StatsGuy>

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.