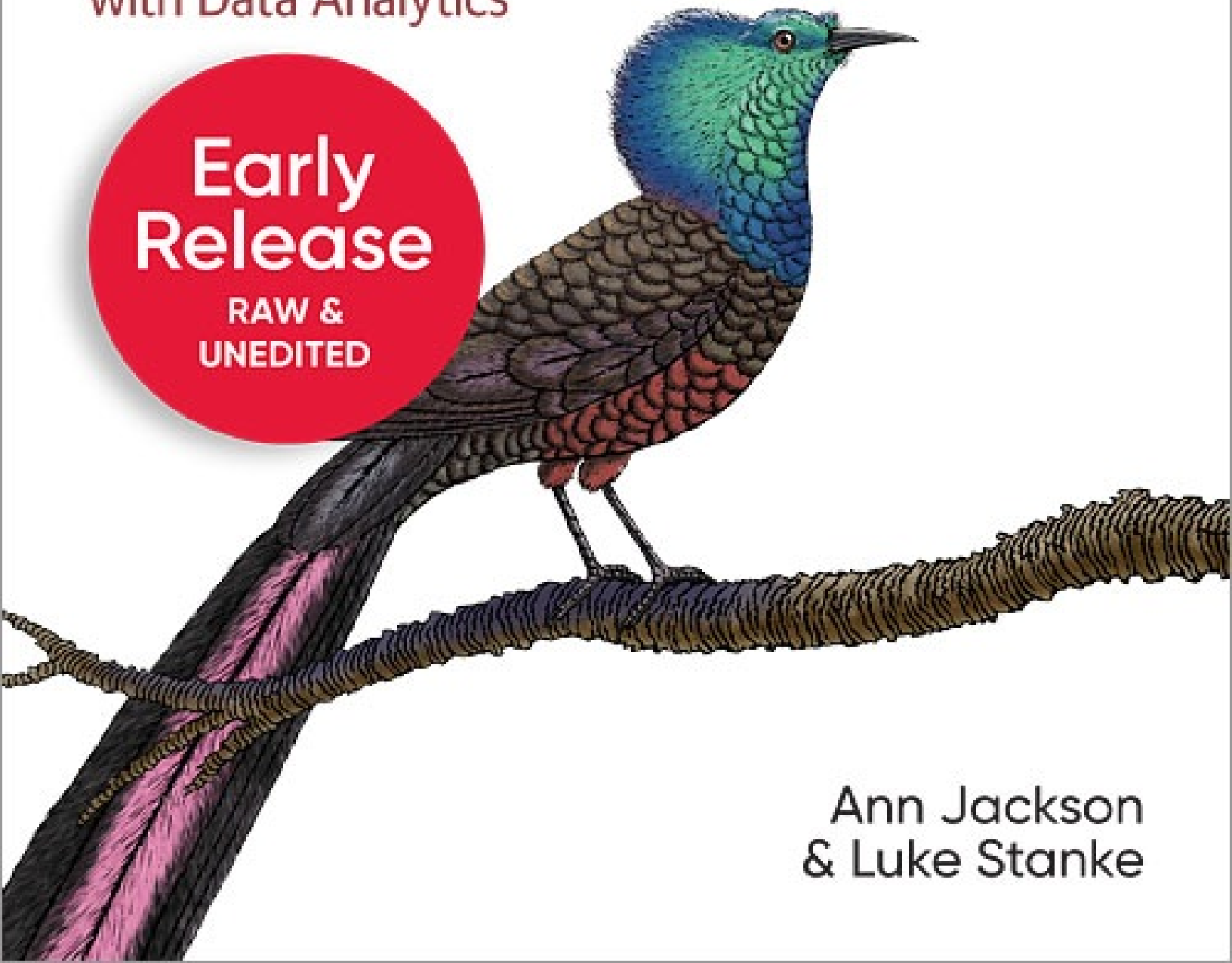
O'REILLY®

Tableau Strategies

Solving Real, Practical Problems with Data Analytics

Early Release
RAW & UNEDITED



Ann Jackson
& Luke Stanke

Tableau Strategies

Solving Real, Practical Problems with Data Analytics

Ann Jackson and Luke Stanke

Tableau Strategies

by Ann Jackson and Luke Stanke

Printed in the United States of America.

Published by O’Reilly Media, Inc. , 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O’Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://oreilly.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com .

Editors: Michelle Smith and Sarah Grey

Production Editor: Caitlin Ghegan

Copyeditor: FILL IN COPYEDITOR

Proofreader: FILL IN PROOFREADER

Indexer: FILL IN INDEXER

Interior Designer: David Futato

Cover Designer: Karen Montgomery

Illustrator: Kate Dullea

May 2021: First Edition

Revision History for the First Edition

- YYYY-MM-DD: First Release

See http://oreilly.com/catalog/errata.csp?isbn=9781492080084 for release details.

978-1-492-08008-4

[FILL IN]

Chapter 1. Categorical Analysis

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 1st chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the authors at ann@jacksontwo.com and luke.stanke@tessellationconsulting.com.

Categorical analysis is the foundation of data visualization. It is the first and most frequent type of data visualization data analysts use. Categorical analysis takes a dimension (example: Regions) and breaks apart by a measure (example: Sales). A dimension is typically a categorical value; these do not get aggregated. They are likely used to create data headers or to generate filters. A measure is a (usually numerical) value that can be aggregated using mathematical functions (like sum, average, or median). Measures create unbroken axes, those that extend from one end of a range to the other.

This type of analysis aids in answering common business questions such as:

- How does A compare to B?
- How is X measure distributed across Y categories?
- How much do A, B, and C contribute to the total?

- How does X measure change over time (where time is the dimension)?

Categorical analysis is usually presented as bar charts. *Bar charts* use height or length as visual encoding to express a measure. *Visual encoding* refers to different techniques for displaying data in charts. Encoding data in bar charts is very effective because humans can quickly analyze the variation among the size of the bars; they are also very easy to understand and label.

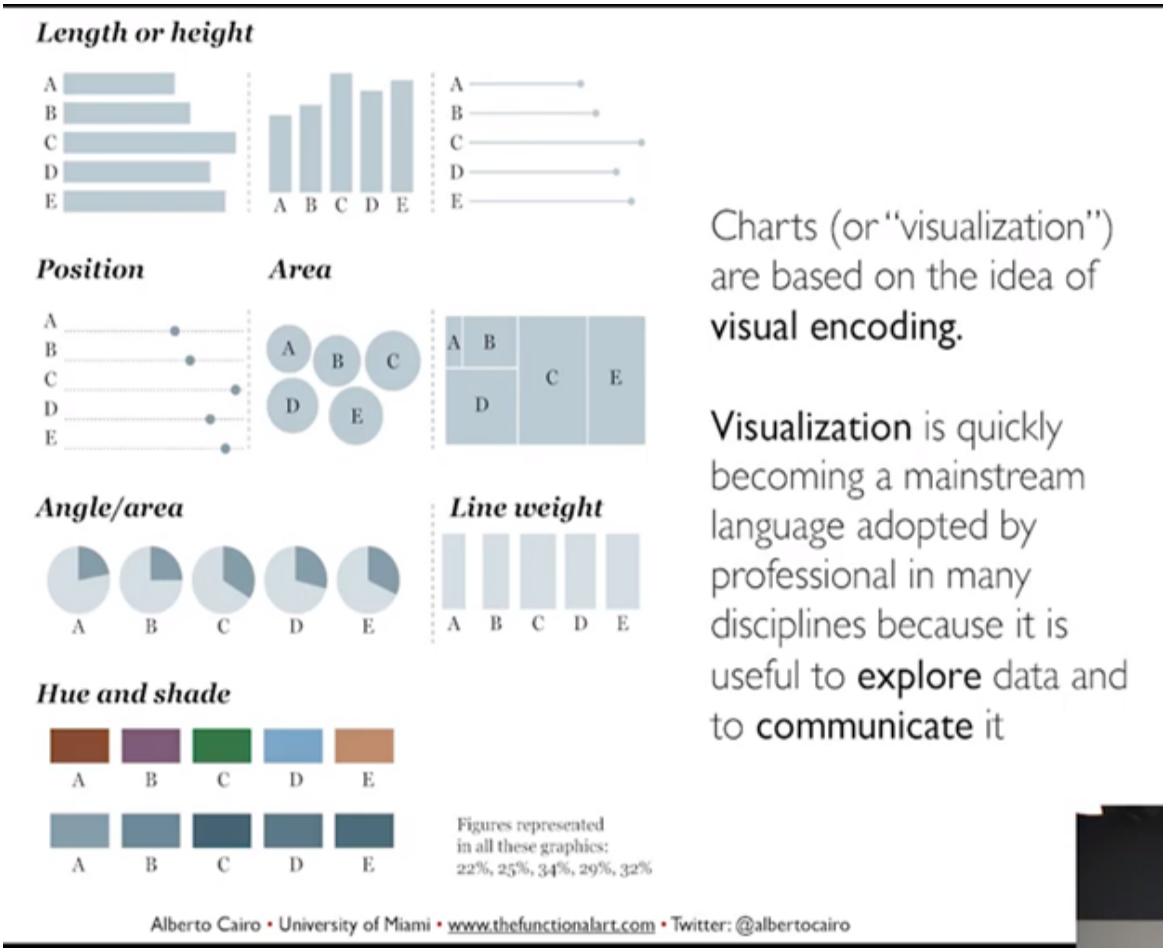


Figure 1-1. This illustration shows the various ways that data can be encoded for display. It aligns them to a comprehension scale of how precise the human eye can discern differences.

In our first use case, we will explore how to make effective **bar charts.** You'll play the role of a large financial institution that wants to understand which merchant categories make up the majority of transactional spend in order to drive marketing efforts and partnerships and better serve

customers’ interests. We will also expand from the defaults and learn two additional methods for making bar charts that demonstrate the most important information.

In the second case study, on treemaps, you’ll learn about working with many dimensions. While bar charts are very useful, there are other data visualization tools you’ll need to leverage when doing categorical analysis. Sometimes when there are many members of a dimension, it becomes problematic to display each member as a bar chart. When this happens, you can use alternative chart forms to conserve space but still display all members. The most useful chart for this scenario is a treemap. Here you’ll play a non-profit organization that controls and awards grant money for Creative, Performing, and Cultural Arts Programs and Initiatives for the state of New York.

In our final use case within the chapter, you’ll learn how to use pie charts and donut charts to visualize whole relationships. Here, you are conducting a survey about IT professionals, their employment, their mental health and their employers’ attitudes toward mental health. Often these are the first type of data visualization you learn in school, but we like to use them sparingly and as an alternative option. By the time we get to this use case, you’ll see how properly executed pie charts can be great tools to craft and share data with your audience.

What You'll Learn in This Chapter

- Create compelling bar charts that work dynamically to display top contributor information and also those that can automatically group together dimensions of small values
- Understand when to utilize bar charts vs. treemaps when faced with a dimension of several members. Utilize drill-down features within TreeMaps to explore tiered dimensions. Leverage additional data encoding by way of color to express alternative information.
- Utilize pie charts to demonstrate part-to-whole relationships. Turn pie charts into donut charts that communicate multiple data points. Utilize small multiple charts to do dual dimension comparisons.

Bar Charts

Bar charts should be the first visualization type you try when exploring categorical analysis. Because they use length and height as visual encoding, they make it easy to interpret and compare members.

If you have many members of a dimension and only need to focus on top contributors, you can create a Top N bar chart that uses a parameter, or dynamic entry value, to limit the amount of data in the view. Since a parameter is defined by the audience, it can make customized charts that suit the audiences' preferences.

To enhance bar charts further, you can also create percent- of- total calculations and dynamically combine small members. This is useful when it's important to see all the data in a chart, but you need to focus on large contributors. We'll walk through how to create a Set that changes based on a parameter.

As you're considering building bar charts, or really any chart, with dynamic features, we recommend adding additional visual cues, like reference lines and color. These small formatting elements will help the audience

understand aid in the audience's cognition of the dynamic elements and reinforce how their input impacts the view.

Bank Case Study: Visualizing consumer transaction data with a bar chart

Imagine you work for a large financial institution and are trying to understand consumer behaviors. Your goal is to understand how and where consumers spend their money. This objective is fundamental to the organization's success because it will drive the direction of marketing efforts, partnerships, and product promotions. It will also provide insight into profiling customers and may even unearth some opportunities to grow the customer population.

To solve this problem, you're going to start with a basic bar chart. It will help you compare types of merchants by how much consumers spend. It will also serve as the first step in understanding the data we have available.

Blueprint: Building a bar chart in in Tableau:

1. Drag on **Merchant Category** as a dimension, put it on rows.
2. Drag the measure **Transaction Amount** onto the column shelf as the SUM(**Transaction Amount**).
3. Sort the merchants using the toolbar in descending order by Spend.

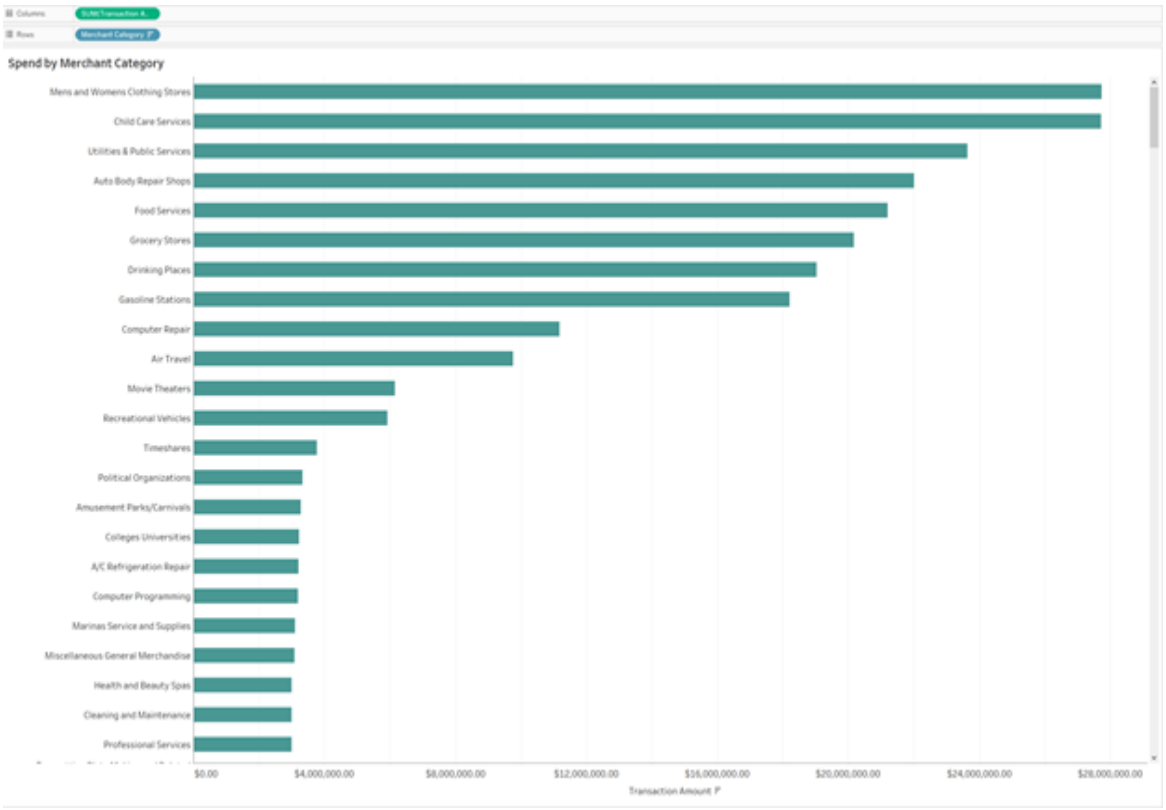


Figure 1-2. A bar chart showing the Merchant Categories sorted in descending order by Transaction Amount

Congratulations: you've built your first bar chart. Now this may not seem revolutionary, but it is the first step in finding out where consumer spend is focused. This simple chart dispels any intuition-based theories and presents us with the facts.

This is a great starting point. We've now visualized the data and can see that the largest merchant category is "Men's and Women's Clothing Stores." We can also see that there are several small merchant categories that aren't responsible for a lot of spend like "Landscaping Services." Knowing that there are many types of merchant categories, some are much larger than others, is additional insight we can act on to improve our visualization.

A good incremental way to do this this would be to limit the number of visible merchant categories by using a Top N filter with a parameter. A *Top N filter* limits the chart by N, the number defined, as the top members who will comprise the chart. A *parameter* is a dynamic entry field defined by the end user. In this scenario, we will create a parameter that allows the user to

dynamically define what number of categories they want to see. Not only does adding the parameter give our audience more control over the visualization, but it also provides them with a more conversational way to understand what is in the chart. A “Top 10” is a much more tangible and bite-size takeaway than a long list of bars.

Blueprint: Creating a Top N Bar Chart:

1. Drag on **Merchant Category** to the Filter shelf
2. Navigate to the “Top” section
3. Select “create new parameter” from the dropdown
4. Name the parameter “Top N” and save it
5. Right click on the parameter in the lower left and expose the parameter control

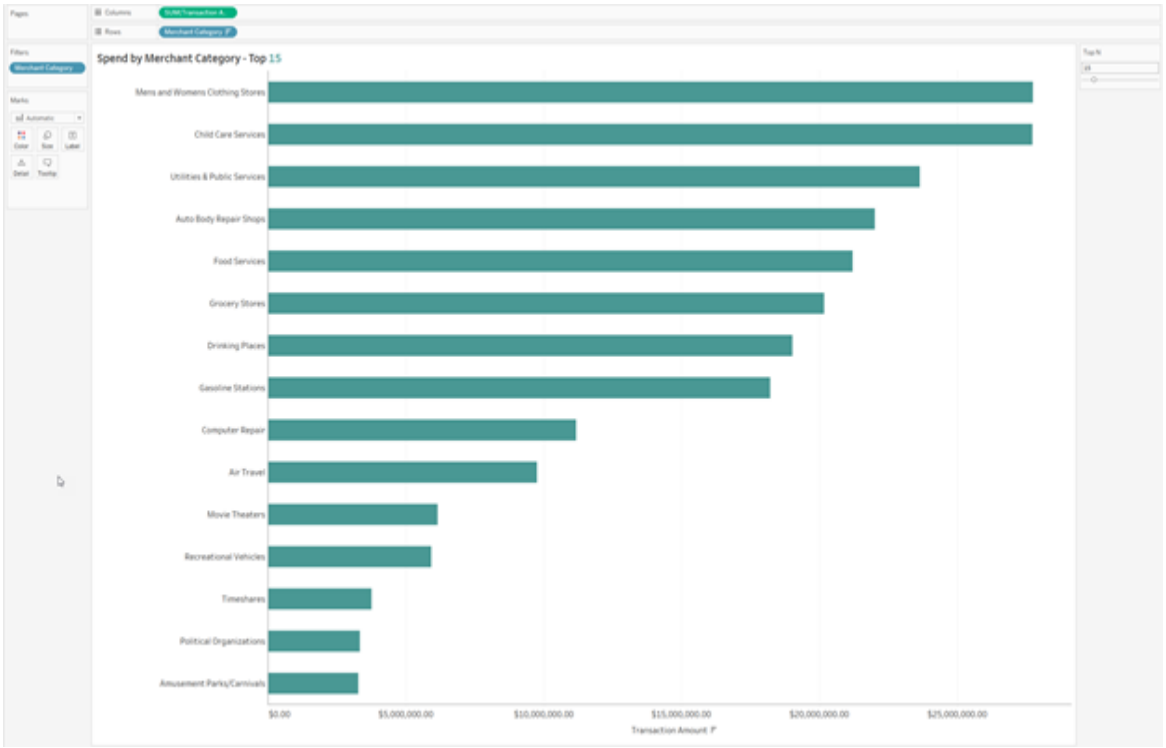


Figure 1-3. A bar chart with Top N filtering and a parameter applied

Notice in the screenshot that we've put the parameter value in the title of the chart. Now, when the user makes a dynamic change, the number will update in response. With this small act, we've now created a portfolio of charts that can be customized to suit the audience's needs. It also makes the chart feel responsive to the audience; their action changes the visualization.

So how can we take this one step further? We're guessing that you've been wondering what percentage of the total each of these categories represents. While it is useful to compare them as we try to synthesize our thinking further, a natural inclination would be to change the commentary that "Men's and Women's Clothing Stores makes up $27 million in customer spend" vs." close to 4% of all customer spend is attributed to Men's and Women's Clothing Stores."

How do we approach presenting this information? Well first, we can change the measure from the direct measure to the percent of total. But we're now left with the lingering notion that if we limit our chart to a Top 10, we'll lose the context of how much spend is in all the other categories.

To work around this constraint, we approached it by allowing the audience to utilize a parameter to define the proportion of customer spend that they want to see broken out into the merchant categories and to automatically group together all categories. They still have dynamic control over the chart, but are left with the full picture of data.

Notice that you'll need to construct a calculation that is equivalent to the percentage of total calculation table calculation you made. To do this, you can utilize level of detail expressions. This calculation takes the SUM(Transaction Amount) and divides it by the total SUM(Transaction Amount) for the entire data set.

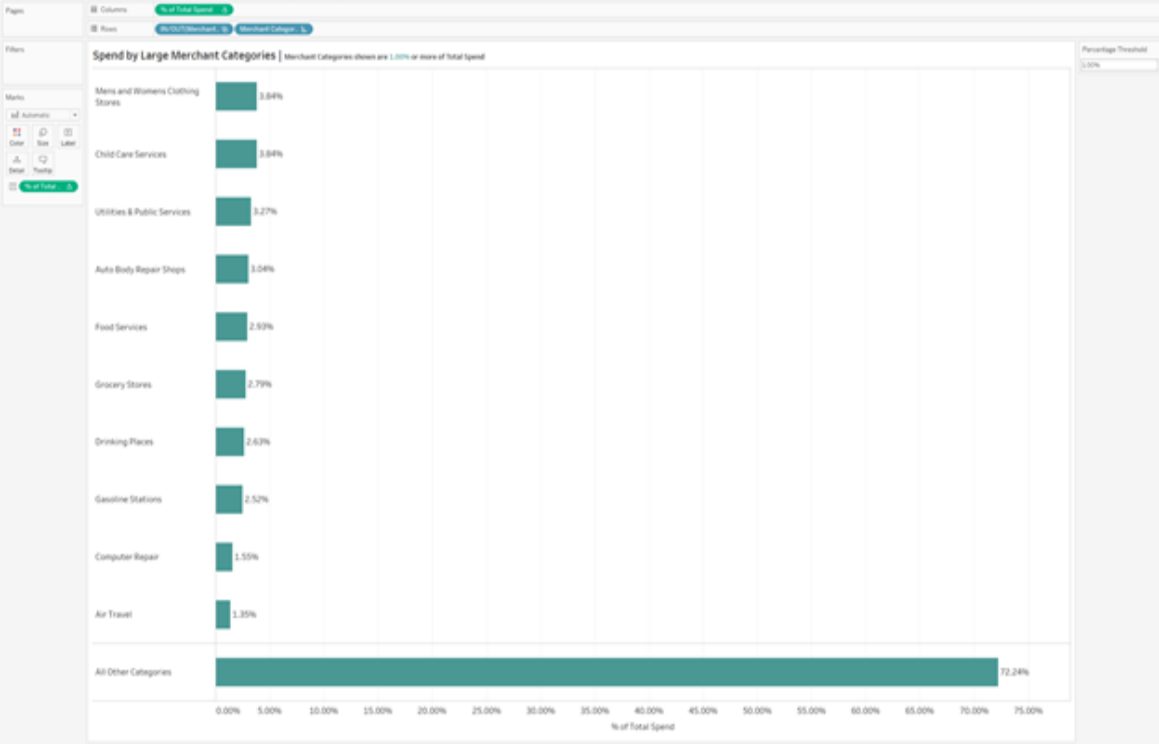


Figure 1-4. The updated bar chart, now with "All Other Categories" grouped at the bottom of the bar chart.

Blueprint: Dynamically group other dimensions in a bar chart

1. Remove the Merchant Category filter from the filter shelf.
2. Change the measure to a percentage of total using Quick Table Calculations. Right click on SUM(Transaction Amount) and select Quick Table Calculations -> Percentage of Total.
3. Create a float parameter with formatting set to percentage. Set the default value called **Percentage Threshold to 1%.**
4. Create a set based on the **Merchant Category** dimension. This will be a formula set based on the calculation that the percent of total is greater than or equal to the parameter

5. Right click on **Merchant Category** and select Create -> Set.
6. Select “Use all”, then navigate to Condition and enter the following in **By Formula**: SUM(**Transaction Amount**)/MAX({SUM(**Transaction Amount**)}) >= **Percentage Threshold.**
7. Create a calculated dimension called Merchant Category to Display. Use the formula IF [Merchant Set] then Merchant else “All Other Categories” END
8. Drag the new Dimension **Merchant Category to Display** on top of **Merchant Category** on the Rows Shelf.
9. Drag Merchant Set to the left of Merchant Category to Display. This will organize how the Categories are listed. Right-click and hide the header.

The updated analysis is much more flexible to the audience’s preferences. Now, they have contextual information about the percentage of total and input to determine how much data is shown. It is a step ahead of the bar chart with the sum of Spend, because we are no longer sacrificing knowing the total distribution of data.

If you’ve reached this point and still want more, there are some items we think can be introduced to add even more context and feedback to your audience. Similar to our original parameter for Top N, these additional techniques will serve as feedback to the audience on how the chart reacts to their input, and it also helps to enhance their trust in the chart.

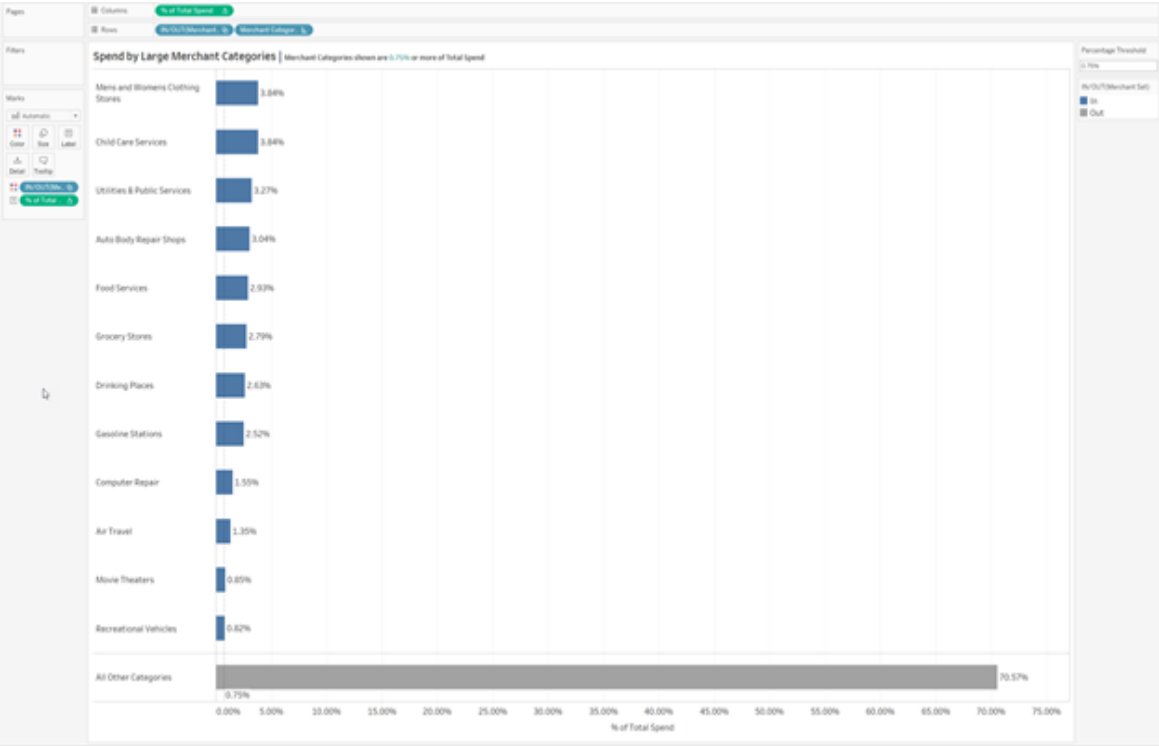


Figure 1-5. The same bar chart, now with color encoding to distinguish between the individual categories and the grouped category.

Blueprint: Enhancing your bar chart with color:

1. First you can use the parameter as a reference line to reinforce the concept of dynamic entry. Right click on the % of Total Spend axis and Add Reference Line. Set the Scope to “Entire Table”, the value to “Percentage Threshold”, and the Label to “Value”. Click OK.
2. Now adjust the Percentage Threshold to 0.75%. Notice that additional Categories display, but none are less than 0.75%
3. You can also further encode the target large Merchant Categories by utilizing our Set for Color. Drag **Merchant Set** onto the Color Shelf. Those in the Set will appear as one color, while those not in it and part of “All Other Categories” will be another color

We love bar charts - they are critical tools for any analysis. While they can start out very simple, through abstracted metrics and dynamic entries, you can take a bar chart from basic to amazing.

Sometimes the value is in how you format text on and around those bar charts. In this section we discuss ways to spice up your bar charts to make them pop. Here are three more ways you can format your bar charts:

Things to note:

- Left-aligned text in bars
- Hidden header
- Labels different sizes and uses multiple spaces
- Retains a darker axis on rows

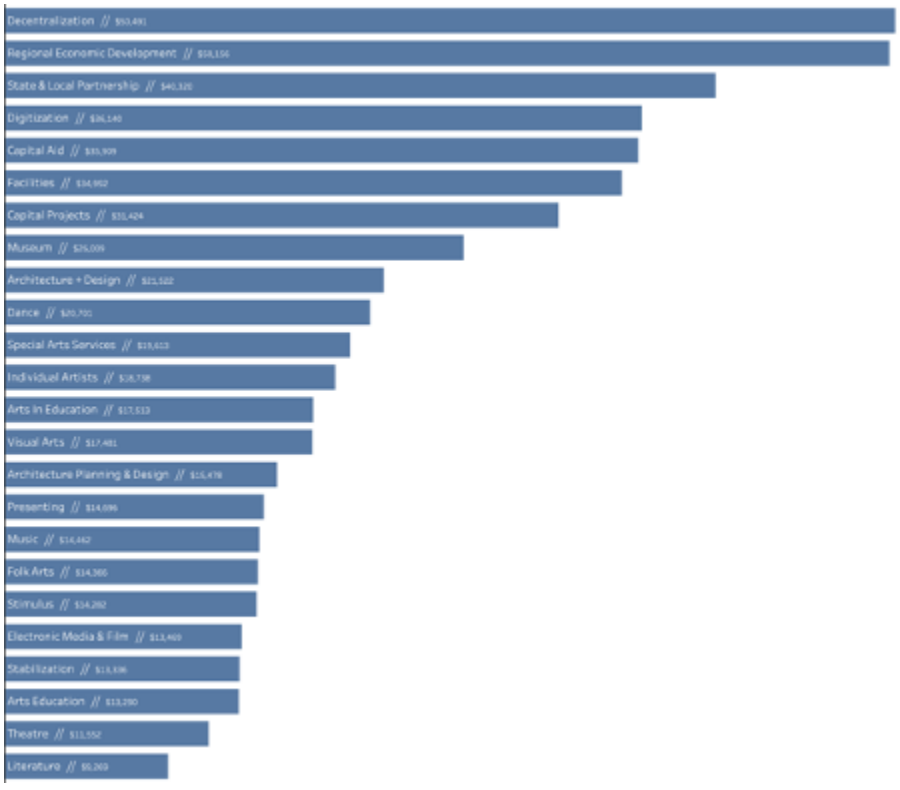


Figure 1-6. Reformatting a bar chart can bring new life to the chart type.

Things to note:

- Uses Dual axis on MIN(0.0),

- gantt mark type,
- max size gantt bar,
- 0% opacity
- left-aligned text
- Very small bar size
- Pay attention to whitespace between bars and text

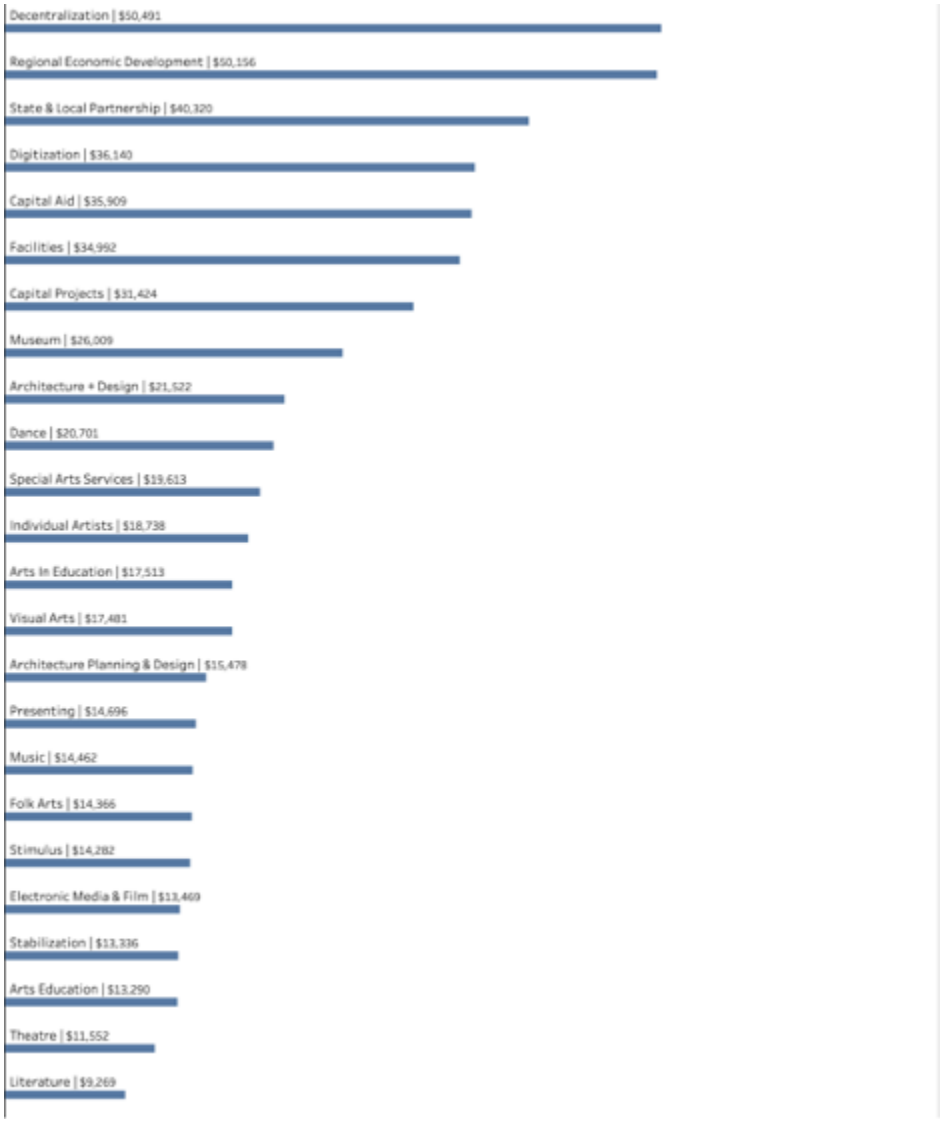


Figure 1-7. Another reformatted bar chart.

Things to note:

- Duplicate dot plot
- Change marks to both bars
- Set size of bar the same across both bar types
- Change color of WINDOW_MAX() mark to be a lighter version of the color associated with **[Avg Grant Amount]** marks.

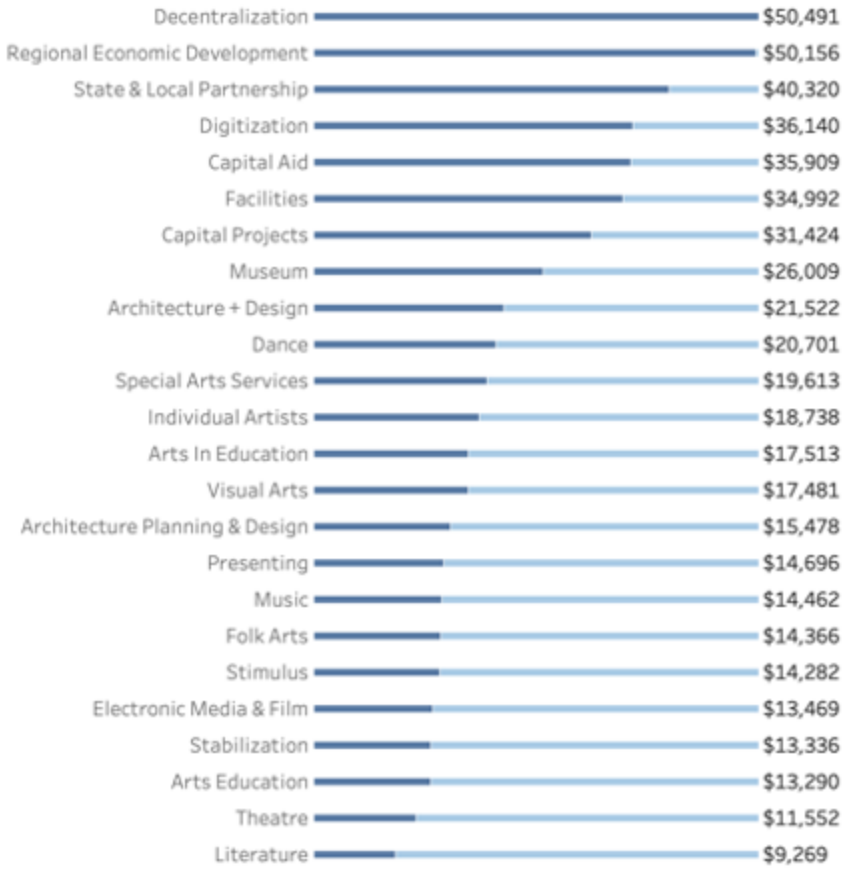


Figure 1-8. A proportional brushed bar chart can be created from a Cleveland dot plot.

Not all comparisons are going to require a simple bar chart. Sometimes the comparisons are more complex. For example, we might have to compare groups on a single metric but across two different time periods. When we are analyzing this information we want to understand across members–like we did with the bar chart–but also how individual groups have changed over time.

When doing this type of analysis we prefer to use a bar-on-bar chart. A common mistake we see with novice developers is using a side-by-side bar chart.

In Figure 1-9, we have the total grant size for 2018 and 2019 for each category.

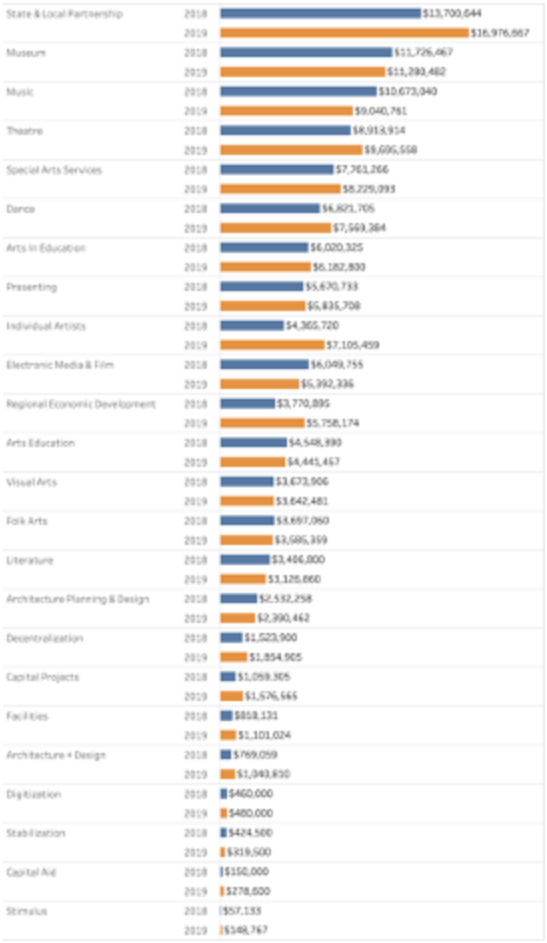


Figure 1-9. Side-by-side bar charts are effective but take up too much space.

When we look at this chart it looks like we have all the information we really want. We've got bars for two years and we can quickly compare. But as we dive into the sorting you'll notice that the data are not sorted on the total for 2019, rather they are sorted on the total across the two years. Additionally, any comparisons we make within a group, for instance, Arts Education, requires us to do mental math on the magnitude of the change from 2018 to 2019. It would be great to have that information on hand

directly on the visualization. It also might be worth being able to quickly note which categories were up from the prior year and which decreased.

We can do all of this using a bar-on bar chart.

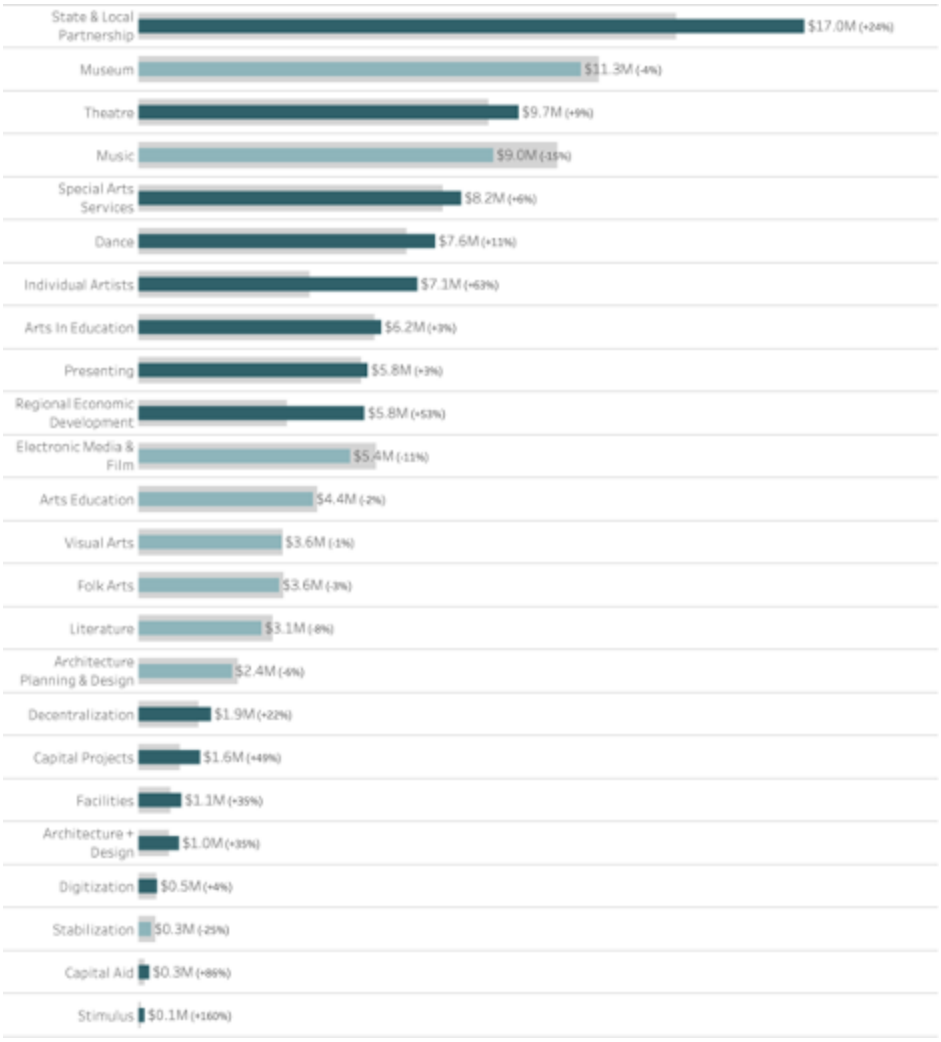


Figure 1-10. A bar-on-bar chart is a better alternative to the side-by-side bar chart—it takes up less space while displaying the same information.

With this chart type we've placed 2019 grant totals over the 2018 grant totals for each category. What's easy to note in this chart type is that your audience retains the ability to quickly compare groups across base year, in this case 2019, and the ability to compare how that group performed versus the previous year.

We aided in the year-to-year comparison by adding color–we didn't choose two distinct colors, rather we selected two colors with the same hue. For

totals below the prior year color is represented with a brighter, less saturated version of the color where totals are above the prior year.

Finally, we added the year-over-year change as a percentage next to the total for 2019. The final result is a chart that consolidates three comparisons: total grant dollars across groups for 2019, changes in total grant dollars from 2018 to 2019 for each group, and changes in magnitude from 2018 to 2019.

So how do you create this chart?

Blueprint: Creating a bar-on-bar chart

Step 1: Create your measures.

Instead of using a date dimension to partition your data, it's more effective to create two separate calculations that filter to the relevant data inside of the calculation. Let's create a calculation for grand amounts in 2018:

```
                // Grant Amount | 2018

                SUM(

                IF YEAR([Date]) = 2018

                  THEN [Grant Amount]

                  END
  )
```

And a calculation for grant amounts in 2019:

```
                  // Grant Amount | 2019

                  SUM(
```

```
IF YEAR([Date]) = 2019

            THEN [Grant Amount]

            END

            )
```

Generally speaking ,when working with, avoid hard-coding anything inside of calculations. In this case we'd normally use calculations or parameters to automate change as the data are updated. (We'll discuss this more in chapter 4.)

Additionally, if we were trying to build an optimized workbook, we'd skip using **YEAR()** for **DATEPART()**. For now, let's keep the calculations hard-coded they are.

Step 2: Create the base visualization.

Add **[Budget Category]** to rows and both **[Grant Amount | 2018]** and **[Grant amount | 2019]** to rows.

Step 2a. Create a synchronized dual axis chart where both mark types are bars. Be sure to place 2018 as the left-most dimension in the dual axis.

Change the bar size of 2019 to be narrower than the 2018 bars. You might have to adjust both to get your bars in a happy place.

Step 2b. Be sure to remove **[Measure Names]** from both cards. Note: you didn't add this, Tableau did this automatically when you created a dual axis chart.

Step 2c. Set the color on the outer bar to a light grey that is still distinguishable from the background.

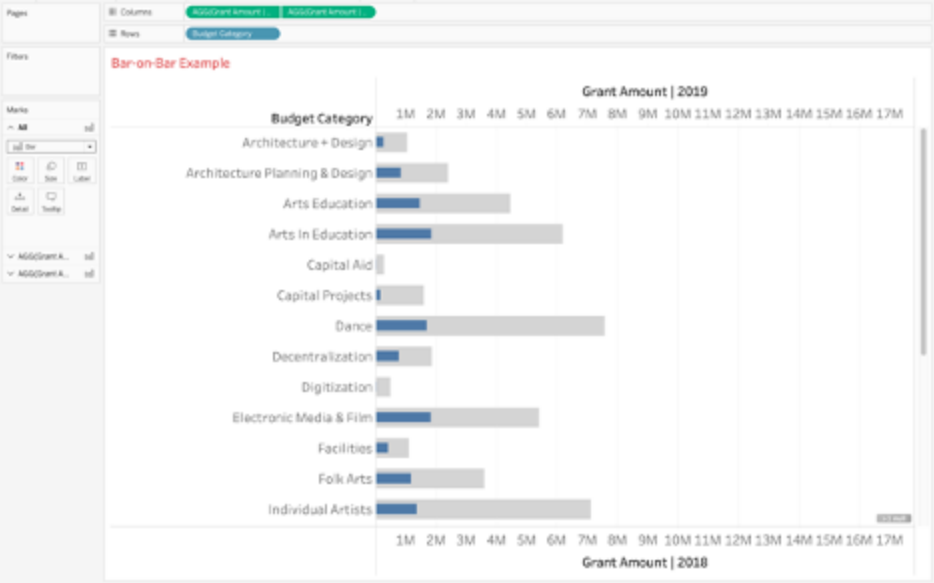


Figure 1-11. Use a dual axis with custom calculations to create bar-on-bar charts.

We prefer to make our outer bars (in Figure 1.x, 2018) equal to the same amount of whitespace between bars. For our inner bars we look to have the width between 50% and 75% of the width of the outer bar.

Step 3. Create labels

We could rely on the axes for comparisons, but because we are using a horizontal bar chart it makes sense to add labels.

Step 3a. On the **[Grant Amount | 2019]** marks card, click-and-drag **[Grant Amount | 2019]** to labels.

Step 3b. Next, create a new calculation called [Grant Amount | % Change] for the percent change from 2018 to 2019:

```
// Grant Amount | % Change

([Grant Amount | 2019] - [Grant Amount | 2018]) / [Grant Amount | 2018]
```

Step 3c. After you create the measure, right-click on the measure and change the default settings of the number format to:

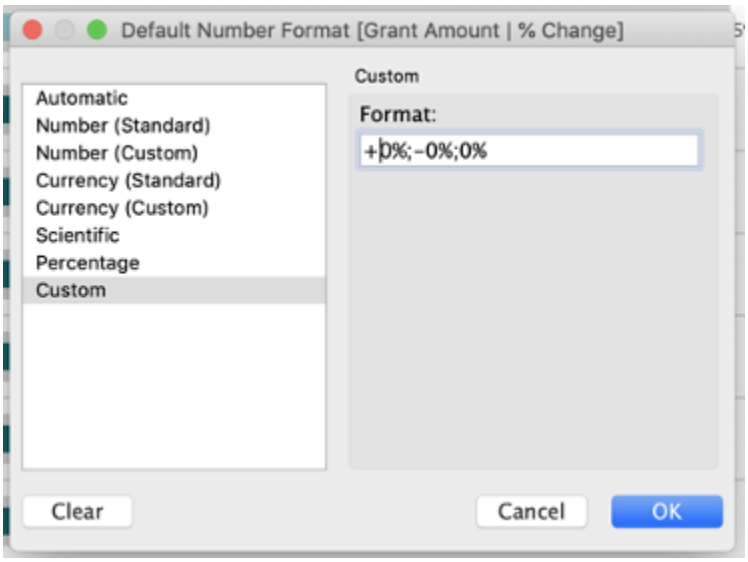


Figure 1-12. Use custom formatting to get your percentage displays just right.

This will display a plus sign in front of the positive values, a minus sign in front of the negative values and no direction when there is no change in the direction.

Step 3d. Add this calculation to Label, as well. Now edit the text of the label. Format **[Grant Amount | 2019]** to be both larger and a darker shade than the **[Grant Amount | % Change]** measure.

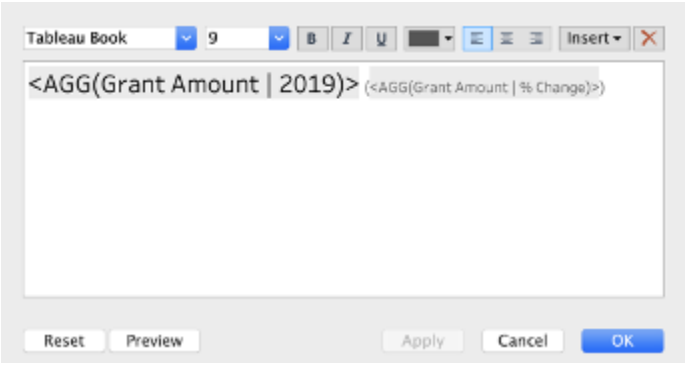


Figure 1-13. The text editor showing how you should format the text labels on the visualization.

For this example I'm using size 15, and size 9 fonts, respectively. Text colors are #000000 (black), and #555555 (dark gray), respectively. Additionally, I've added **[Grant Amount | % Change]** between parentheses.

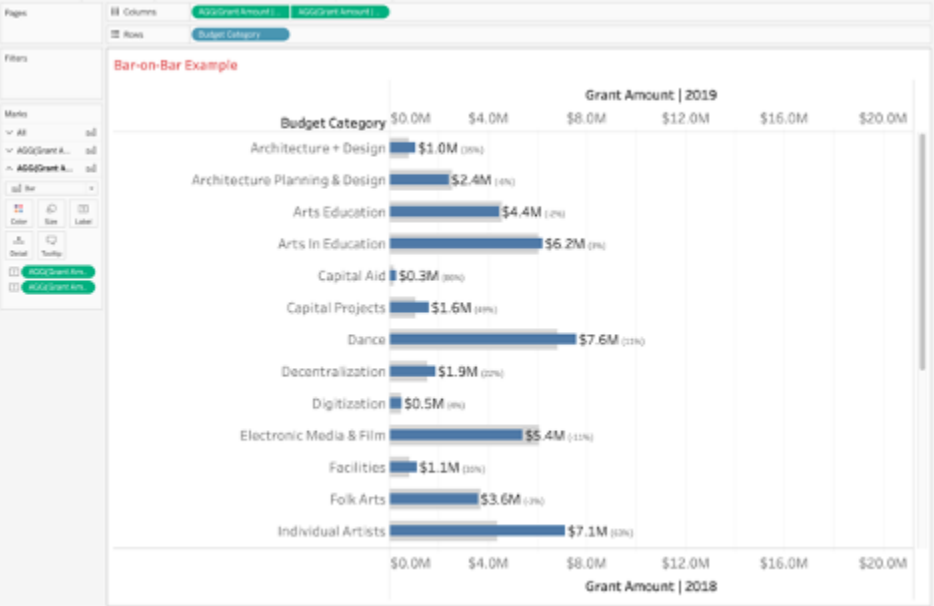


Figure 1-14. A look at the bar-on-bar chart before we finalize formatting.

Step 4. Add color.

Create a simple boolean called **[color]** that compares 2019 to 2018 and add it to the [Grant Amount | 2019] marks card:

```
// Color

[Grant Amount | 2019] > [Grant Amount | 2018]
```

You can edit the color and select two colors that start with the same hue: for instance the base hex color #19626B for values that are TRUE and second color, #84B6BC, that is brighter and less saturated.

Step 5. Add finishing touches.

Step 5a. Sort your categories by total grant amounts in 2019.

Step 5b. Hide your axes and row header labels. Remove all extra lines.

Step 5c. Remove your vertical divider. Keep your horizontal divider, but make sure it separates each member.

The final result is the following bar-on-bar chart:

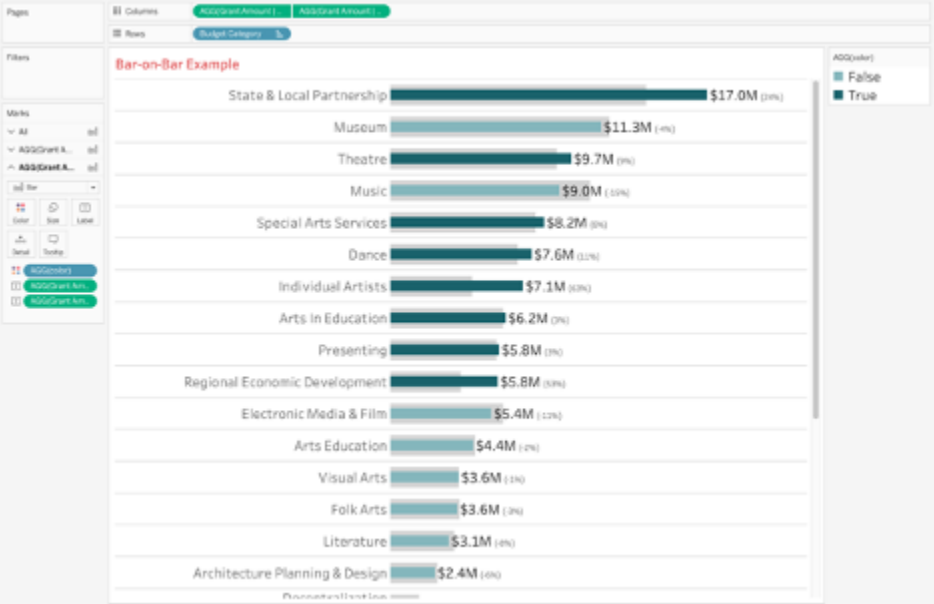


Figure 1-15. The bar-on-bar chart after adding color.

Treemaps

A treemap is similar to a bar chart, but uses the area of a rectangle relative to the height or length to encode data. If you have many members of a dimension and must show all members, a treemap is a great alternative to a Bar Chart.

Area is a less precise measure, but often when working with treemaps the goal is not to be completely precise, but rather to display all the members of a category in a single, compact visualization that is sorted from largest to smallest.

One of the main benefits we'll explore with treemaps is how to use color to represent a measure or a dimension. In the example with a drillable treemap, we'll use color to represent both Budget Categories and the Program Categories within them. We'll then show how to show additional detailed members to the audience on demand. With this feature, the audience is free to explore multiple facets without being overwhelmed. We'll also represent color as a measure, both directly and indirectly.

Nonprofit Case Study: Showing all members of a dimension in a single visualization

Next, we'll be focusing on what to do when you need to show all members of a dimension in a single visualization. You've already seen the problem of a long scroll when using a bar chart, so what chart type can you employ to get around this barrier?

Imagine you are working with a nonprofit organization that controls and awards grant money for Creative, Performing, and Cultural Arts Programs and Initiatives for the state of New York. Grant money is broken into 2 category types: one is related to the organization's budget and the other more directly categorizes the programs' initiatives.

In this scenario, you can't sacrifice small members and it's actually crucial to have visibility into some of the smaller categories, which can provide insight into where additional grant money should go.

If you're facing a similar scenario, we recommend a treemap. While you'll lose some precision on comparing your chosen measure, you will get a very well ordered and compact visualization that will show all the members of your dimension.

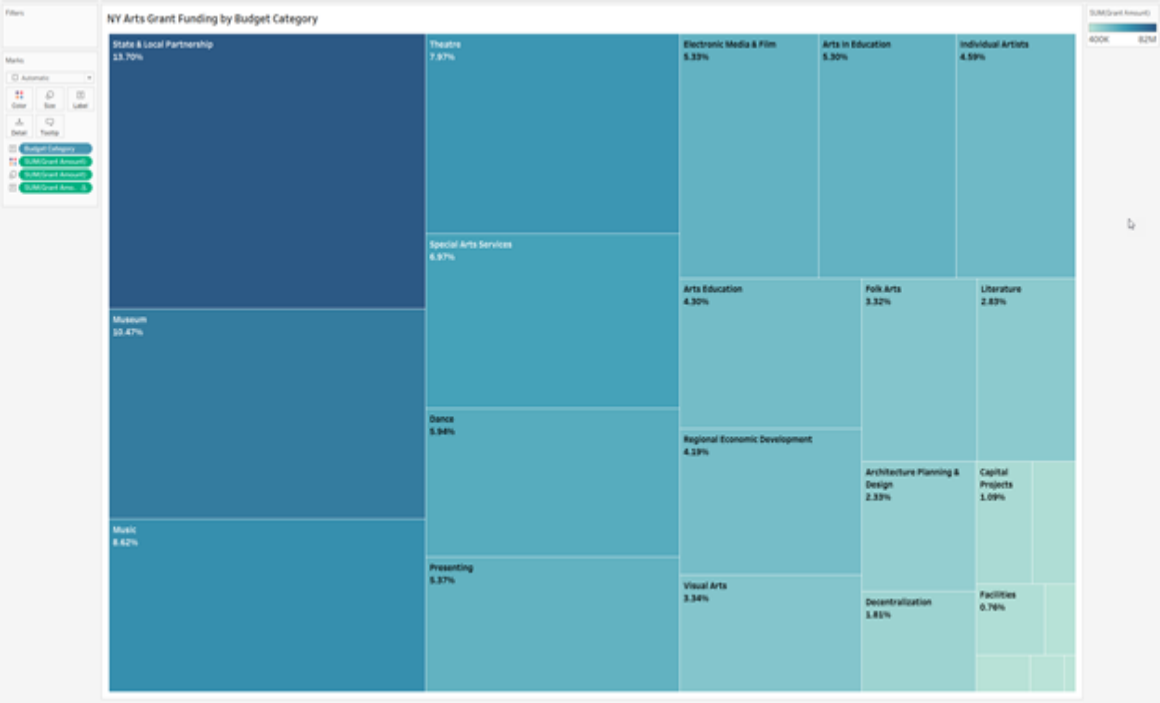


Figure 1-16. A treemap showing the Budget Categories ordered by Grant Amount.

Blueprint: Creating a basic treemap

1. Drag on **Budget Category**" to Label
2. Drag on "SUM(**Grant Amount)**" to Size**,** CTRL+Drag this to Color as well
3. Ensure mark type is set to "square"

This will give you a full picture of how Grants are distributed. There aren't any Budget categories that have a significant majority of funding, but some smaller categories take up less than 1%.

TIP:

The standard convention when working with treemaps is to double encode a measure using size and color. It helps to further distinguish the different pieces and members. This is not a requirement. As an alternative, you could consider encoding the categories on color, but we would recommend caution since there are many members and, in this case, you would be encoding redundant information.

This treemap is a great start, but only takes us halfway to getting specific with the Grant data. Not only are dollars divided among Budgets, they are also assigned to Program Categories. There are 55 unique Program Categories, a significant additional level of detail that could be overwhelming. We only really need to know which program categories that budget dollars are tied to within one given Budget Category at a time.

To solve the “next level question” of Program Categories within Budgets, we chose to create a *drillable treemap*: an interactive treemap where the audience clicks on a specific Budget Category to see further information.

Blueprint: Creating drillable treemaps

1. Create a set based on both **Budget Category** and **Program Category** by first dragging **Program Category** onto the Marks Card of the treemap view. This will allow you to click on a mark and create a set that combines both dimensions. Right-click on any mark and create a set called “Program & Budget Set”. It does not matter what values are in the set initially, only that there are 2 columns, one for each dimension.
2. Now create a calculated field called “Label Program”. This will evaluate if something is part of the set and return the program if it is.

3. IF **Program & Budget Set** then **Program Category** END
4. Drag this calculated field on top of Program Category on the Marks Card.
5. Create the Drill-down functionality. Go to **Worksheet -> Actions -> Add Action -> Change Set Values**.
6. Call it **Drill Down to Program**. It will be run on “Select”. The Target Set is **Program & Budget Set**. The action you want when clearing the Selection is “**Remove all values from set.”**
7. Now click on “**Dance**” and the treemap rectangle will drill in to show all the Programs that comprise the Dance budget.

Figure 1-17. A treemap showing both the Budget Category and Program Category; the treemap is colored by Budget Category and the size of the rectangle is Grant Amount.

Notice that we’ve changed how color is leveraged here. Instead of tying color to the repetition of the grant spend, we’ve chosen to use it to distinguish between the budget categories.

Let’s go back to our treemap example one more time, and take color encoding in one more direction. In this scenario you’re going to start with

Program Categories to create the treemap. This time, the organization is trying to ensure that not only are different program types getting sufficient distribution of funding, but also that there is diversity in the types of programs that are funded and supported.

For this final visualization, you'll use color encoding to spot opportunities for programs to be revitalized. You'll use it to highlight another continuous measure: days since the most recent grant was funded. (A continuous measure is one that spans an infinite range, typically on a number line or timeline.)

Blueprint: Encoding a continuous measure with color

1. Create a treemap of Program Category and Grant Amount.
2. Create a calculated field that evaluates how long it has been since a grant was awarded in a category.
3. **Days Since Last Grant**:
   DATEDIFF('day',MAX(**Date**),TODAY(),) - When you use this calculation in the view, it will evaluate the maximum or most recent date per Program Category and then calculate the number of days since Today.
4. Put this measure on Color.

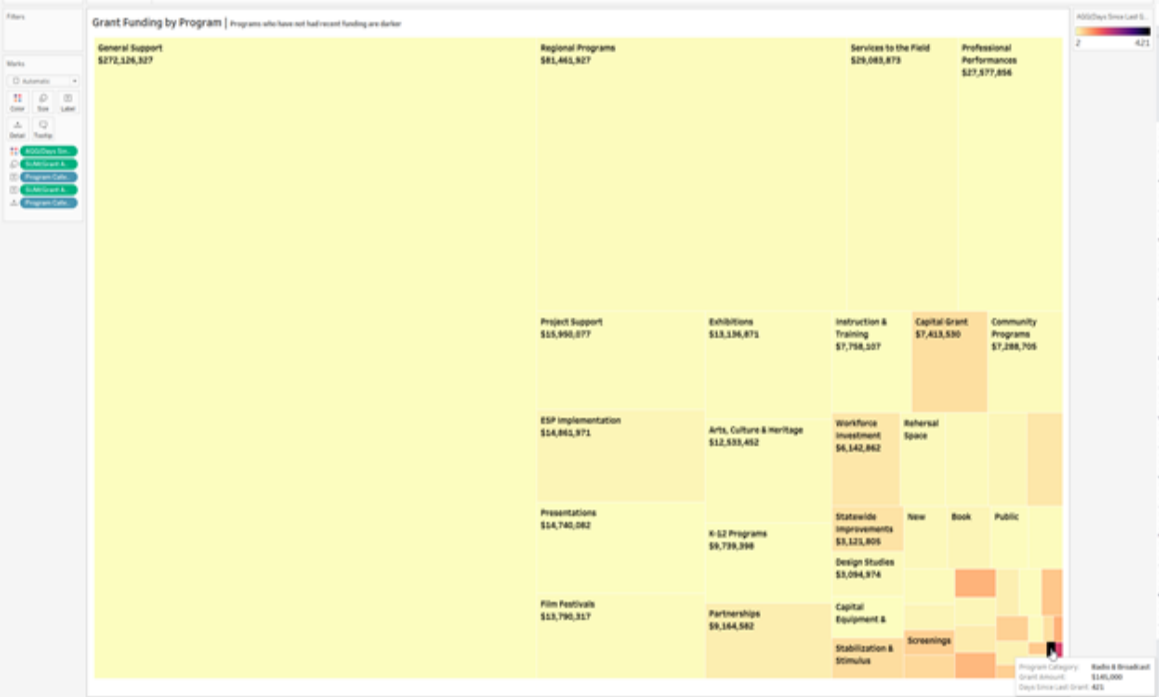


Figure 1-18. A treemap showing Grant Amount by Program Category; color for the treemap has been encoded to show how many days since the last program has been funded.

You've now seen three different approaches to treemaps, utilizing different color encoding techniques and dynamic elements that allow the audience to dig deep into a category and explore it in even more depth.

Pie and Donut Charts

In the last case study, you'll use a pie chart to represent the distribution of respondents to a survey. While it is important to exercise caution with pie charts, we think they are extremely appropriate as compact color legends or interactive filters when used correctly.

You'll also turn a pie chart into a donut chart by adding a hole to the middle. This allows you to represent two concepts in a single chart: the number of respondents (sample size) and the distribution of gender. Adding sample size tells your audience how much they can trust the survey responses.

Finally, you'll take your donut charts one step further by making small multiples, or repeated versions of the same chart, separated out by profession. This will let you compare gender distribution among several professions at the same time.

IT Survey Case Study: Using a pie chart to represent survey results

Imagine you're working with results from a survey of IT professionals focused on their employment.

As with most surveys, there are many questions to analyze to determine attitudes, but a core task is to provide a demographic overview of respondents. This example will show the gender distribution among the survey respondents.

To tackle this problem, you decide to utilize a pie chart. If you consider yourself a visualization purist, you may be cringing at the thought of a pie chart, but pie charts do have their place. For this request, it's important to focus on the positive components of a pie chart. One major positive: it is the first data visualization (besides a map) most people are exposed to. Pie charts represent a natural part-to-whole relationship. They also use space efficiently and can serve as color legends or interactive filters.

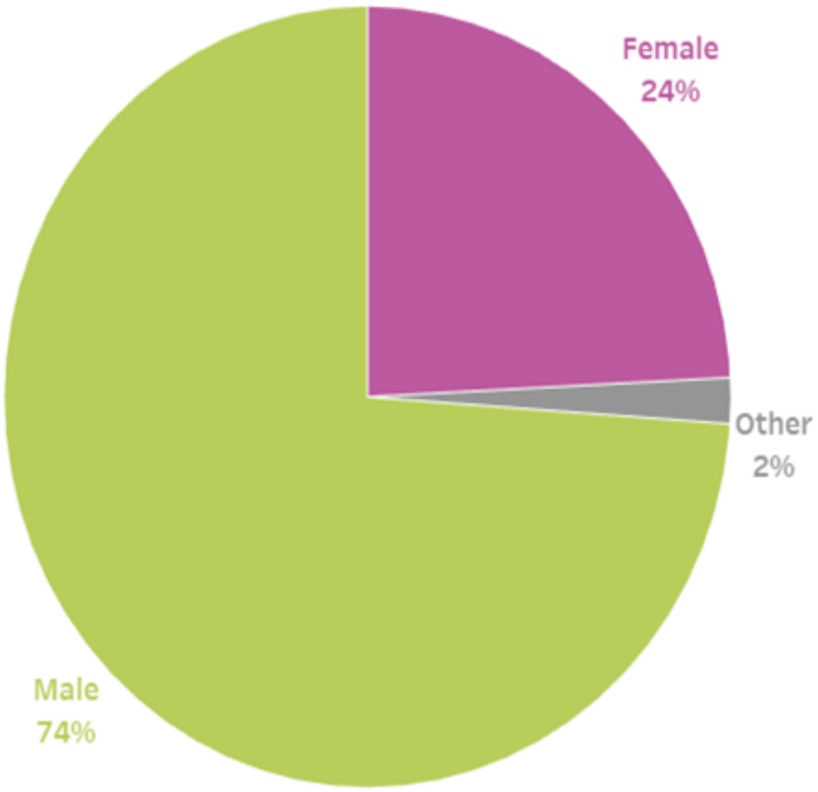


Figure 1-19. A pie chart showing the distribution of gender for survey respondents.

Blueprint: Building a pie chart:

1. Drag **Gender** onto Color
2. Change the mark type to Pie
3. Drag # Respondents onto Angle

Why does this pie chart work? First, there are only three slices – this pie doesn't have many pieces, so it's pretty easy to compare the difference. Next, we've gone the extra mile to directly label and put the percentage next to each slice. This makes it very easy for the audience to comprehend. If we had many more values, or had several slices of relatively the same size, our recommendation would be a bar chart, but here that's clearly not the case.

What if you want to take further advantage and display two pieces of data at once? This is where a donut chart comes in handy. A *donut chart* is a fancy pie chart with a hole in the middle. The hole in the middle allows you to communicate an additional piece of information: in this case, along with the gender distribution, you can include the number of respondents. This compact format adds an immediate contextual element to the gender distribution.

Blueprint: Building a donut chart:

1. Create a dummy measure MIN(1) - this will be used as multiple measures for a dual axis chart
2. Drag it on to Rows twice, right click and make dual axis
3. Click on the first measure and add "Gender" to color, add "# respondents" to angle
4. Make the size of the first measure the right tick on recommended size
5. Click on the second measure and drag "# respondents" to label
6. Make the size of the second measure the left tick on recommended size
7. Align the label middle and center

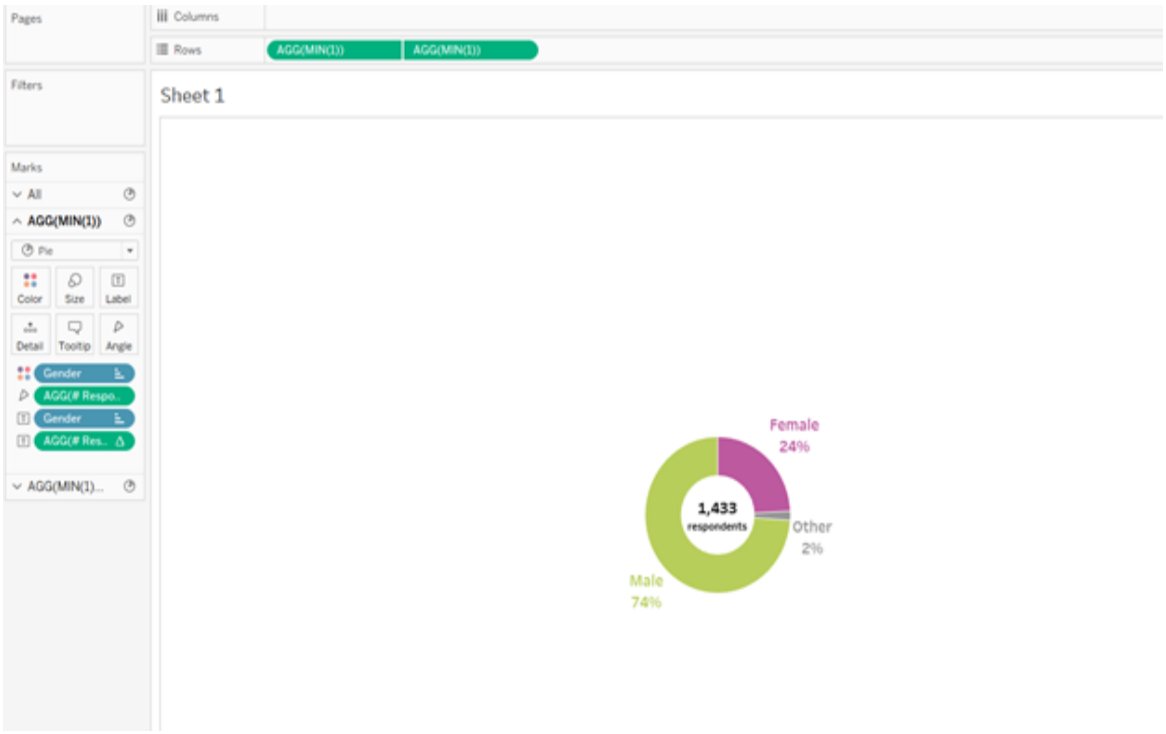


Figure 1-20. A donut chart with number of respondents has been added to the center; the slices represent the distribution of survey respondents by gender.

Now you have a donut chart that serves two purposes and is fantastic at generating insightful sentences. We take the statement of “24% of respondents were female” to immediately know the survey’s sample size, which will be necessary for the audience to make decisions from the results.

To complete the donut chart, you could utilize it as an interactive filter: a user clicks on a slice and subsequent visualizations filter. It most likely can also serve the purpose of a color legend - one that does more than just say that green means male.

Since we’re working with dessert charts, there’s one last visualization we’d like to introduce: th*e small multiple,* any chart repeated multiple times in a smaller format. With pie and donut charts, small multiples become pretty powerful. You can take a dimension with more members and use that to create repetitive charts for comparisons.

Let’s say you want to see how gender distribution changes among different professions. Small multiples can help you separate out the answer quickly. Yes, you could create a filter to select the role, but you’ll get more insight at

one glance when you create small multiples that can show the distribution and still provide the sample size in context. Starting with the donut chart, you only need to make a few tweaks

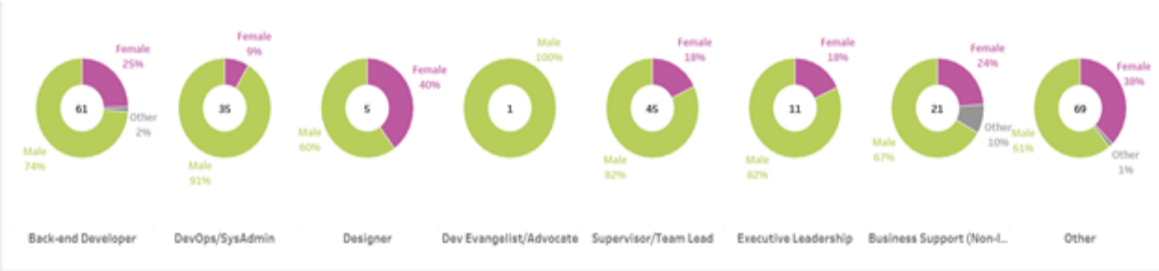


Figure 1-21. A small-multiples donut chart, which includes separating out the distribution by gender among different professions.

Blueprint: Using a donut chart as an interactive filter

1. Remove the word *respondents* from the label to save space.
2. Drag Professional Role onto columns.

Now that you've seen some examples of pie charts, we hope you'll recognize when it is appropriate to utilize them and more importantly, when utilizing them can enhance the data presentation for your audience.

Conclusion

You've now had a chance to see different chart types used for categorical analysis.

With these techniques, you will be able to create flexible, compelling, and insightful visualizations that let your audience explore questions and analyses dynamically.

Chapter 2. Quantitative Analysis

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 2nd chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the authors at ann@jacksontwo.com and luke.stanke@tessellationconsulting.com.

This chapter is about crafting visualizations that show distribution, spread, and plenty of nuance among the numerical representations of data values. If you've ever worked with data, you know it can be a challenge to communicate detailed information. This chapter offers some techniques that will help you overcome this challenge.

Visualizations in this chapter will focus on measures and statistics. Using numeric fields, we'll show you how different charts can provide different insights. *Measures* are numeric fields to which we can apply functions, like SUM. *Statistics* would be considered a more specialized use of functions, specifically AVG, MEDIAN, MIN, MAX, and some derivative functions like STDEV. These functions don't necessarily produce a summative value, like total sales, but instead usually describe the components of a dataset, like the distribution, commonalities, and extremes.

What You'll Learn in This Chapter

1. How to create a histogram
2. How to create a dot plot and a jitterplot
3. How to create line charts with statistical control lines
4. How to create a pareto chart

What You'll Need

1. Superstore data set
2. Logistics data set

Histograms

A histogram is a specialized bar chart that is used to visualize distributions of data. Unlike a normal bar chart, the x-axis has a numerical value, range, or interval (often referred to as a bin) represented as a header and a y-axis that represents the frequency. Histograms are very useful if you are trying to understand the spread of data points that have a numerical value – think about the number of customers who spent between $100 and $200 last year at a retail store or the number of times someone purchases a specific product.

Office-Supply Store Case Study: Analyzing Consumer Behavior with Histograms

Imagine for our first use case that you work for an office supplies store. You have been asked to analyze and share purchasing behavior of your customers. So where do you start? What metrics would you want to tease out of the data?

A natural place to begin would be to look at order frequency. How many orders do your customers place? Are most of them frequent return shoppers, or do they only purchase once?

To answer this question, the best chart to turn to is the histogram. A *histogram* takes a frequency, like the number of orders per customer, and creates a bar chart with the frequency (for example, 1, 2, 3, or 4 orders per year) on the X-axis and the number of customers on the Y-axis.

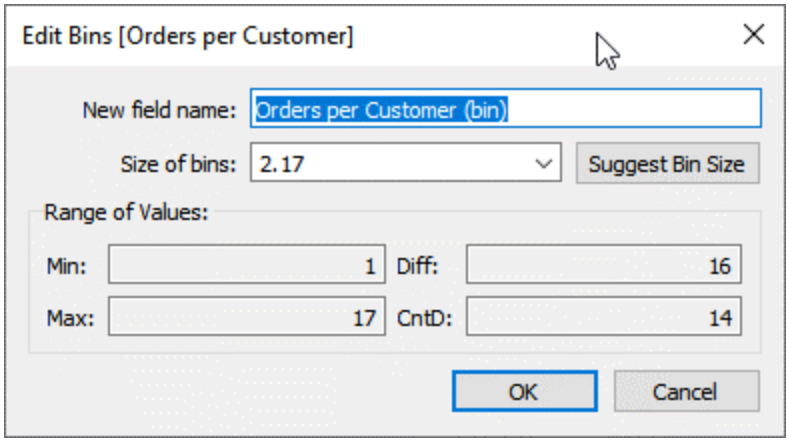


Figure 2-1. Built-in functionality to create numeric bins.

Blueprint: Making a simple histogram of purchasing behavior

1. Start with a measure: in this case, Orders per Customer
2. Right click on the measure, go to Create, and then select Bins…
3. When you do this, a dialog box will open. Tableau will automatically suggest a bin size after reviewing the various values for your measure. You are free to adjust these to suit your needs.
4. For our analysis, it doesn't make sense for the number of orders per customer to be a decimal, so we will change the bin size to 1.
5. Once we create the bin, it shows up as a discrete dimension in our data pane. This is an important point, because it will give us insight into how the field will behave when we use it in a chart.
6. Now, drag on the Orders per Customer (bin) onto Columns. As expected, Tableau has created discrete headers.

7. Now, drag on the # of Customers, which is the COUNTD(Customer Name) measure, onto Rows.

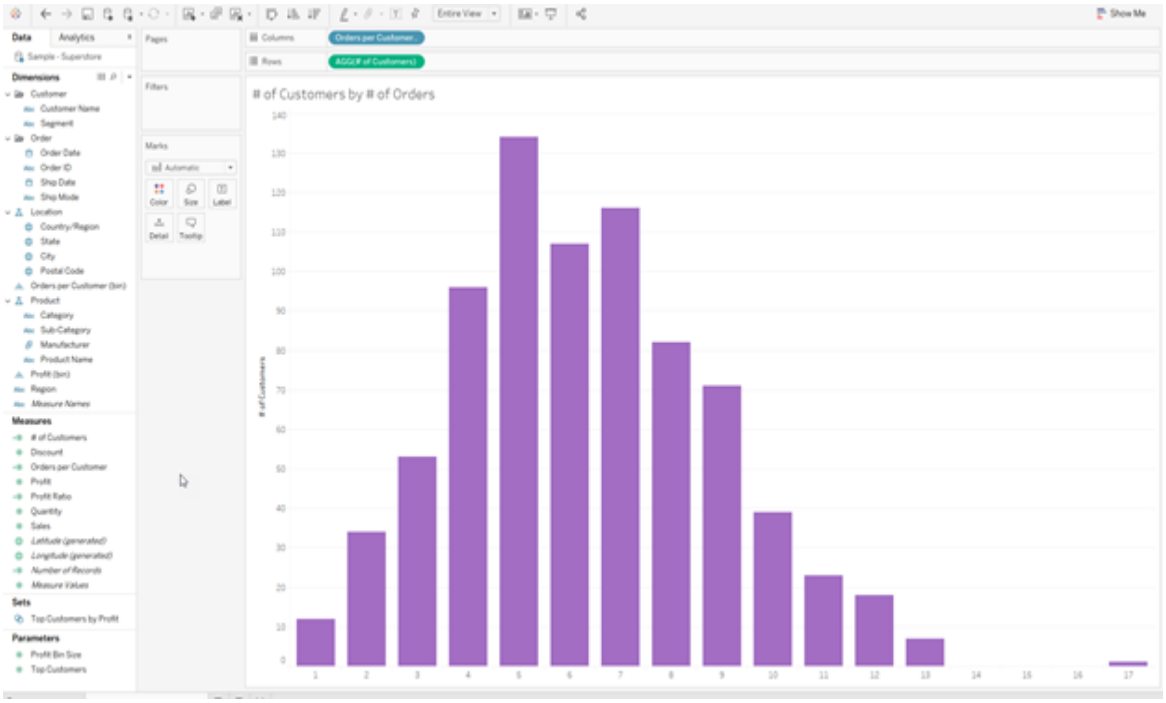


Figure 2-2. A histogram of order frequency by number of customers.

Let's stop and digest this visualization. Whenever you're looking at a histogram, a chart that is common in statistics should come to mind - the bell curve, or normal distribution. The *normal distribution* is when data is represented such that the top of the curve represents the mean or average. It is symmetrical: exactly half of the data is to the left and half is to the right of the mean. Finally, a majority of the data is within one standard deviation (SD)of the mean.

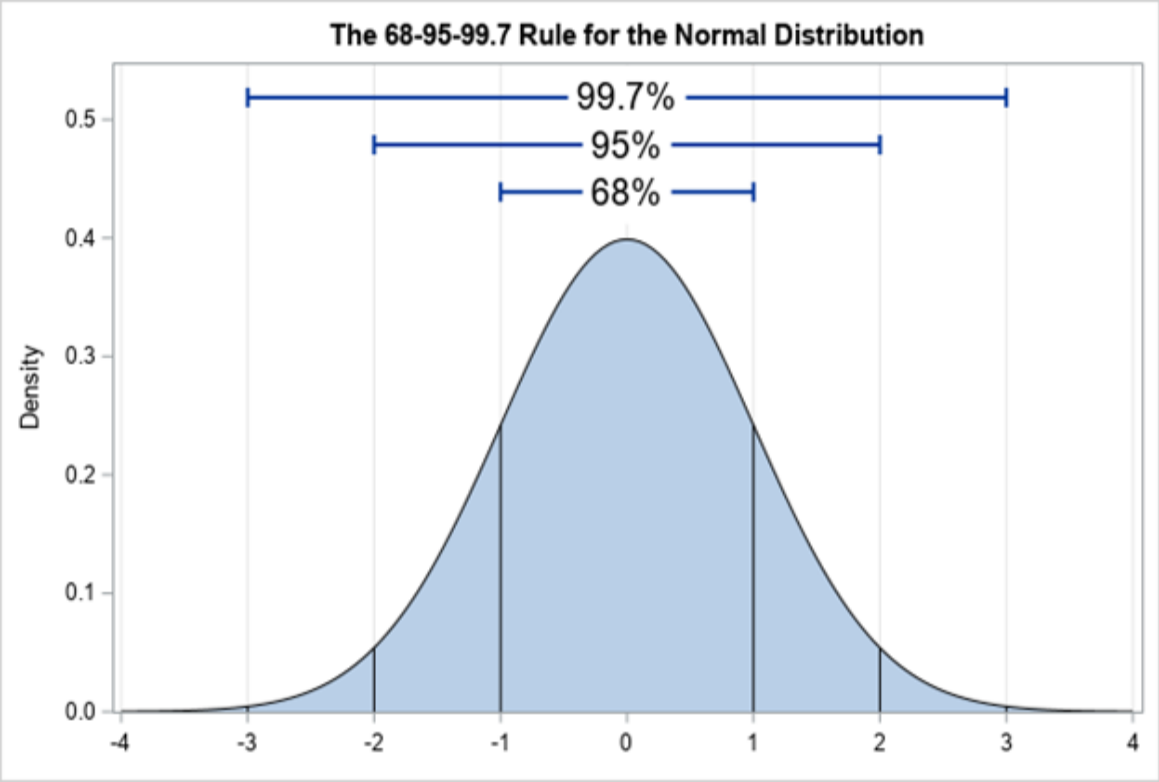


Figure 2-3. Figure 2-3. A normal distribution with percentages for each standard deviation.

While the histogram you created is *not* a normal distribution, comparing it to the normal distribution will prove useful in trying to explain the shape of the data.

You can see from the chart in figure 2-2 that the order frequency is definitely centered between 4 and 8 orders, and that there are a few customers (l *outliers*) who have a much higher ordering frequency.

We can create even more immediate value out of the histogram by cutting it into small multiples and comparing the distribution, or spread, of the data among a dimension such as segment, as shown in Figure 2-4.

Drag Segment onto Rows and also onto Color.

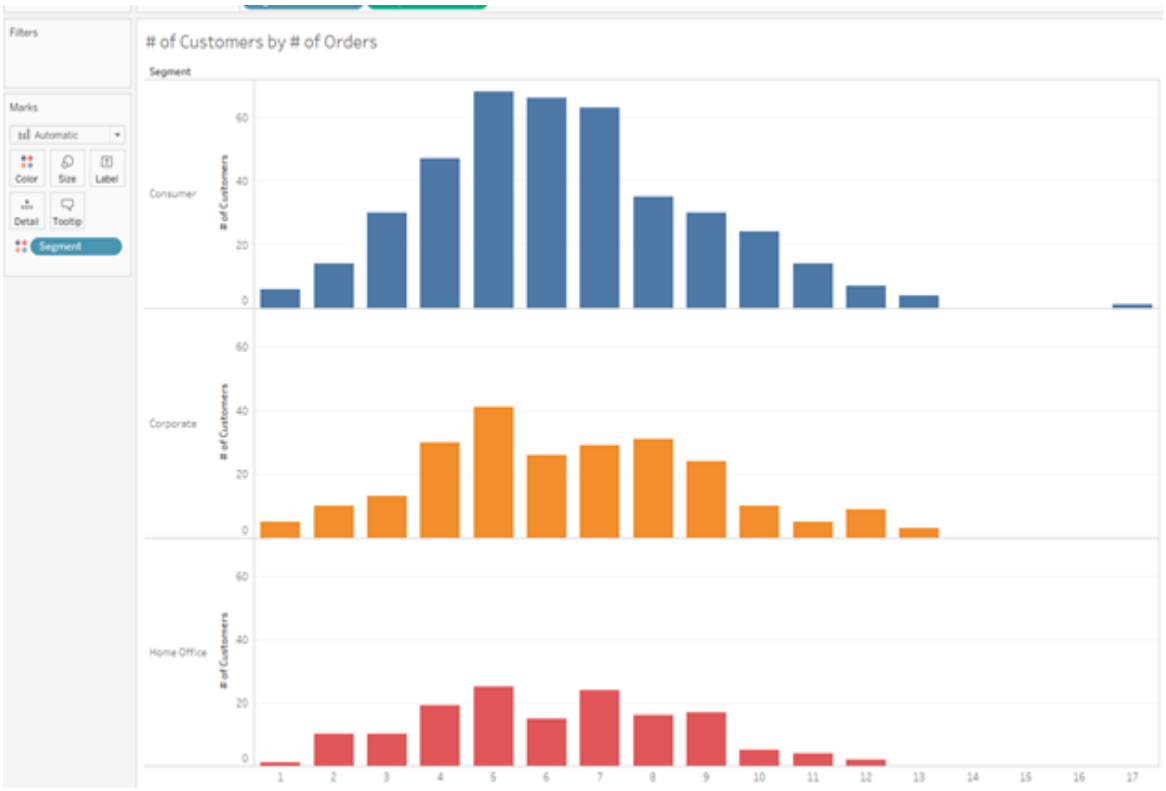


Figure 2-4. Figure 2-4. Histograms of order frequency by customer separated by segment.

Now here is where things get interesting. Previously we saw the overall distribution of order frequency among customers, but now we've separated it out by business unit/segment.

While the data shapes of each of the histograms is **similar** to the original, there are definitely some discernible differences among them. First, we can see that the overall number of customers purchasing from Consumer dominates the other two Segments. We can also see that those outlier customers only exist within Consumer. Additionally, it looks like the order frequency for Home Office is relatively stable between 4 and 9 - the amounts are much closer together.

For more precision, we might suggest changing the Y-axis to be independent for each row, as shown in Figure 2-5.

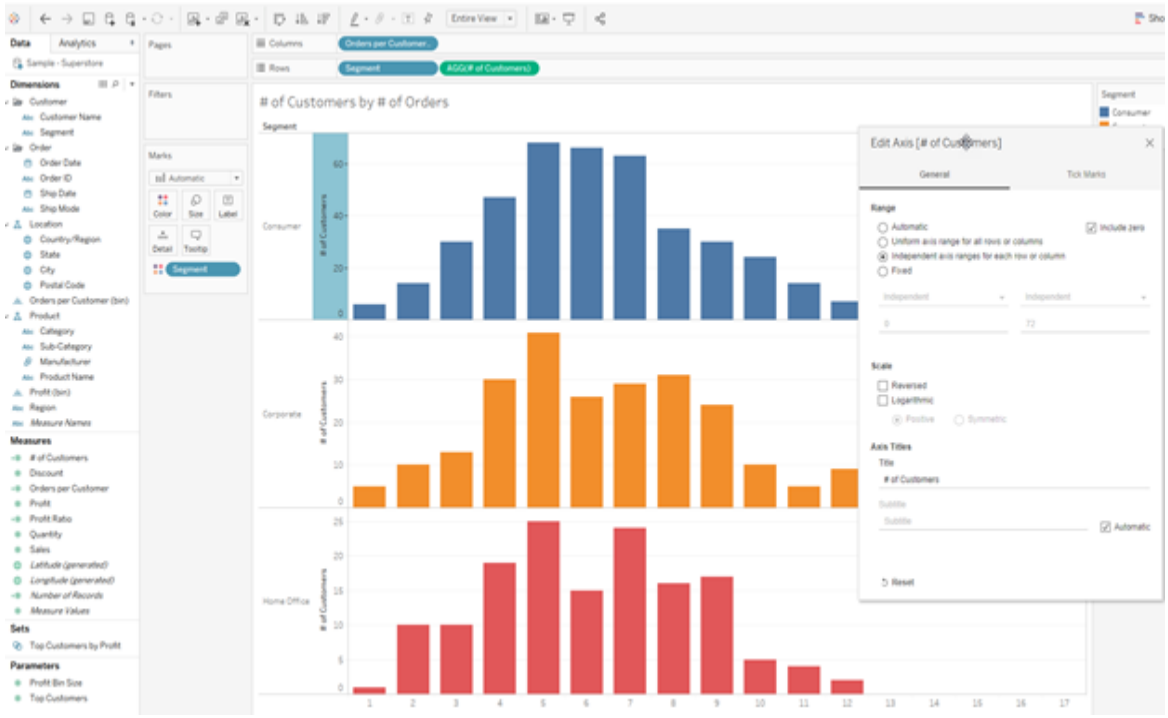


Figure 2-5. Histograms of order frequency by # of customers with independent Y-axes.

When you make this change, do so with caution. It should be very clear what the bar lengths represent and that they are unique between each of the Segments.

Let's look at a few other techniques you can utilize to make even more effective histograms.

Blueprint: Creating a histogram with a continuous bin

1. Change your bin from discrete to continuous. Recall, from Chapter 1, that continuous fields will draw axes. This means that instead of having a header for each order bin, we will have an unbroken number line. The width of the bars will automatically change to the size of the bins, in our case, 1.

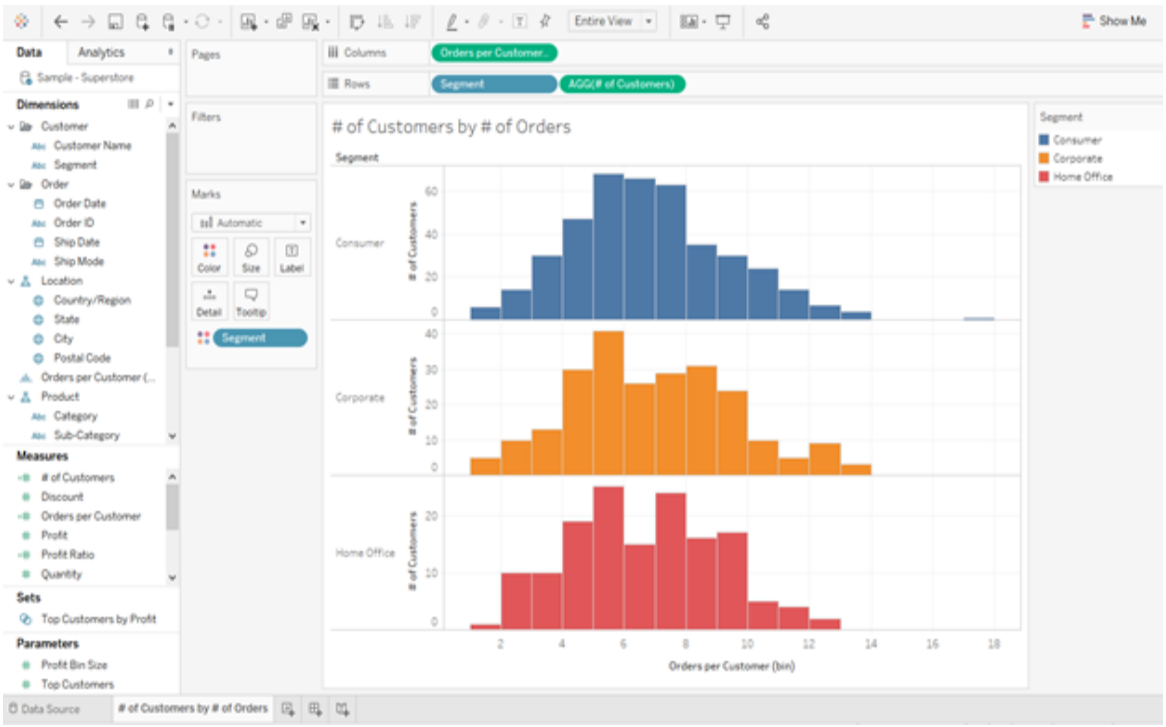


Figure 2-6. Continuous histograms of order frequency by number of customers. Note the lack of space between bars.

This is how n histograms are usually expressed in mathematics and statistics books. Showing the bars without space in between is probably more appropriate for a histogram.

Blueprint: Using level of detail expressions in histograms

The last thing we can do is take the existing bins and create a bin for anything of a certain number or greater. For this type of chart, we will utilize a level of detail expression instead of bins to construct histograms. A *level of detail expression* (LOD) in Tableau allows you to define the aggregation of a measure. By default, the aggregation of any field is based on the detail or fields in the view, but by using LODs, you can create customized aggregated fields.

Step 1: Create a field

Create a field called # Orders per Customer that is defined as {FIXED CUSTOMER NAME: COUNTD(ORDER ID)}. This tells Tableau to count the distinct number of orders by each customer name.

Step 2: Reaggregate

Once you make a level of detail expression, you can reaggregate it in your view. If you are creating a fixed LOD, you can even choose to bring it on without an aggregation, as either a continuous or discrete dimension.

Figure 2-7 shows the resulting visualization.

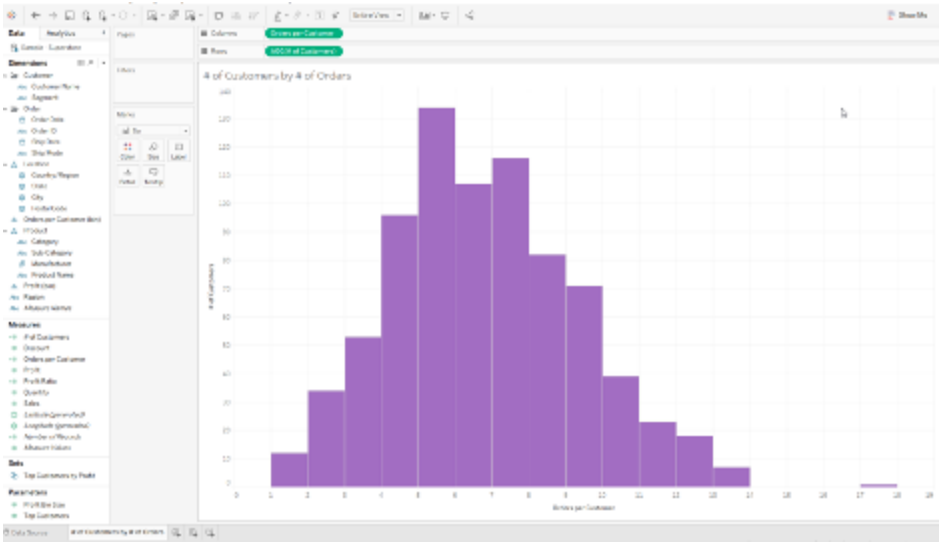


Figure 2-7. Histogram with level of detail expression.

Step 3: Combine orders

Combine orders per customer that are greater than 11 into a single section. This will get rid of the empty bins and group together everything into one final bar (see Figure 2-8).

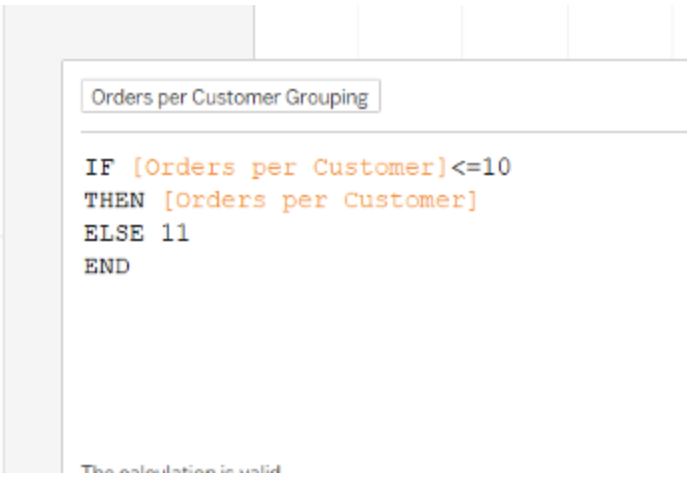


Figure 2-8. Orders per Customer grouping.

Drag and drop to replace this with the measure on Columns. You should right-click drag this into the visualization so that you can quickly drop it as a dimension as opposed to an aggregated measure (see Figure 2-9).

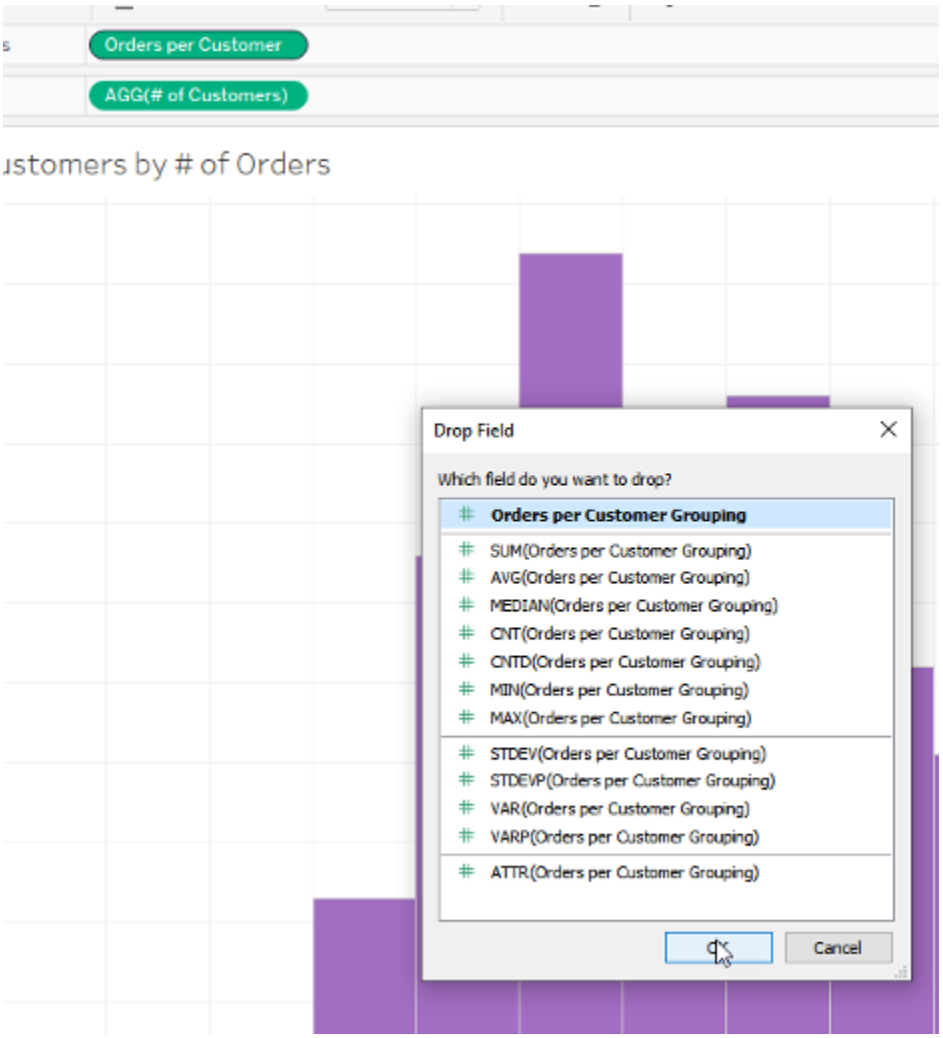


Figure 2-9. Dropping a field.

Now we have a chart that has everything over 10 in a single bar, as shown in Figure 2-10.

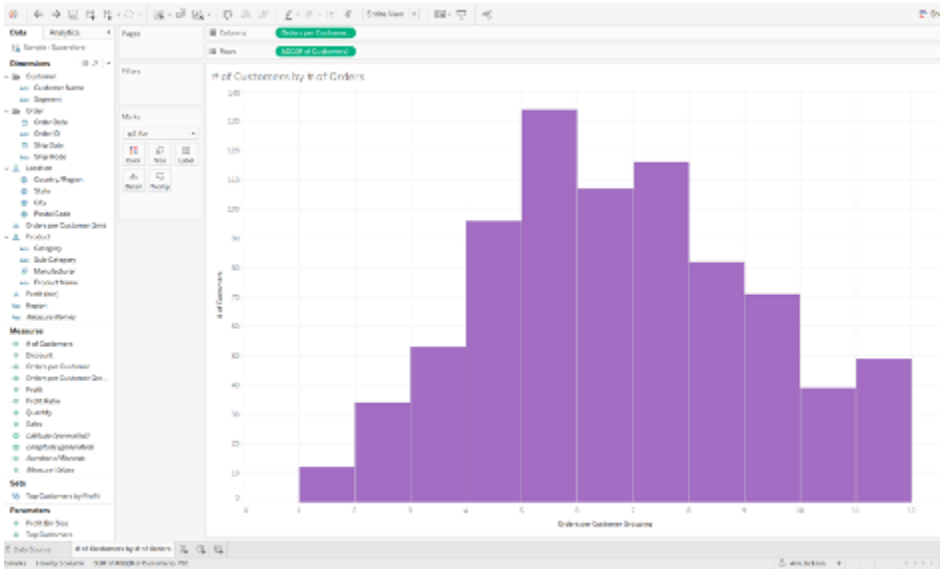


Figure 2-10. Histogram with everything over 10 in a single bar.

Consider this an intermediary step to our solution. We would never suggest that you should share this histogram with your audience: the final bar would be confusing and potentially misleading.

Step 5: Rename labels for values of 10+ for labeling accuracy

Instead, we will make one last change. First we'll change the field to discrete. Next we will *alias* (that is, assign an alias to) the value of 11 to 11+.

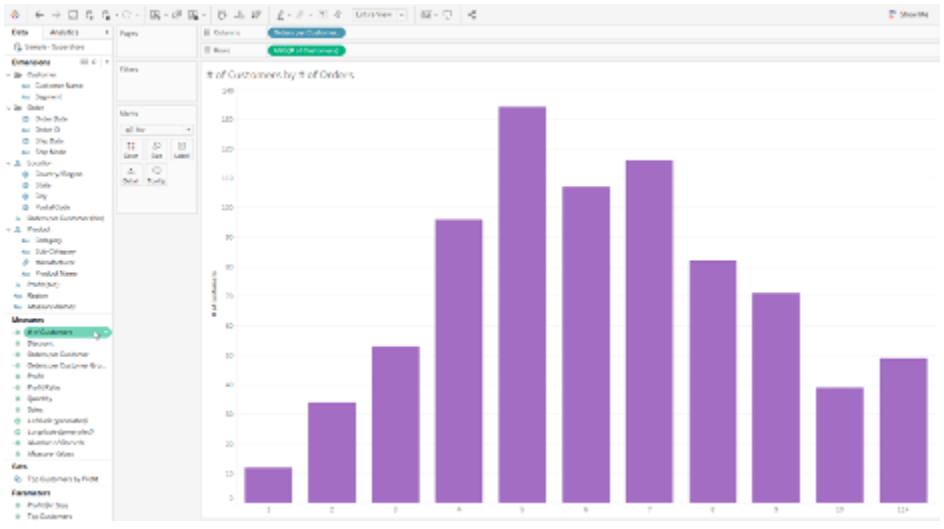


Figure 2-11. Completed histogram.

Now you have neatly dealt with the right “tail” of the data. And you’ve clearly communicated to the audience that the final bar is everything 11+.

Why go through this extra effort? First, we reduce the overall noise in the chart. Odds are, when you looked at the initial histogram, your eyes were drawn over to the right. This may have even prevented you from accurately remembering the overall shape of the histogram. Next, it makes the chart easier to read aloud. Since there are so few members in 11 through 17, bucketing them together allows us to assign a single number to a single grouping.

Dot Plots

Another type of chart you can use to display quantitative information is a dot plot. A dot plot is a one dimensional chart where data is represented by circles, or dots. This type of chart is effective for situations where you have lots of data points, members within a dimension, or the distribution of data points is an important insight. We also like working with dot plots because they take up very little visual screen space.

Office-Supply Store Case Study: Analyzing Consumer Purchasing Behavior with Dot Plots

For our next scenario, let’s stick with our office supplies store. Now that we’ve gotten a handle on the frequency with which our customers order, it’s time to find out even more about them. Next you may want to explore is the *average order value*. And yes, we *do* mean average. Your company likely wants to know: “On average, how much do our customers spend on each order?” So let’s dig into various ways to find that answer.

If anyone ever asks you how to provide the average of a metric, proceed with caution. An average is a very aggregated value; whenever you only present that level of information, you run the risk of oversimplifying the

underlying dataset. To avoid that, we suggest starting with the detailed data and then presenting the aggregated average.

For this use case, we always start with the dot plot. A *dot plot* is a one-dimensional chart where each dot represents a single data point. Let's go ahead and build out one for our scenario.

Blueprint: Creating a basic dot plot

1. Drag Order ID onto Detail on the Marks Card - this will define the level at which you will display your data.
2. Drag on SUM(Sales) to Columns.
3. Change your mark type to Circle.
4. Since we have such dense data toward smaller values, add Transparency by changing the Opacity of Color to 60%.
5. To ensure that you answer the original question, drag on an Average Line from the Analytics Pane across the entire Table.

Now that you have the underlying picture, what conclusions would you draw? We would conclude that certainly there is a lot of data-density around smaller values, and most likely around the average, but we can't neglect the obvious outliers.

To clean up this view, you need a way to expand all the data points that are collecting on top of each other. The technical term is to *jitter* the data-- that is, turn the dot plot into a *jitterplot.*

Jitterplots

A *jitterplot* is almost identical to a dot plot, but includes a numeric value on the Y axis (rows). The value on the Y-axis has no analytical value and

should not be shown; its only purpose is to spread out the data and de-densify it.

If you are using a Tableau Extract or Published Data Source, you will have access to a built-in function called RANDOM(). This function returns a random decimal number between 0 and 1. We highly recommend that you take advantage of this method if you have that type of data connection - it will ensure that your jitter is natural and not influenced by any other metrics (other methods involve indexing your data, requiring some type of ordering to be placed on the data).

Office-Supply Store Case Study: Analyzing Consumer Purchasing Behavior with Jitterplots

When you create a jitterplot, you get the advantage of a dot plot, but with the members spread out randomly among the y-axis. The result is that data poitns no longer overlap as much, so you can more clearly see the data points that are clumped together at a numerical value.

Blueprint: Creating a jitterplot

1. Double-click into rows and type RANDOM(), then press enter. You will notice that the aggregation will be the SUM(RANDOM()). Change this, so it doesn’t over-aggregate and return values greater than 1.
2. Change the aggregation to MIN(RANDOM()) by right clicking on SUM(RANDOM()) and changing the Measure to Minimum (see Figure 2-12).

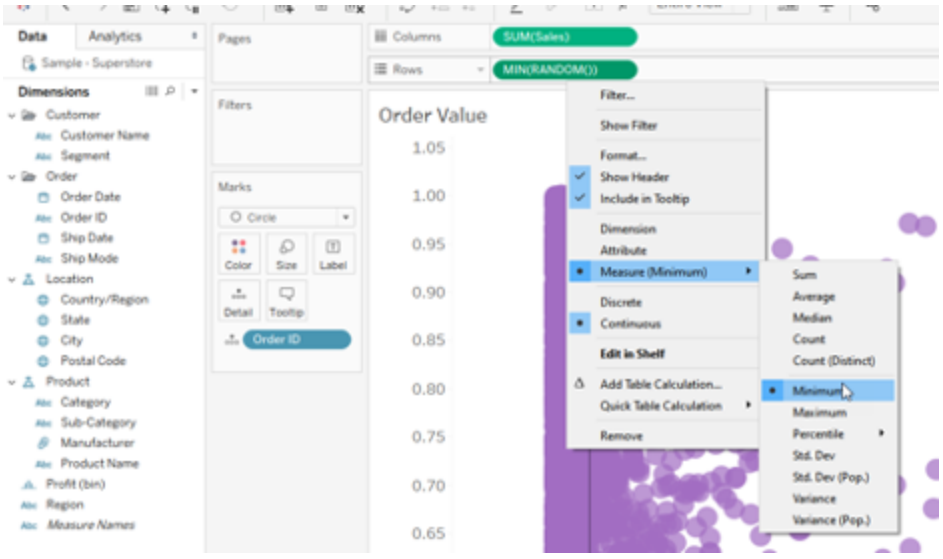


Figure 2-12. Changing the measure to minimum.

3. Right-click on the Y axis (rows) and un-check “Show Header”.

Let’s look at the resulting chart:

Figure 2-13. Completed jitterplot.

Now we have a much clearer picture of the data. When we looked at the previous chart, it was one-dimensional, so we really had no idea how much density there was toward the left side of the chart.

What might happen if you brought this chart to your office-supply executives? They’d be likely to tell you, “That’s great, but too much detail.” When your authors encounter this situation, we respond with a happy

medium: we will show the aggregate but include the detail in a secondary place, like the tooltip.

You might also find those executives asking what the average is; when you reply, they might answer, “But what if there are outliers that skew the average?” If you only had the aggregate, it would be difficult to respond - so including this additional detail immediately provides a response and allows the audience to be more analytical in their thought process.

We think that dot plots and jitterplots work particularly well when trying to show spread among one measure and identify outliers. But, if your audience still thinks that’s too much data, we have another solution.

Ranged Dot Plots

There are some situations where showing all the data points to your audience may be overwhelming. When we find ourselves in that situation, instead we choose to use Ranged Dot Plots. Ranged Dot Plots plot values like average, minimum, maximum, or median instead of all the individual data points. They can be great visual representations of the overall scope of a data set.

Office-Supply Store Case Study: Distilling Your Data with Ranged Dot Plots

So you can’t convince your executives to embrace the jitterplot - it’s just too much data for them to consume. You might need to try a distilled version that still demonstrates some key features of the data shape. The key features you’ll want to focus on are the average, of course, but also the minimum, the maximum, and the range between the two.

Figure 2-14. Ranged dot plot.

Figure 2-14: Ranged dot plot.

Blueprint: Creating a ranged dot plot

1. Make sure there is a measure that represents the total sales per Order ID. In this dataset, that means you will create an LOD expression { FIXED [Order ID] : SUM(Sales)} named Sales by Order ID. This calculation is telling Tableau to find the SUM of Sales for each Order ID, because your final visualization will not have Order ID on detail.
2. Right-click and drag Sales by Order ID onto the Column Shelf to select the aggregation, and choose AVG.
3. Make a calculated field called Range of Order Sales to find the difference between the maximum and the minimum (shown in Figure 2-15). This will be MAX(Sales by Order ID) - MIN(Sales by Order ID).

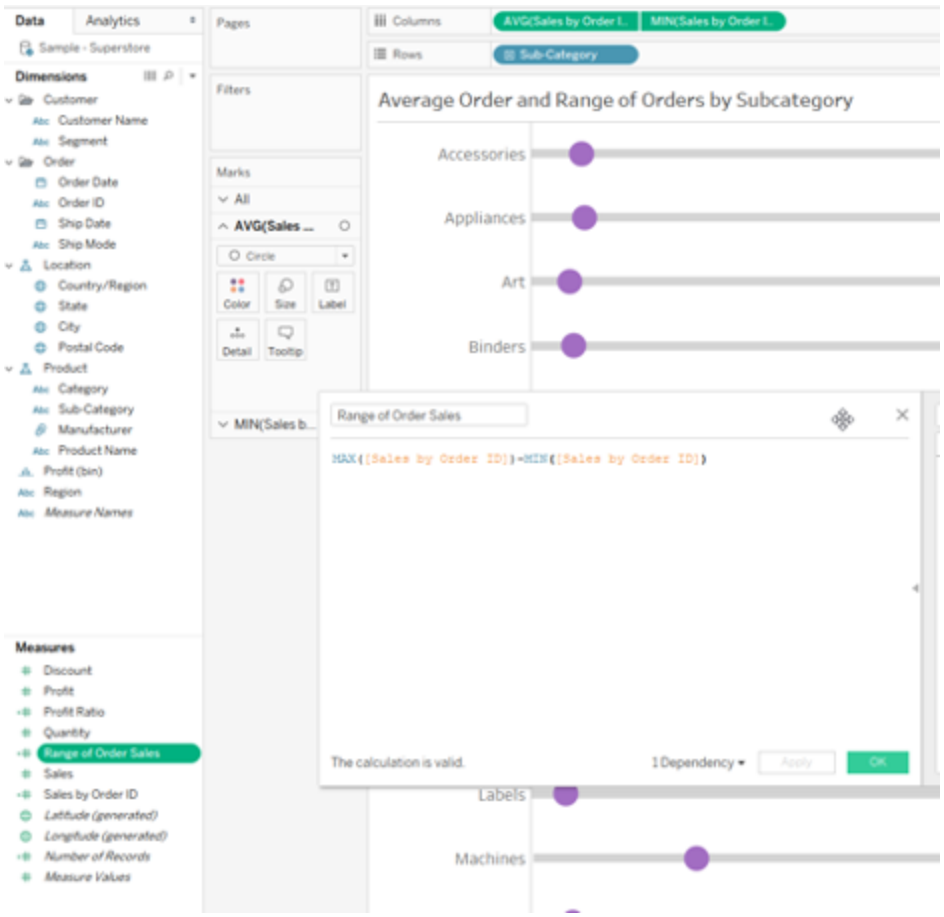


Figure 2-15. Making a field called Range of Order Sales.

4. Right-click drag on the Sales by Order ID to the Columns shelf again, but this time select Minimum. (You need a starting point for your range, and what better starting point than the minimum?)
5. Change the mark type to Gantt Bar. This will allow you to put a measure, namely, our Range of Order Sales, onto Size.
6. Drag Range of Order Sales onto Size.
7. Drag Sub-category onto Rows so you can compare the range and averages among the product categories.
8. Adjust formatting and sizing until you like the final visual.

This visualization tells a much more compelling story. At a glance your audience can see many elements, like the average per sub-category and the range, and do some natural visual comparisons. They can see that

Bookcases orders average $905, and that the range isn't too extreme compared to peers.

If you like this visualization, there are a few ways to take it further. One way to enhance this view, and one that may be more appropriate to your data, would be using the median instead of the average. If you go through this process and find that you have heavy skew or outliers, try changing the average to the median. If your audience can't decide, build in a parameter that allows them to toggle between the two.

You can also leverage percentiles instead of the minimum and maximum range. If you imagine your data spread out, percentiles allow you to cut up your data points into chunks, ranging from 0% to 100%. The most common percentiles are quantiles and deciles. *Quantiles* is when data is broken up into four equal portions, each 25% of the dataset. *Deciles* would be breaking up the data into 10 equal pieces, each 10%. Percentiles are also used in statistical scenarios that relate back to the normal distribution. Remember that in a normal distribution, 95% of the data is 2 SD from the average. So it may be appropriate to limit the range to between 2.5% and 97.5%, or to say "95% of the data" is within a specific range. Figure 2-16 shows the updated view.

Figure 2-16. Ranged dot plot with leveraged percentiles.

Blueprint: Creating a ranged dot plot using the median and leveraged percentiles

1. Change the aggregation of the average to the median. That is a simple-right click.
2. Construct the Percentile calculations (Figure 2-17).

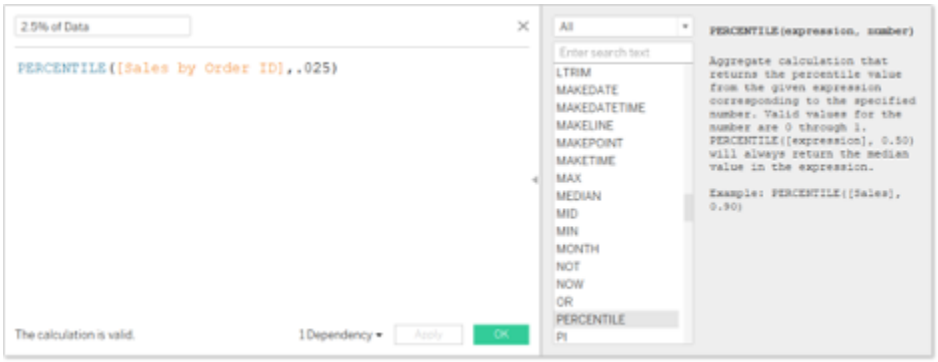


Figure 2-17. Constructing the percentile calculations.

3. You can repeat this calculation for 97.5% or .975.
4. Next we can create a new Range that is nearly identical to our original.
5. You will complete the chart in the exact same fashion

TIP BOX:

Percentile is a function in Tableau that takes in two arguments: the measure for which you want to find the percentile, and the actual percentile.

So what makes this view even more special? Sure, we've reduced the range, but we've also reduced the amount of noise in the view.

Any time we make a chart, we always need to ask ourselves what insights the audience needs to take away. If the goal is to present the average or median order value and to provide a soundbite of how 95% of orders range, we have accomplished this. It is also more likely that the insights they will

be sharing will be more relevant to their needs. If you see that there are one or two extremes that significantly pull the data shape, make it known, but provide a solution.

Box Plots

Whenever you have a highly controlled process, we find it best to lean on statistics to communicate performance. We have touched on a few different statistics, so for this use case we will combine a few concepts into one visualization, the box plot (Figure 2-18). Box plots take their origin from two different people: Mary Eleanor Spear, who created the range-bar, and John Tukey, who developed the box and whiskers plot (https://en.m.wikipedia.org/wiki/Box_plot).

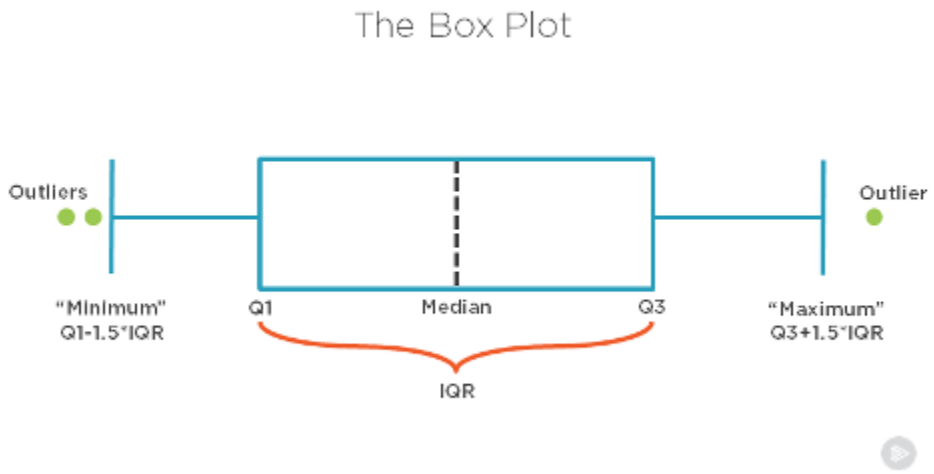


Figure 2-18. Anatomy of a box plot.

A box plot is comprised of the following elements:

- The *median*, or the midpoint the of data, is represented as a line inside of a box
- Q1 and Q3 are quartiles, Q1 being at 25% of data and Q3 being at 75% of data

- The range between Q1 and Q3 is considered the Inner Quartile Range, or IQR
- The “whiskers” of a box plot can be constructed two ways
- As shown in the figure above, the whiskers can extend to Q1-1.5 times the IQR and Q3 + 1.5 times the IQR respectively; here any data points that extend beyond the whiskers are shown as dots and called outliers
- Alternatively, the whiskers can extend to the minimum and maximum of the dataset

Shipping Case Study: Mapping Logistics with Box Plots

Now let’s imagine that you’re an analyst at a transportation and logistics company. Your company ships products all over the world. Delivery must occur within a specified number of days, depending on the shipping method the customer selects.

There are four distinctly different shipping methods, each with its own delivery deadlines, as shown in Table 2-1.

Table 2-1. Shipping methods and deadlines.

Method	Delivered within X Days
Same Day Air	1
Priority Air	2
Priority Ground	3
Ground	5

You’ve been asked to show how often your company meets these required deadlines per method. There have been some complaints, but so far, no one has analyzed or collected examples to demonstrate that there is a problem.

Before we get too deep, let’s think about expectations. With the office supply store, we came in without an expected answer; we may have had

some institutional or intuitive guesses, but there was no right answer. Here, though, we have a process that should be controlled; the objective is to find examples where that process isn't working as designed.

Blueprint: Creating a basic box plot

Step 1: Create Days to Delivery field

Construct a calculated field that computes the days between the ship date and the delivery date. Call it Days to Delivery (see Figure 2-19).

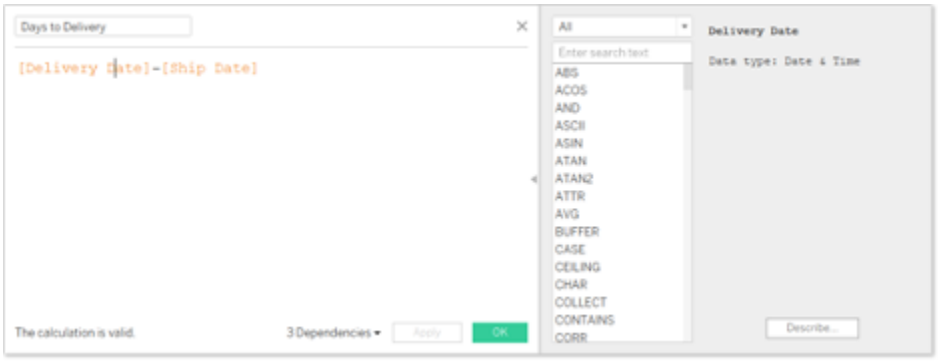


Figure 2-19. Constructing the Days to Delivery field.

Step 2: Place on row shelf

Now drag that field onto the Row Shelf. Notice that when you drag the field, Tableau wil aggregate the data using a measure. We don't want to look at aggregated data, so we can disaggregate the data one of two ways:

1. Go to Analysis -> Uncheck Aggregate Measures; or
2. Drag on Manifest ID. This is the lowest level of detail in the dataset and will construct an aggregate that is equal to the row level value.

Step 3: Include shipping method

Also bring on Ship Method to columns.

Step 4: Finish

Change your mark type to Circle and reduce opacity to 60%. Notice that you again have something very similar to the dot plot, as shown in Figure 2-20.

Figure 2-20. Finished box plot.

To turn this into a box plot, you can go to the Analytics Pane and drag on Box Plot. When you drag it on, select Cell. Tableau has now drawn a box plot on top of your data.

When you hover over the box plot, the summary tooltip shows each of the values mentioned before: the median, the quartiles (labeled as hinges), and the whiskers.

How might you interpret this box plot and explain it to your company?

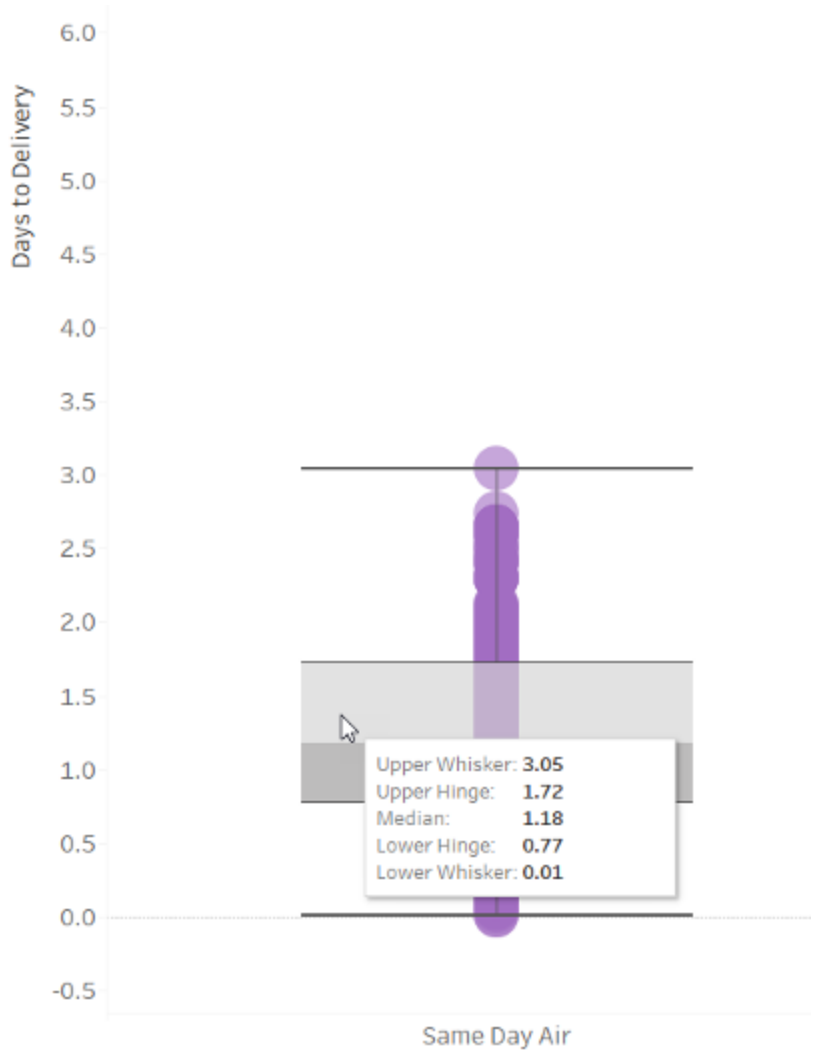


Figure 2-21. Interpreting a box plot.

For Figure 2-21, we would say that the Median time for a Same Day Air delivery is 1.18 days: that is to say, 50% of our Same Day Air deliveries arrive within 1.18 days. Additionally, 25% of our deliveries arrive in less than one day (0.77 days), and 75% arrive in less than 2 days (1.72). It looks like the maximum and upper whisker are at the same point, 3.05 days, so we can actually say that 100% of our Same Day Air deliveries arrive within 3 days. In Tableau we can customize this visualization further, by right-click editing.

Blueprint: Customizing a box plot for presentation

1. You will have two options for how Whiskers extend, both methods which were listed above. See Figure 2-22.
2. Click the box to hide the underlying marks within the box plot.

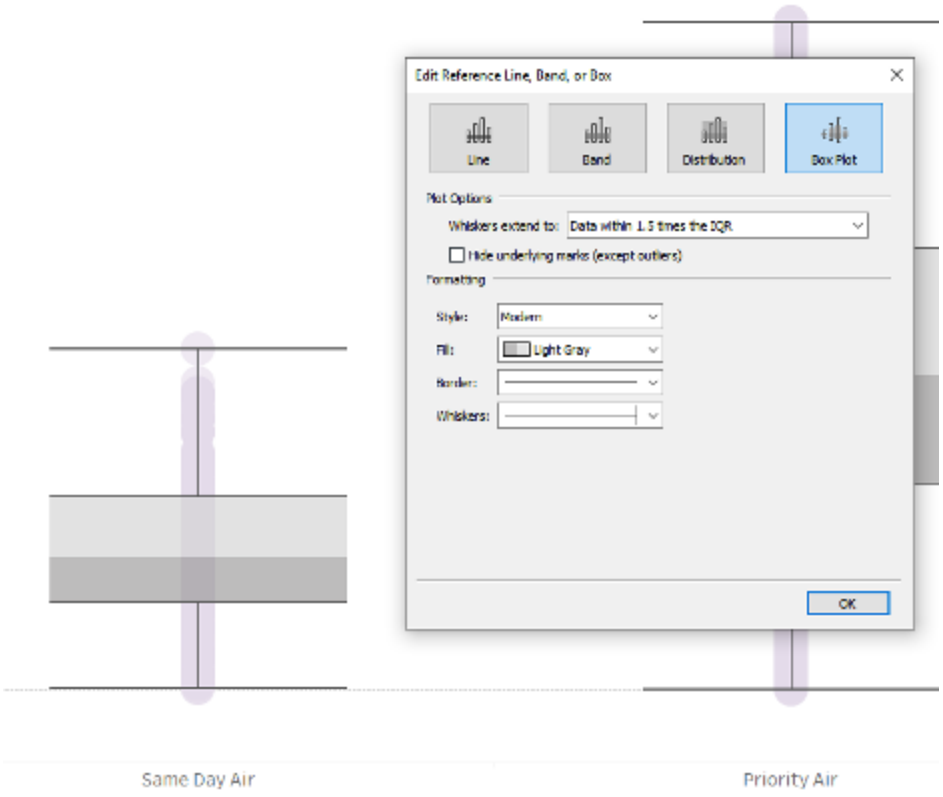


Figure 2-22. Customizing a box plot.

Our updated visualization has reduced the total marks on the screen, now leaving us only with the box plots and any outliers that may exist beyond the Whiskers. You can see the finished version in Figure 2-23.

Figure 2-23. Customized box plot.

Sometimes working with box plots can intimidate your audience.

Knowing that, we usually pair the box plot with a less intimidating chart, like the histogram, to help communicate what the visual represents.

To create this chart, you will start with the building blocks of the box plot you built before.

Blueprint: Combining a box plot and histogram

1. Instead of disaggregating your measures, you will need to aggregate them up. Click on Analysis -> Aggregate Measures.
2. Select Average for Days to Delivery.
3. Drag on Manifest ID to the Marks Card. Figure 2-24 shows what this should look like.

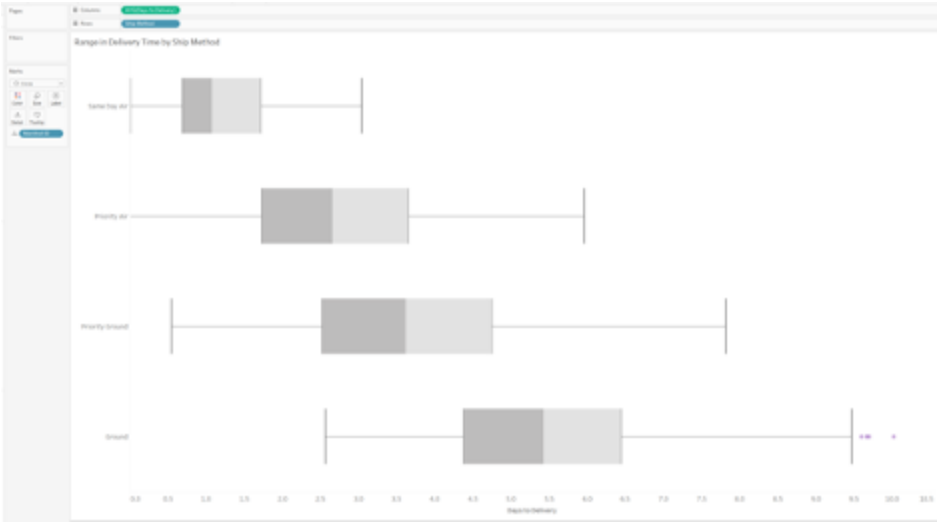


Figure 2-24. Box plot and histogram. Notice that this chart is identical to the one we made with disaggregated measures.

4. Modify the box plot so that the range of the whiskers go to the minimum and maximum.

5. Now that you have this view, you can bring on other measures that will need to be aggregated, specifically as a dual axis. Create an LOD called Days to Delivery by Manifest ID.

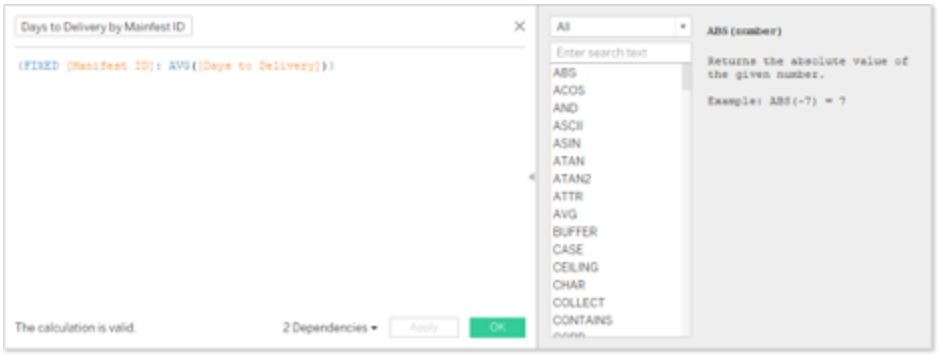


Figure 2-25. Creating an LOD called Days to Delivery by Manifest ID.

6. Create BINs, using the same method that you used when constructing the histogram. Make the BINs at every 0.5 days. Right click on the field in the data pane and change it to continuous.

7. Drag on the new BIN calculation Days to Delivery by Manifest ID (bin) to Columns, to the right of AVG(Days to Delivery).

8. Go to the Marks Card for this visual and change the Mark Type to Bars. Remove Manifest ID from Detail.
9. Drag on the COUNTD(Manifest ID) to Rows.
10. Right click on Days to Delivery by Manifest ID (bin) and select dual axis; right click on the newly created dual axis and synchronize axis.
11. Remove Measure Names from Color on the All Marks Card, and right-click hide the extra axis on the top.
12. Rename the Y axis for clarity, to “# of shipments.”

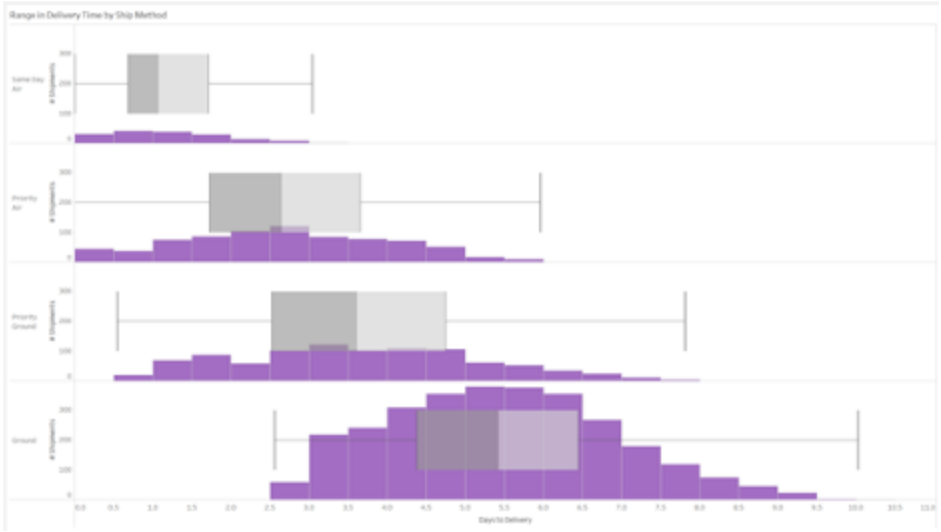


Figure 2-26. Completed box chart and histogram combined.

In Figure 2-26, the audience can see how the construction of the box plot relates to the underlying data. They can see the frequency which is constructing each of the bars, and understand how the data shape presents itself in box plot form. It also provides an even better script for explaining the data. They can see both the frequency and the density of the data at one time.

Line Charts

Like bar charts, line charts are a very popular tool in data visualization. They take a metric and display it over time. Whenever someone asks how a measure is trending, in their mind they've drawn a mental picture of a line chart.

Here we won't dive too deep into working with time (we'll save that for Chapter 4), but we do want to introduce some ways that you can use statistical values and derivative measures to make really impactful line charts.

Office-Supply Store Case Study: Line Charts with Statistical Elements

Consider a basic line chart for our office supply store showing Sales by Month, as shown in Figure 2-27.

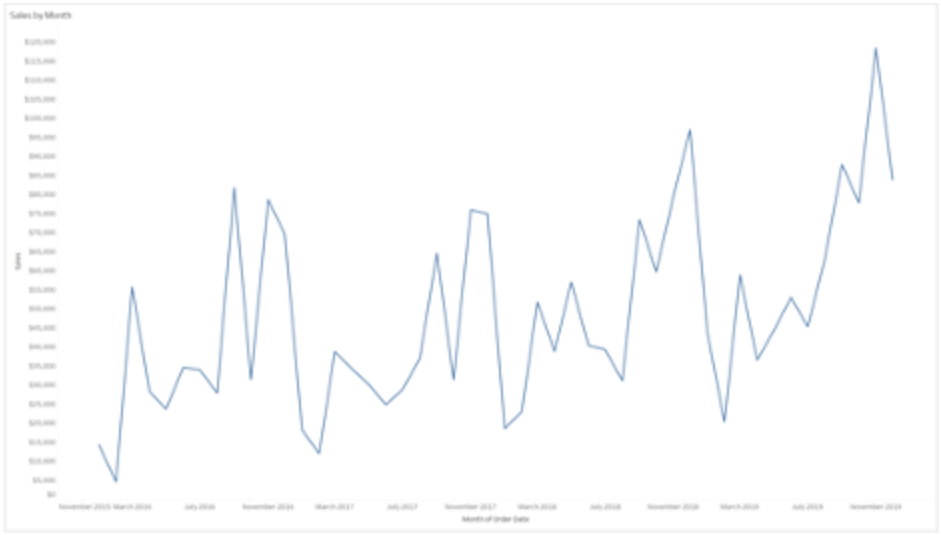


Figure 2-27. Line chart showing sales by month.

This provides some immediate insight. We can tell there is seasonality in the data: every year there are spikes in September, November, and December. And in general, it looks like sales are going up over time.

But we can get more sophisticated in the story that we're telling by including some of the statistical measures we have been exploring.

First let's start with some tools that are built-in and easy to utilize in Tableau. We've explored adding a Reference line for an Average, but let's actually go a step further and add in reference distributions.

Blueprint: Adding reference distributions to a line chart

1. 1. Go to the Analytics Pane and drag on an Average line across the Table.
2. 2. Focusing on the middle section, add a confidence interval in addition to your line, and specify the level of confidence. As you're doing this, change the Label to the Value instead of the computation.

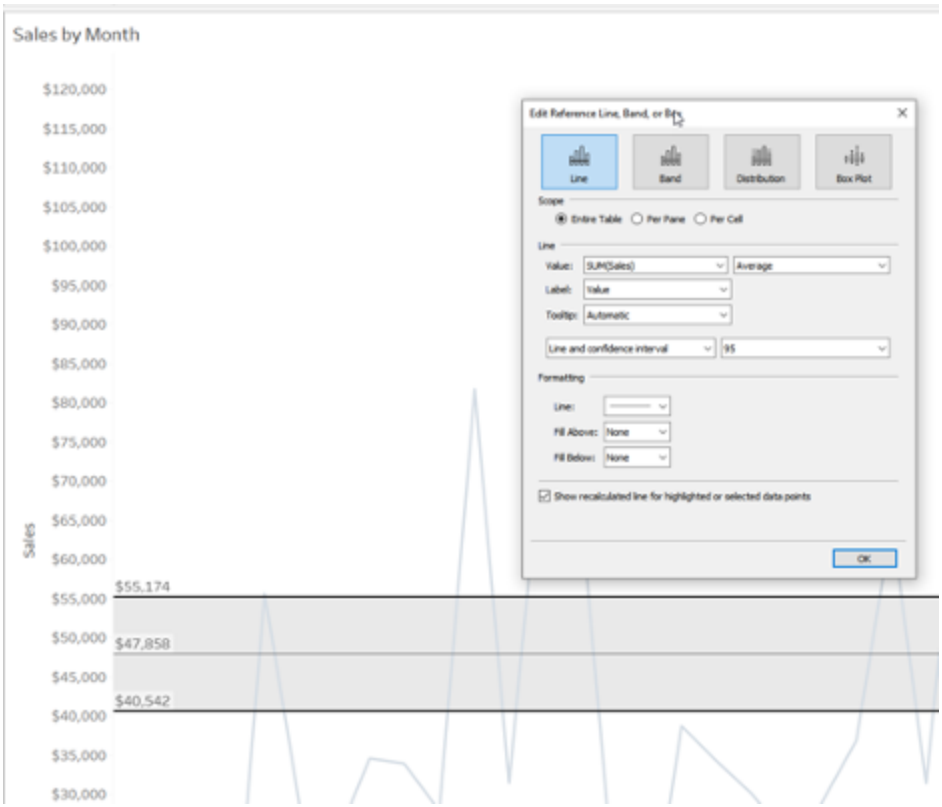


Figure 2-28. Average monthly sales line chart with confidence interval.

The chart, as shown in Figure 2-28, now shows average monthly sales, with a gray band for the confidence interval. A *confidence interval* is a range of values computed from a random sample that is likely to contain the true value of a chosen measure. In this example, the computed monthly average is $47,858 based on the data available (the sample), but we can say with

95% confidence that if we had *all* the values of monthly sales, the average would fall to between $40,542 and $55,174.

It's important to note that the confidence interval is a function of two items of your data: the sample size or number of data points and the overall confidence level. As you increase the sample size of data, your confidence interval is likely to shrink in width, because there are more observations. The confidence level has the inverse effect, the more confident you need to be with your data, say 99% vs. 95%, the wider the band.

Why add this type of context to charts? For this scenario, it is to add detail for your audience, who are relying on the average value that you present. While this shows a computed average of $47k, there is actually a margin of error of ~$7k on what the average could be.

You can also use distribution bands and reference lines in Tableau to create percentiles or standard deviations. Let's focus on Standard Deviation. When you are working with numbers, standard deviation is the spread of the numbers in your dataset from "normal" or the average. In practice, the larger your standard deviation, the more variation there is among your data points, the smaller your standard deviation, the less difference or data spread there is.

Blueprint: Adding a standard deviation to a line chart

1. Right click on the average line and click Edit
2. Change back to "Line Only"
3. Right click on the Axis and let's create another reference line, this time click on Distribution
4. Change the Computation Value to Standard Deviation and leave the factors on -1 and 1. Similarly, leave it on Sample, since we don't have ALL the data, just a subset.

5. Change the Label to Value and click OK.

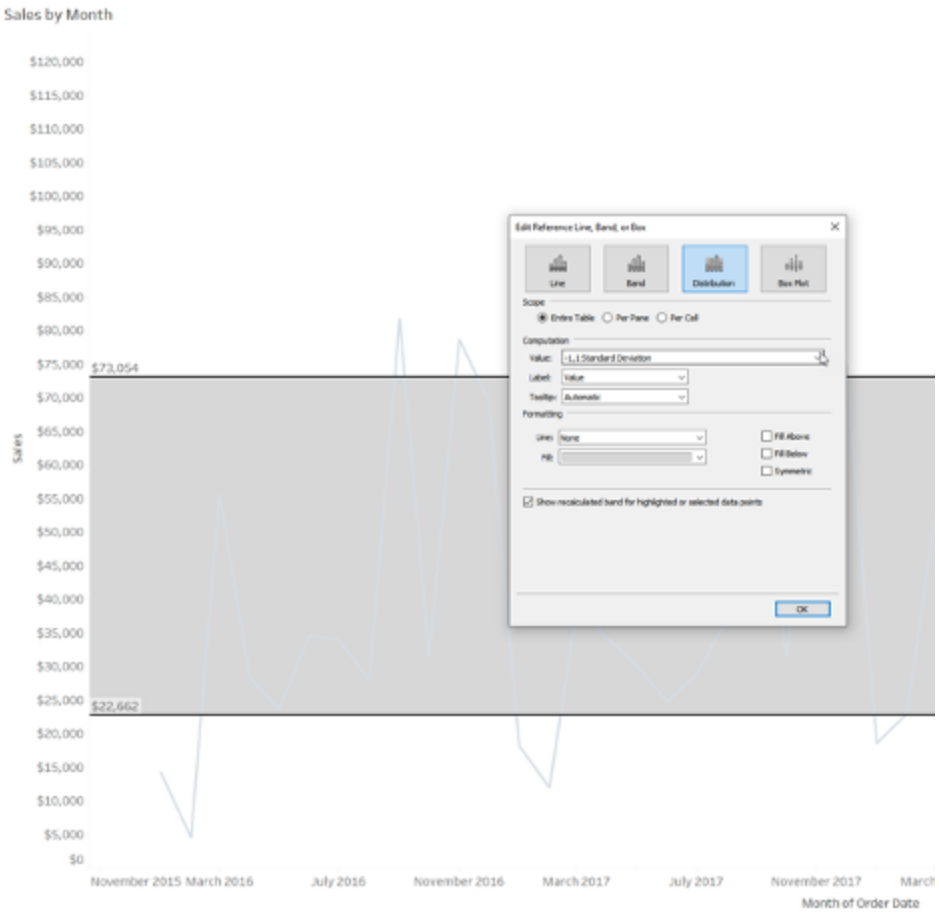


Figure 2-29. Adding a standard deviation to a line chart.

So what can we conclude from the resulting chart (Figure 2-29)? There is a lot of spread from the average. For any given month, we have a range of $25k on what the distance from average might be: not that great when you realize that's nearly half the computed monthly average.

Again thinking about this in context with your audience and why you might want to share: yes, we know what the average monthly sales is, but it would likely not be an accurate guess at what next month's total sales may be since the range of values within 1 standard deviation closely resembles the minimum and maximum of the dataset.

For our last example in this section, we're going to focus on standard deviation, but instead of showing reference distributions, we'll use it as an alerting tactic. In statistics, it's common practice to say that something is

“out of control” when the reported value is outside of a certain number of standard deviations from the average. It’s an unbiased way to review your dataset and see where there truly is meaningful statistical variation. Figure 2-30 shows the chart you are going to build.

Figure 2-30. Line chart with reference distributions.

This chart has monthly sales and the average line, but this time we have added red dots to months that are more than +/- 1 SD from the average.

Blueprint: Building a line chart using reference distributions as an alerting tactic

1. Create a new calculated field called “SD Alert Dot” (see Figure 2-31).
2. This calculation will compare the monthly sales amount against the average of the view plus 1 SD and likewise for minus 1 SD. If the condition is true, then it will return the sales amount. We will then turn that sales amount into a dot.

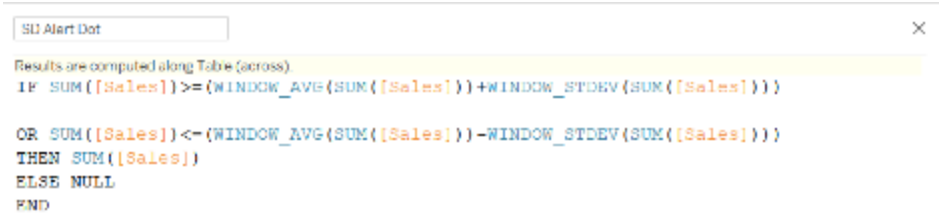


Figure 2-31. Creating a calculated field for SD Alert Dot.

2. When you read this calculated field, it looks like there's a lot going on, but let's simplify the different components.
3. SUM(Sales) is just the sales amount, which will be drawn by month because that's the level of detail you have in your view.
4. WINDOW_AVG(SUM(Sales)) is exactly the same as the Average Reference line: it is the average of all the data points in the view (in the "window").
5. WINDOW_STDEV(SUM(Sales)) is the same concept as the average, but this time the standard deviation.
6. The rest of the conditional is determining if the Sales value is above or below those points. And if it is, you want to return the sales; otherwise, you can null it out.
7. Drag this onto your Rows Shelf
8. Right click and make it a dual axis and synchronize your axes.
9. Change your mark type for the secondary axis to be a dot. Make sure your first mark is still a line.
10. Color the measures appropriately.

So what's the difference between this chart and the previous example with reference bands? Both show the trended metric and standard deviation, but the key difference is how the audience will perceive the charts. With the reference bands, we are presenting them with the standard deviation spread, but with the alert dots, we are inviting the audience to take action.

This is the difference between showing charts with information and being data-driven. In a strong, data-driven organization, alerts or thresholds will be defined with metrics so that the audience knows when to take action vs. the suggestion that something is outside of normal.

Bank Case Study: Using statistics to build new dimensions

While we are on the topic of statistics and working with numerical outputs, let's discuss how to take a statistical output and turn it into a dimension.

Let's go back to our bank example from chapter 1 again. There are a lot of categories, and we need a way to group them and score them based on the total spend. Sure, we have crafted a top N chart before (see Blueprint 1-2), but we want a more holistic way to communicate results for every category in the dataset.

One way to do this would be to assign a summary statistical value to the data point based on what data is in the view (Figure 2-32).

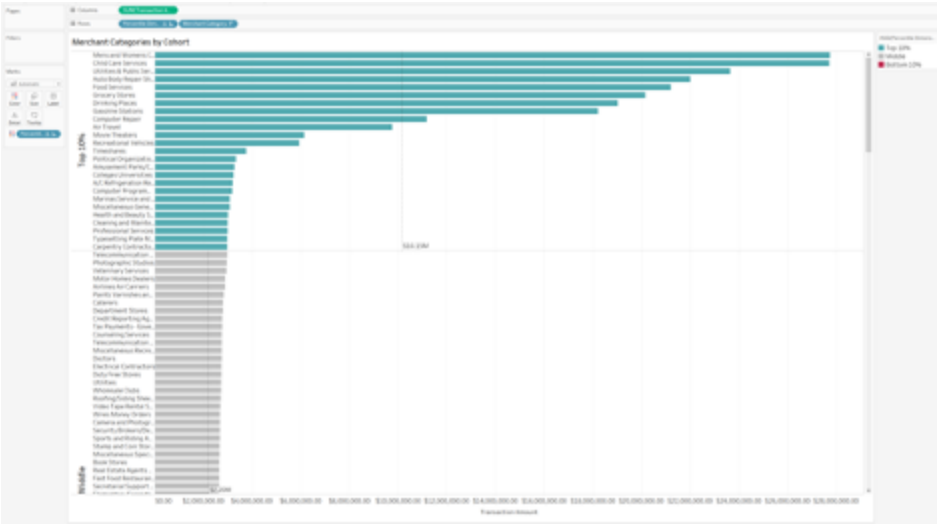


Figure 2-32. Assigning summary statistical value to data points.

Blueprint: Assigning summary statistical value to data points

Step 1: Sort

Start with a new bar chart that has the total transaction amount for each merchant category. Sort the bars in descending order.

Step 2: Categorize

Build out a calculated field that will allow you to categorize our merchants into 3 cohorts: one for the top 10%, one for the bottom 10%, and one that contains the rest of the data.

Breaking apart this calculation, there is really only one essential evaluation: whether the transaction amount is above or below a specific percentile. Remember that here the window_percentile will be the equivalent percentile based on the data in our view.

Drag the calculated field onto Color.

Step 3: Add the calculated field as a header to the view.

This will allow you to create new aggregates, like the average, for the different cohorts. You may have to manually reorder the headers.

There are a few reasons to do this. Now, with mathematical analysis, we have separated out our merchant categories evenly. It also lets us speak in aggregate about those new categorizations. We can add back a reference line that only addresses merchants within a percentile to draw even more pointed insights.

As an example, the average spend in the top 10% of merchant categories is $10.1M. Compare that to those in the bottom 10% of merchant categories whose average spend is a mere $1.75M.

Using statistics to create the different groupings also makes the chart dynamic. If our scope of analysis changes when we filter our data, the subsequent visualization will also update. We could filter the data to look at

transactions completed during a certain time period or even in a specific region and the visual will respond.

Pareto Charts

Up to this point we have talked about how to show categorical data and some of its spread, but we haven't gone deep.

This next visualization is designed to look at the distribution of numeric values that relate back to categorical data. We can even use our example from Chapter 1, the Merchant Categories for consumer transactions.

Recall that we discussed various ways to show which Merchant Categories took up a significant portion of consumer spend. We went through and found a dynamic top N value, and we bucketed those outside of a percentage into an "Other" category.

Have you ever heard of the 80/20 rule? Often called the Pareto principle, suggests that 80% of a metric or outcome is related to 20% of the data.

To see that more directly in a visualization, we can construct a Pareto Chart. A Pareto chart is a dual axis chart that takes the initial sorted bar chart, and adds on a running percent of total line. The running percent of total is what makes up the 80/20 rule.

Pareto charts are great when you need to help the audience focus on what is important.

Office-Supply Store Case Study: Helping the Audience Focus with Pareto Charts

We can start by building a basic bar chart, this time vertical. Then we will turn it into a Pareto chart. If we were to apply the Pareto Principle directly to our use case, we would find that 20% of Merchant Categories make up 80% of consumer spend.

Blueprint: Using Pareto charts to show categorical data

1. Drag Merchant Category on to Columns.
2. Drag Transaction Amount onto Rows.
3. Use the toolbar to sort the Merchant Categories in Descending Order by Transaction Amount (shown in Figure 2-33).

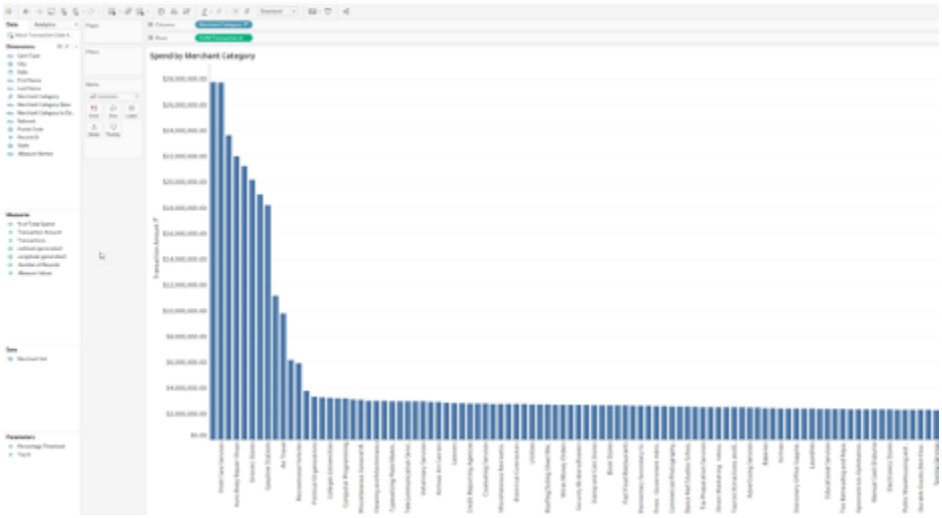


Figure 2-33. Sorting merchant categories in a Pareto chart.

4. Hold down your CTRL key and drag SUM(Transaction Amount) as a copy onto your Rows shelf
5. Right click on the new measure, and create a Quick Table Calculation, use the Running Total.
6. Right-click on the Table Calculation again and Edit it. This time click on “Add secondary calculation” and select Percent of Total. This takes the running total and presents it as a cumulative total percentage (shown in Figure 2-34).

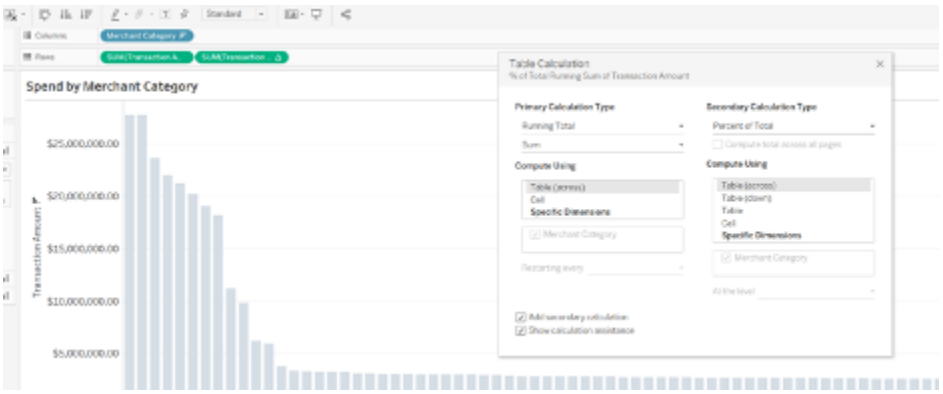


Figure 2-34. Table calculation for a Pareto chart.

7. Change your mark type for the second metric to a line and then right-click on it to make it a dual axis chart (shown in Figure 2-35).

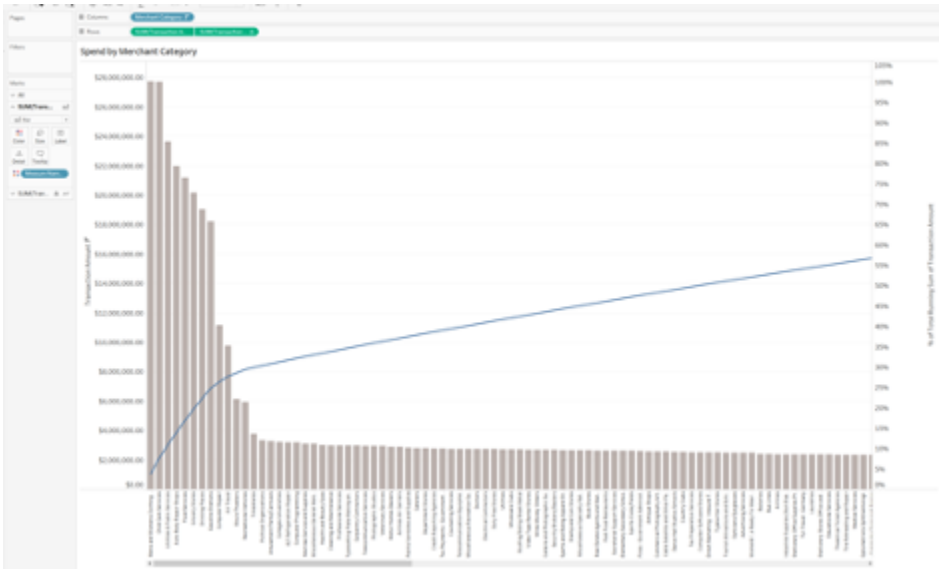


Figure 2-35. Dual-axis chart.

8. Clean up the view by editing your axis labels.

9. You can also add a parameter to this to highlight exactly where 80% of the data is.

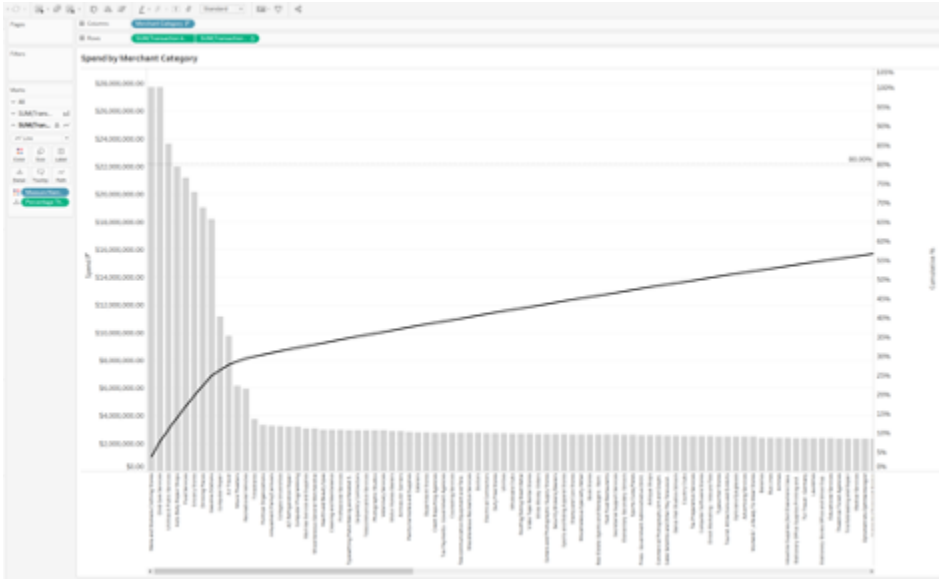


Figure 2-36. Figure 2-36. Completed Pareto chart.

So did the data follow the 80/20 principle? Not in this case. You probably had this suspicion from our work in Chapter 1, but in this scenario, while there are certainly a few Merchant Categories that dominate Spend, it is not the case that 20% of them make up 80% of Consumer Spend.

Conclusion

Throughout this chapter you've seen several ways to work with numeric data. We've explored how to manage the spread of data, as well as the different statistics you may want to pair with your analyses. We've discussed how it is critical to start at the most basic level of data and work your way up before providing aggregates. We've also explored ways to leverage statistical values for your data to create new dimensions and therefore new ways to categorize your data.

Chapter 3. Comparisons

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 3rd chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the authors at ann@jacksontwo.com and luke.stanke@tessellationconsulting.com.

Comparisons are at the heart of data visualization. Our audiences are often laser-focused on comparing two or more values. Our job as practitioners is to simplify the comparison of each set of values.

Some comparisons are easier than others. If you're comparing two groups on a single value, then your comparison is straightforward: use a bar chart. Most times you should just use a bar chart simply because bar charts are the most versatile and common chart type used for comparisons.

But there is one challenge with all bar charts: it's a bar chart! Your audience has seen this chart type hundreds, thousands, or millions of times. In this chapter we'll offer you a few alternatives to the bar chart, such as Cleveland dot plots and lollipop charts, as well as some different ways to format your bar charts.

When creating these alternative charts, really think about whether you're seeking a bar chart for your audience's needs or for your own. If it's for you: keep it a bar chart. If it's for your audience: perfect, use the alternative.

Not all comparisons are straightforward as bar charts. Sometimes we are comparing across members a dimension and at different time points within that member. Other times we’re comparing multiple members against multiple dimensions. Sometimes we’re worried about the actual metric, sometimes the rank of the value. Sometimes we’re interested in comparing values not against other members, but against the overall group.

In this chapter we’ll provide some other methods for comparison including the bar-on-bar chart, displaying the change in rank between two time periods, bump charts, barbell charts, trellis, and parallel coordinates.

What you'll learn in this chapter

- How to format bar charts so they have a fresh/unique look-and-feel.
- How to build a lollipop chart (and some of their drawbacks).
- How to build a Cleveland dot plot
- How to show change in year-over-year on a single bar chart
- How to build a bump chart
- How to build a barbell chart
- How to build a trellis chart or create small multiples
- How to create a parallel coordinates chart

What you'll need

In this chapter you will work with two different data sources: the non-profit dataset and the retail sales dataset (Sample – Superstore).

Bar Charts and Alternatives

As we've already discussed, data visualization practitioners create a lot of bar charts–justifiably so. As you learned in chapter 1, a bar chart compares two or more groups across a single metric. Bar charts are easy to use because you only have to compare the length or height of one bar against another. With a quick glance your audience understands the magnitude of the difference.

Nonprofit Case Study: Comparing Grant Sizes

Let's imagine we (that's you, reader, and your authors) work together for a nonprofit. We're trying to understand our grants. We want to know how the average grant size compares based on the type of grant. Our boss has shown

the same chart to funders for the last four years. This year, she's looking for different ways to spice it up!

In the grant data we have a group called Budget Category that we will use to compare the data. We'll show you how to create a measure called Avg Grant Amount. You will use this group and measure for five different examples throughout the chapter.

Blueprint: Draw comparisons with a basic bar chart

1. Connect to the Community Grant data.
2. Create a calculated field called Avg Grant Amount**.**
3. Use existing fields and write the following formula: `// Avg Grant Amount`
   `SUM([Grant Amount])/COUNT([Record ID])`
4. Rotate the labels on the columns shelf so the text is vertical.

We started by creating this vertical bar chart because almost everyone learns how to make a bar chart by plotting it vertically. This is often reinforced when we use document editors or spreadsheet tools, where the default settings are vertical bar charts. The alternative: a horizontal bar chart. Now let's take a look at two versions of bar charts: one where the bars are vertical and the other where the bars are horizontal. Then we'll discuss why. But first, let's dissect the vertical bar chart in Figure 3-1.

Here, we can see that Budget Category is on columns and Avg Grant Amount is on rows.

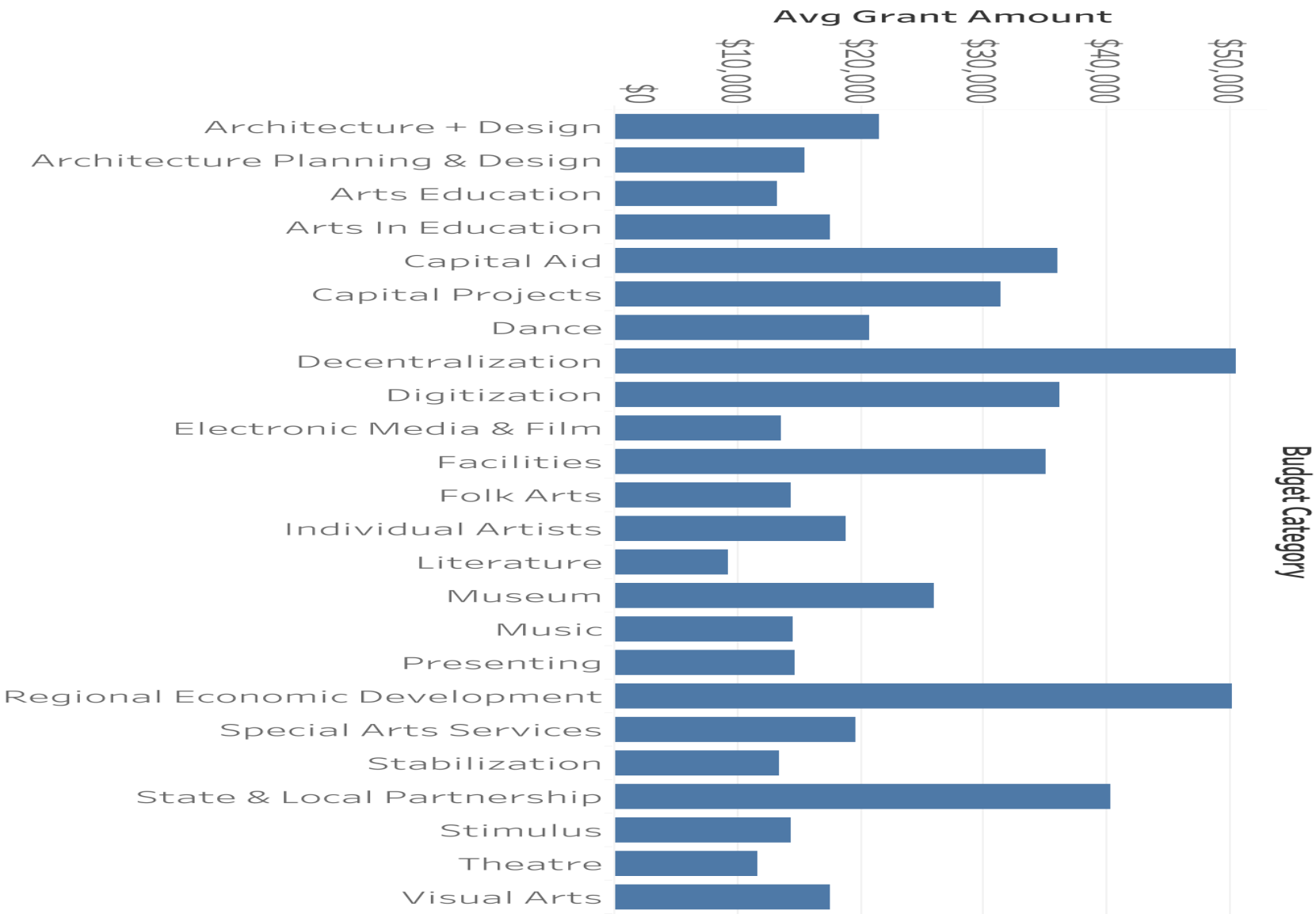

Avg Grant Amount
$0
$10,000
$20,000
$30,000
$40,000
$50,000
Architecture + Design
Architecture Planning & Design
Arts Education
Arts In Education
Capital Aid
Capital Projects
Dance
Decentralization
Digitization
Electronic Media & Film
Facilities
Folk Arts
Individual Artists
Literature
Museum
Music
Presenting
Regional Economic Development
Special Arts Services
Stabilization
State & Local Partnership
Stimulus
Theatre
Visual Arts
Budget Category

Figure 3-1. Vertical bar chart depicting the average grant amount by category.

Now look at this chart through our boss's eyes. At first glance, the vertical bar chart Tableau produced looks pretty good. You can identify Decentralization and Regional Economic Development funds as the two categories with the largest average grant amounts.

When you start to dig into the comparisons, however, it becomes harder to determine the order of the groups. While alphabetical order is great, it's not that meaningful. Additionally, to read the labels you need to bend your neck slightly. If only there was a better way.

We recommend horizontal bar charts over vertical bar charts. It's completely okay if you made a lot of vertical bar charts before reading this. It's not your fault that your grade-school teachers made you create vertical bar charts for all those math and science projects. Many of us read from left to right, especially if we grew up speaking a left-to-right language, so it's easy for us to process text and information the same way. This means choosing horizontal bar charts over vertical bars. (Remember, that doesn't mean you have to give the vertical ones up entirely!)

Blueprint: Converting a vertical bar chart to horizontal

For our next example you are going to make a horizontal bar chart.

1. Duplicate the sheet you made in the previous Blueprint.
2. Set the sheet to fill the entire view.
3. Click the Swap Axes button to swap rows and columns.
4. If necessary, rotate the labels on rows.
5. Sort the Budget Category descending by Average Grant Amount.
6. Right click on the visualization and select format. Add row dividers use the 2nd lightest gray option in Tableau's color palette

to match Figure 3-2.

7. In the format pane on the Line button (5th of 5 options), remove all gridlines, zerolines, and axis ruler by setting the values to None.
8. Size the bars to be about 50% of the maximum value.
9. Click on label on the marks card and check Show marks label.
10. Right-click on the Avg Grant Amount Axis and select Show Header to uncheck and hide the header.

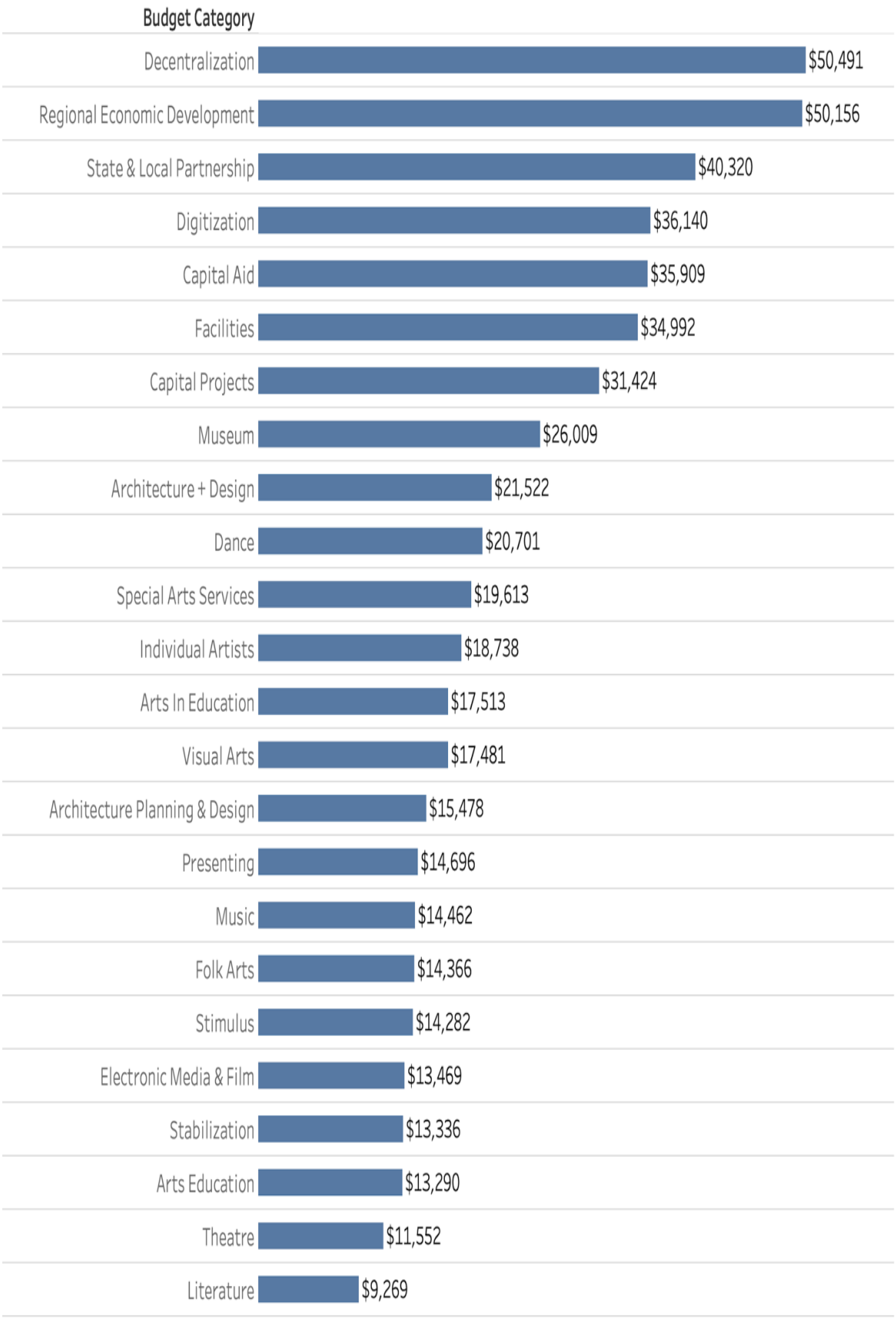

Budget Category
Decentralization $50,491
Regional Economic Development $50,156
State & Local Partnership $40,320
Digitization $36,140
Capital Aid $35,909
Facilities $34,992
Capital Projects $31,424
Museum $26,009
Architecture + Design $21,522
Dance $20,701
Special Arts Services $19,613
Individual Artists $18,738
Arts In Education $17,513
Visual Arts $17,481
Architecture Planning & Design $15,478
Presenting $14,696
Music $14,462
Folk Arts $14,366
Stimulus $14,282
Electronic Media & Film $13,469
Stabilization $13,336
Arts Education $13,290
Theatre $11,552
Literature $9,269

Figure 3-2. Horizontal bar charts are easier to interpret because the text can be read left to right.

Horizontal bar charts open up opportunities for labeling your bars. You could include a label or add additional context on the bars, like change versus a previous period, percent of a total, or rank vs. other members.

If you choose to add labels, you can hide the axis. Axes and labels accomplish the same goal: they give scale to the bar size. Including both means including redundant information, and as a practitioner you want to minimize redundancy. If you do find your audience often misinterpreting information, you may want to include redundancies, provide additional context in text form, or choose a different visualization.

For your horizontal bar chart example, you've chosen to add labels, so we had you hide your axis and removed gridlines, grid rulers, and all other line types included in Tableau's formatting pane. We also recommend you sort your values on the metric. It's Luke's personal preference to also increase the whitespace between each bar and include row dividers.

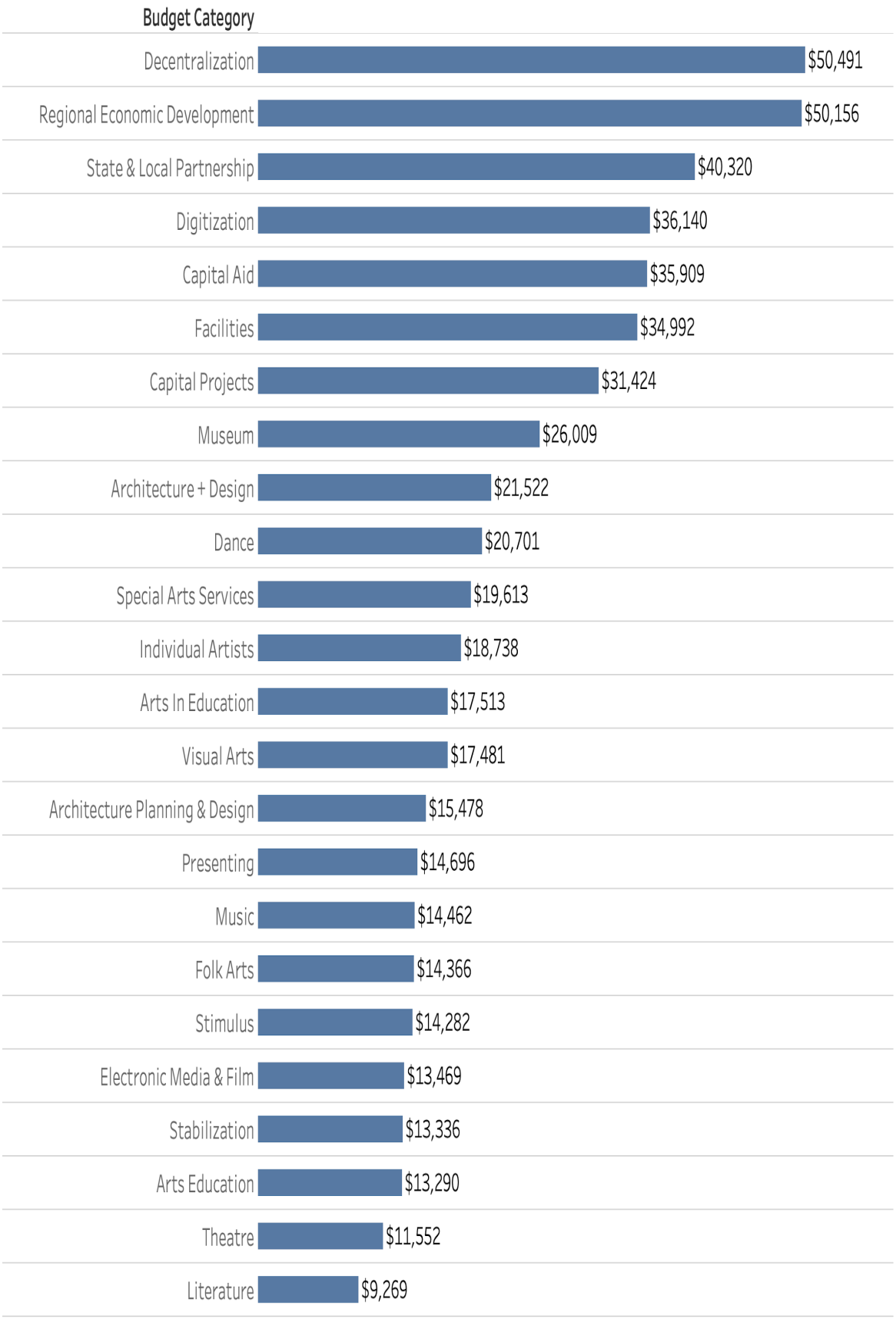

Budget Category
Decentralization $50,491
Regional Economic Development $50,156
State & Local Partnership $40,320
Digitization $36,140
Capital Aid $35,909
Facilities $34,992
Capital Projects $31,424
Museum $26,009
Architecture + Design $21,522
Dance $20,701
Special Arts Services $19,613
Individual Artists $18,738
Arts In Education $17,513
Visual Arts $17,481
Architecture Planning & Design $15,478
Presenting $14,696
Music $14,462
Folk Arts $14,366
Stimulus $14,282
Electronic Media & Film $13,469
Stabilization $13,336
Arts Education $13,290
Theatre $11,552
Literature $9,269

Figure 3-3. The finished horizontal bar chart.

Now let's once again look at the finished chart (Figure 3-4) through the boss's eyes. We can quickly see that Decentralization and Regional Economic Development are the two categories with the largest average grant amounts. We can also see that State & Local Partnership is next, with an average grant size approximately $10,000 lower. We see that the majority of categories have an average grant size between $14,000 and $20,000. Finally, we see that Arts Education, Theater, and Literature have the smallest average grant sizes. Of course, all this information is available in the vertical bar chart, too, but with our horizontal bar chart we can get to it faster.

Now let's talk about some alternatives that help your audience break up the routine of seeing bar charts. In this next section we'll show you how to create lollipop charts and single-point, or Cleveland, dot plots.

One of the easiest ways to jazz up a bar chart is to create a lollipop chart (Figure 3-4). A lollipop chart is simply a bar chart with a circle at the end of the bar. For whatever reason–perhaps it's the rounded end points–audiences and developers love this chart type.

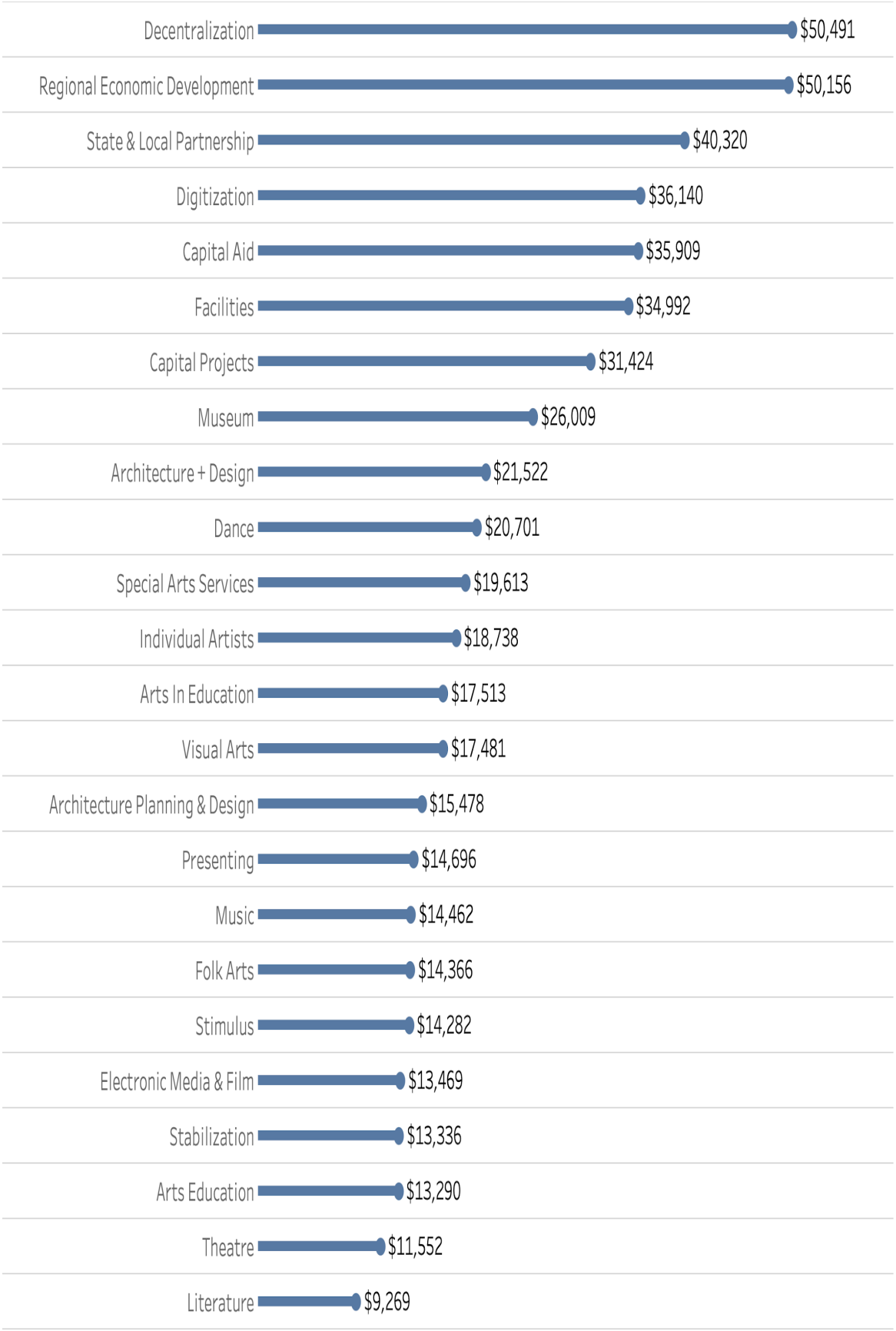

Decentralization $50,491
Regional Economic Development $50,156
State & Local Partnership $40,320
Digitization $36,140
Capital Aid $35,909
Facilities $34,992
Capital Projects $31,424
Museum $26,009
Architecture + Design $21,522
Dance $20,701
Special Arts Services $19,613
Individual Artists $18,738
Arts In Education $17,513
Visual Arts $17,481
Architecture Planning & Design $15,478
Presenting $14,696
Music $14,462
Folk Arts $14,366
Stimulus $14,282
Electronic Media & Film $13,469
Stabilization $13,336
Arts Education $13,290
Theatre $11,552
Literature $9,269

Figure 3-4. : A lollipop chart is a nice break from looking at bar charts–even though it technically is a bar chart.

Blueprint: Creating a lollipop chart

Step 1: Create your chart

Create a new sheet.

Step 2: Make it a dual-axis chart.

Create a dual-axis chart using the same dimensions–in this example, using Avg Grant Amount. This will give you two Marks Cards you can control. If you haven't made a dual axis chart, here's how:

Step 2a. Add Average Grant Amount to the columns axis twice, as shown in Figure 3-4.

Step 2b. Right click on the right measure on columns and select Dual Axis.
Step 2c. On the top axis on the view, right click and select Synchronize Axis.

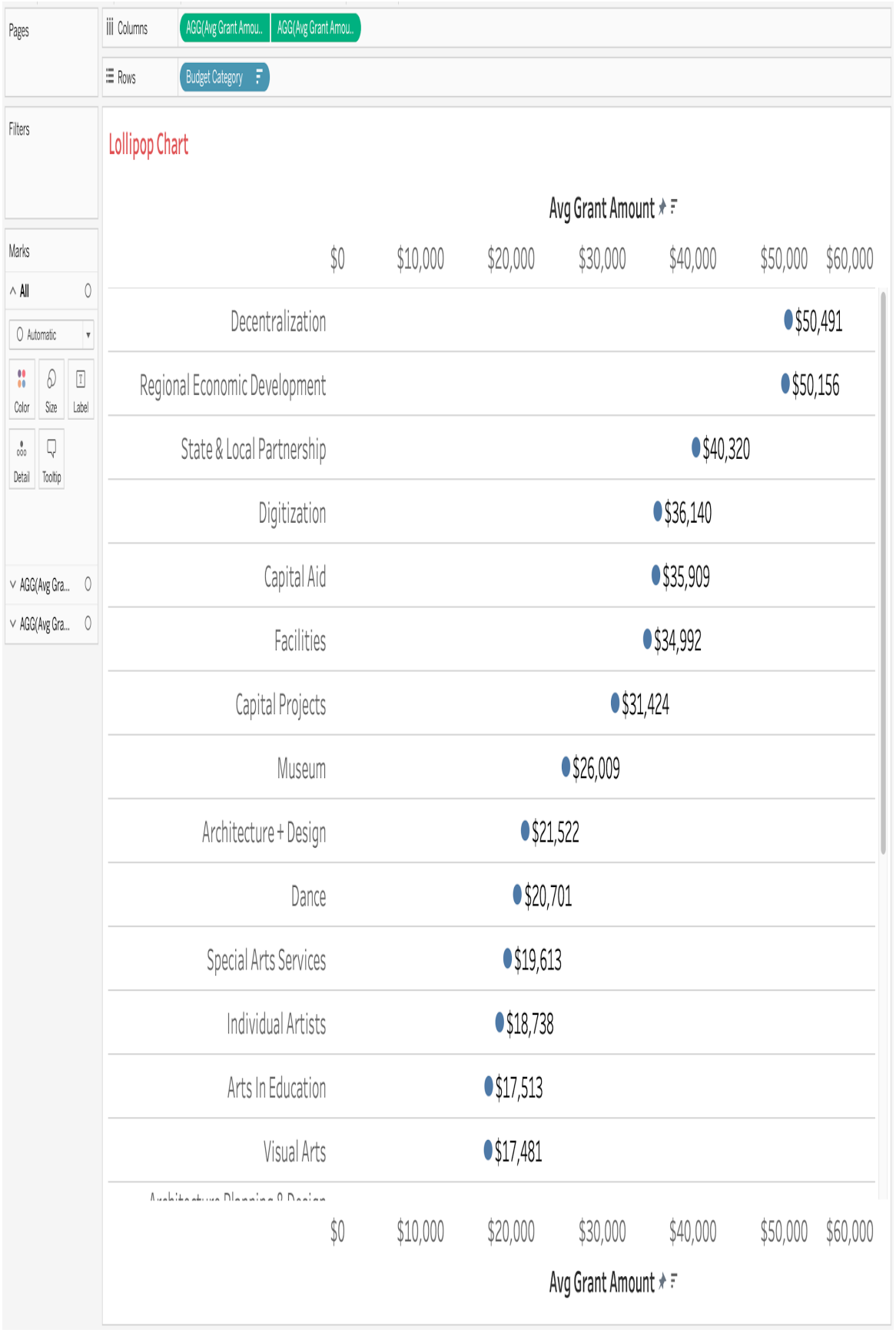

Pages
Columns
AGG(Avg Grant Amou..
AGG(Avg Grant Amou..
Rows
Budget Category
Filters
Lollipop Chart
Avg Grant Amount
$0
$10,000
$20,000
$30,000
$40,000
$50,000
$60,000
Marks
All
Automatic
Color
Size
Label
Detail
Tooltip
AGG(Avg Gra...
AGG(Avg Gra...
Decentralization
$50,491
Regional Economic Development
$50,156
State & Local Partnership
$40,320
Digitization
$36,140
Capital Aid
$35,909
Facilities
$34,992
Capital Projects
$31,424
Museum
$26,009
Architecture + Design
$21,522
Dance
$20,701
Special Arts Services
$19,613
Individual Artists
$18,738
Arts In Education
$17,513
Visual Arts
$17,481
$0
$10,000
$20,000
$30,000
$40,000
$50,000
$60,000
Avg Grant Amount

Figure 3-5. The first step to a lollipop chart.

Step 3: Adjust the marks card

Step 3a. Click on the left-most Avg Grant Amount on columns. Change the mark type to bar.

Step 3b. Change the size of the bar to be smaller than the circle, as shown in Figure 3-5.

Step 3c. Click on the rightmost Avg Grant Amount measure on columns. Change the mark type from Automatic to circle.

Step 3d. Show the mark labels on this marks card and change the alignment to be right-middle. Adjust the circle size if necessary.

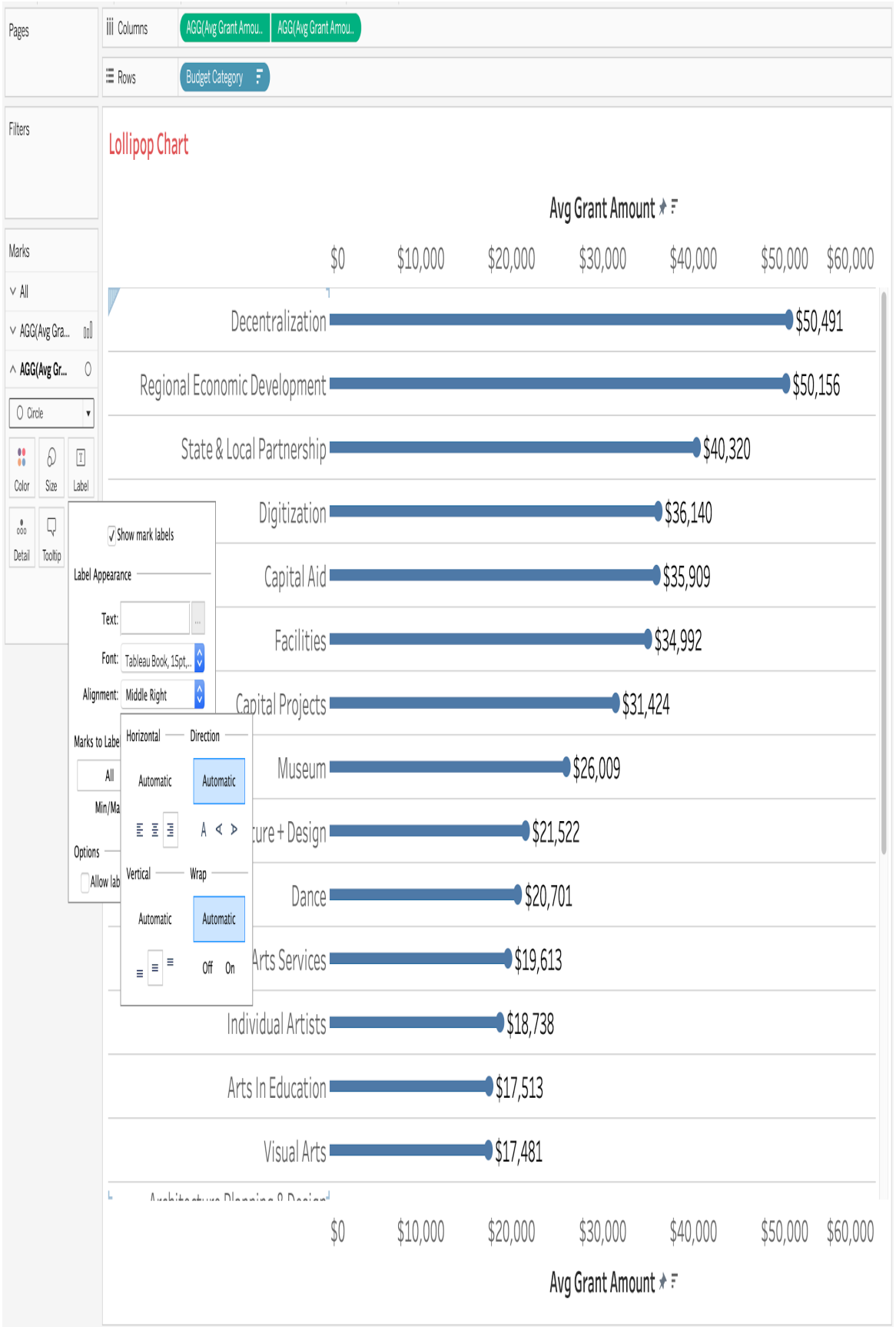

Pages
Columns
AGG(Avg Grant Amou..
AGG(Avg Grant Amou..
Rows
Budget Category
Filters
Lollipop Chart
Avg Grant Amount
$0
$10,000
$20,000
$30,000
$40,000
$50,000
$60,000
Marks
All
AGG(Avg Gra...
AGG(Avg Gr...
Circle
Color
Size
Label
Detail
Tooltip
Show mark labels
Label Appearance
Text:
Font:
Tableau Book, 15pt,...
Alignment:
Middle Right
Marks to Label
All
Min/Ma
Options
Allow lab
Horizontal
Direction
Automatic
Automatic
Vertical
Wrap
Automatic
Automatic
Off
On
Decentralization
$50,491
Regional Economic Development
$50,156
State & Local Partnership
$40,320
Digitization
$36,140
Capital Aid
$35,909
Facilities
$34,992
Capital Projects
$31,424
Museum
$26,009
ture + Design
$21,522
Dance
$20,701
Arts Services
$19,613
Individual Artists
$18,738
Arts In Education
$17,513
Visual Arts
$17,481
$0
$10,000
$20,000
$30,000
$40,000
$50,000
$60,000
Avg Grant Amount

Figure 3-6. Create a synchronized dual axis with a bar and a circle to make a dual axis chart. Be sure to add labels to the circle, not the bar.

Step 4. Format

Hide your headers. Remove gridlines, zero lines and axis rulers, and add row dividers.

This produces the visualization in Figure 3-6.

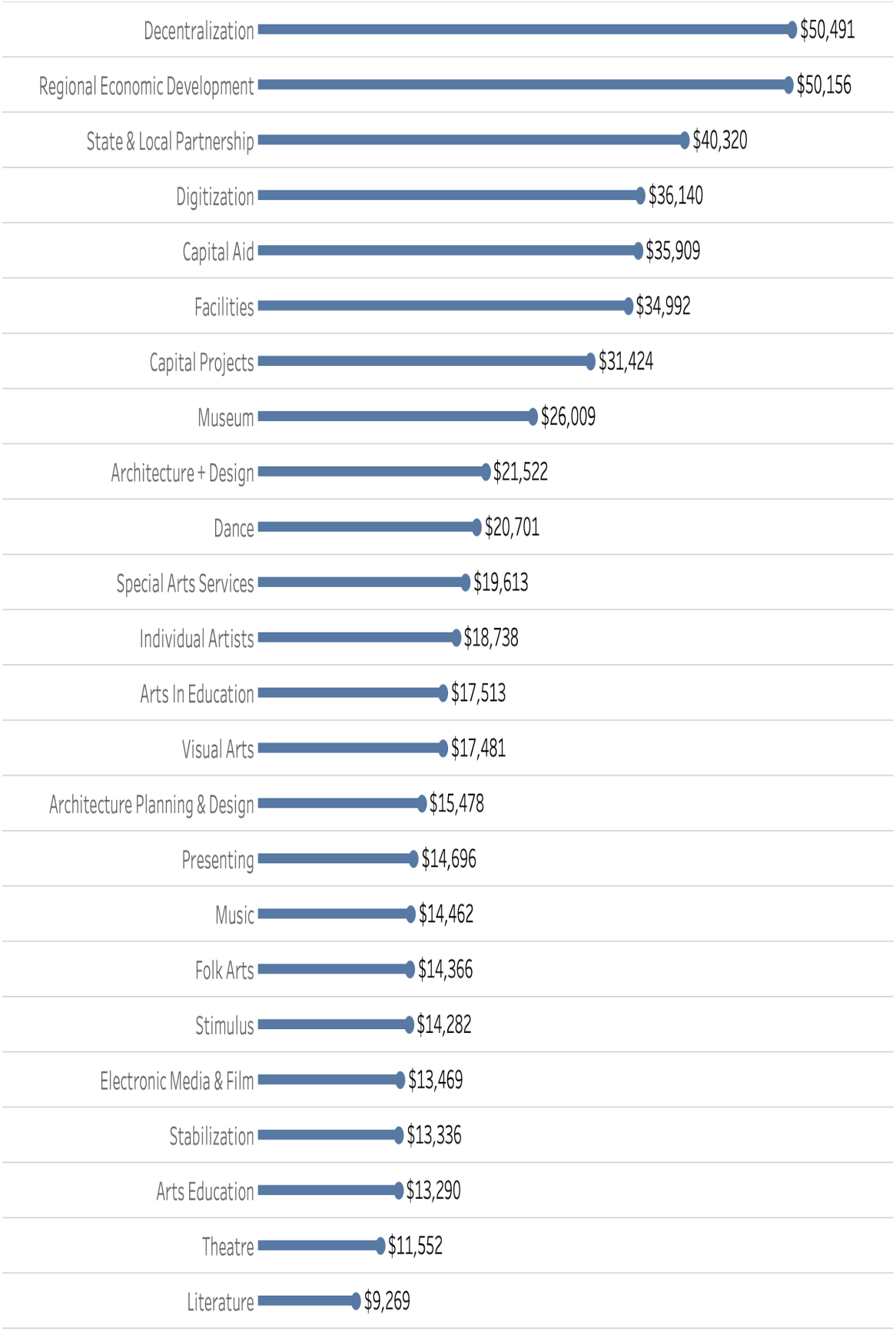

Decentralization $50,491
Regional Economic Development $50,156
State & Local Partnership $40,320
Digitization $36,140
Capital Aid $35,909
Facilities $34,992
Capital Projects $31,424
Museum $26,009
Architecture + Design $21,522
Dance $20,701
Special Arts Services $19,613
Individual Artists $18,738
Arts In Education $17,513
Visual Arts $17,481
Architecture Planning & Design $15,478
Presenting $14,696
Music $14,462
Folk Arts $14,366
Stimulus $14,282
Electronic Media & Film $13,469
Stabilization $13,336
Arts Education $13,290
Theatre $11,552
Literature $9,269

Figure 3-7. The final result showing the average grant amount by category using a lollipop chart.

Audiences and developers love lollipop charts. It’s a subtle way to take a break from bar charts and it doesn’t take a lot of work.

One drawback of lollipop charts: They are not a 100% accurate representation of the data. The end of the bar chart is the accurate representation of the data, and then we tack a circle onto the end of the bar. The center of the circle is aligned with the center of the bar. This means that half of the circle misrepresents the data (see Figure 3-7).

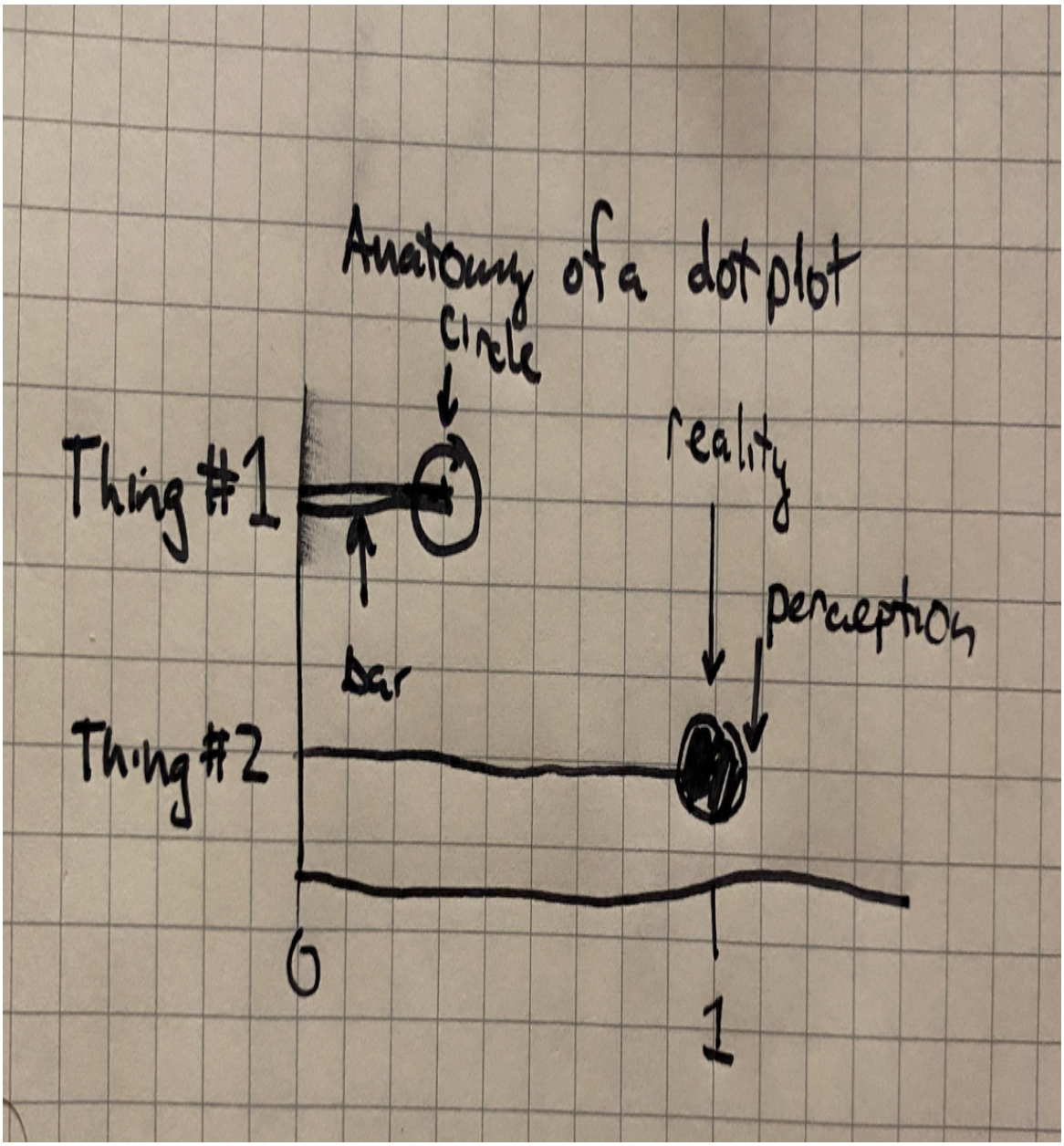


Figure 3-8. Lollipop charts are not pixel-perfect when it comes to creating a 100% accurate depiction of the visualization, but that's okay.

Here's our take on this inaccuracy: It is our job to create visualizations that approximate the data. Most chart types are already approximations of the data. If this level of abstraction would change readers' interpretation of the data, don't use it. If it's still a close approximation, feel free to use it. Just be ready to justify your decision!

Nonprofit Case Study: The Cleveland Dot Plot

As you know, our boss at the nonprofit is looking for alternatives to bar charts. Luke's personal favorite is the Cleveland dot plot. He likes to use it because it increases the spacing of the data elements and de-emphasizes non-data elements while still providing the same information as a bar chart.

The Cleveland dot plot is not from the city of Cleveland, Ohio; rather, it was a design published by two statisticians, William Cleveland and Robert McGill, in 1984. It shows values with circles along an axis, then displays the value of each group at the end of the axis (Figure 3-9).

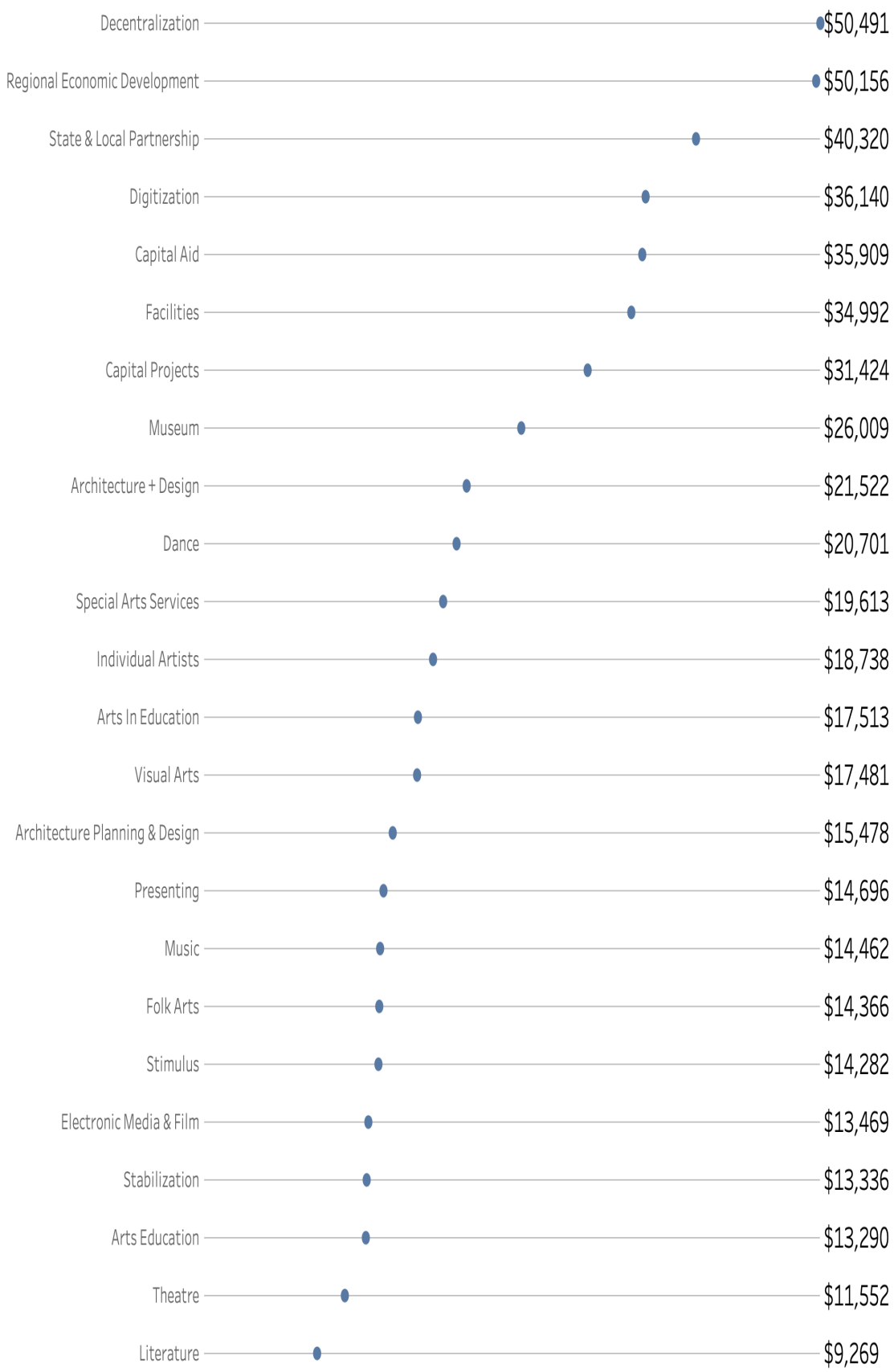

Decentralization
$50,491
Regional Economic Development
$50,156
State & Local Partnership
$40,320
Digitization
$36,140
Capital Aid
$35,909
Facilities
$34,992
Capital Projects
$31,424
Museum
$26,009
Architecture + Design
$21,522
Dance
$20,701
Special Arts Services
$19,613
Individual Artists
$18,738
Arts In Education
$17,513
Visual Arts
$17,481
Architecture Planning & Design
$15,478
Presenting
$14,696
Music
$14,462
Folk Arts
$14,366
Stimulus
$14,282
Electronic Media & Film
$13,469
Stabilization
$13,336
Arts Education
$13,290
Theatre
$11,552
Literature
$9,269

Figure 3-9. : A Cleveland dot plot depicting the average grant amount by category.

Blueprint: Creating a Basic Cleveland Dot Plot

Step 1: Build the visualization

Create a new sheet.

Step 2: Create rows and columns

Add Budget Category to rows. Like you did in the lollipop chart, add Avg Grant Amount to columns twice and create a synchronized dual axis.

Step 3: Modify Avg Grant Amount

Double-click **o**n the leftmost Avg Grant Amount measure on columns and edit the calculation. Wrap Avg Grant Amount in the WINDOW_MAX() function. This is a table calculation and will return the maximum value across all Avg Grant Amounts. (See Figure 3-9.)

Edit the table calculation to run on Table (down).

Figure 3-10. Creating an ad-hoc calculation WINDOW_MAX() calculation on columns.

Step 4: Customize your bars

Step 4a: Mark Type. Change the mark type from Automatic to Bar. Change the size of the bar to the smallest value possible. We don't want the bar to stand out too much, so change its color to a light gray.

Step 4b: Labels. Add Avg Grant Amount to labels. This gives you what you see in Figure 3-10.

Step 4c: Formatting. If you want to make your background bar size even narrower, you can click on color on the marks card and set the border color to match the background color.

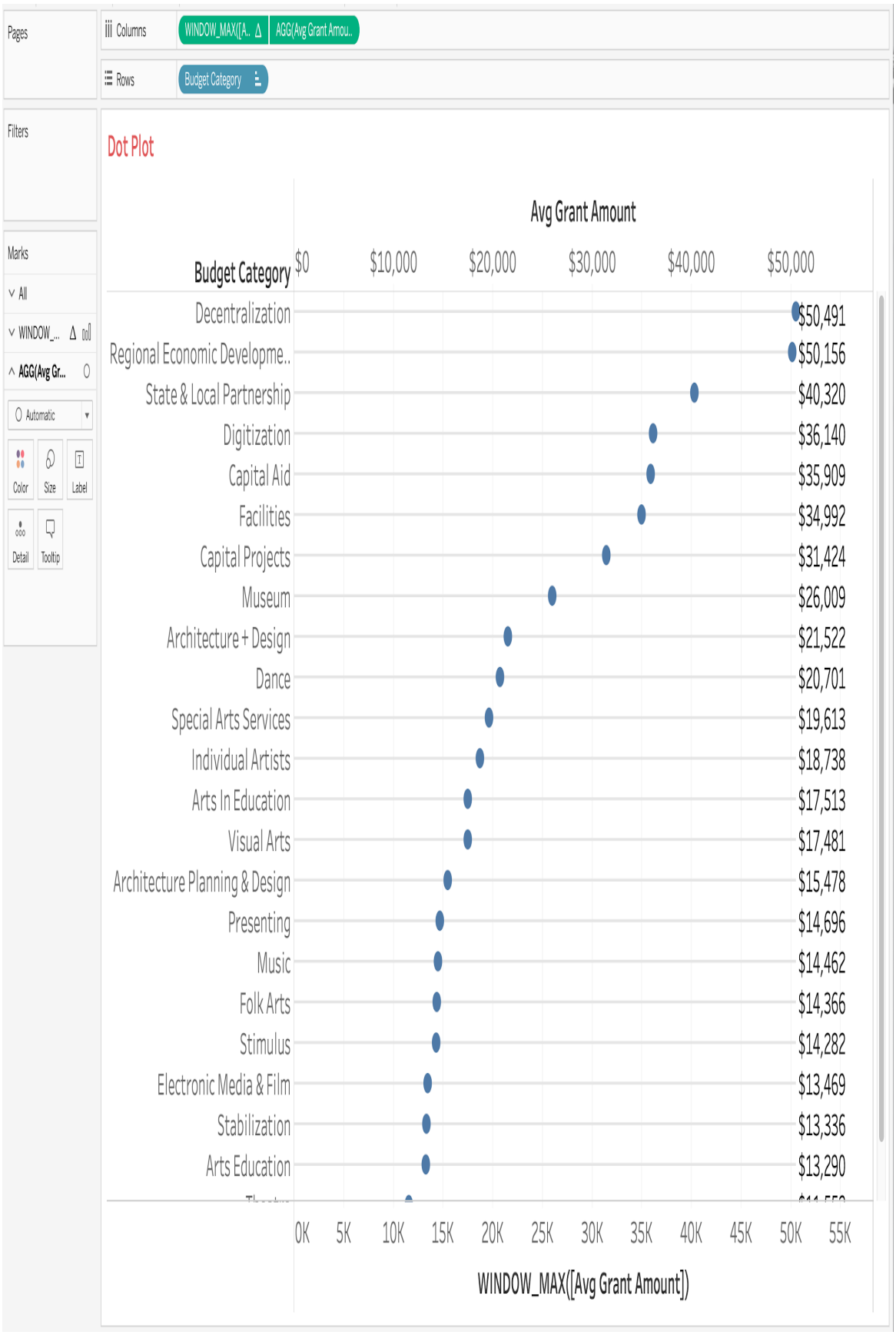

Pages
Columns
WINDOW_MAX([A..
AGG(Avg Grant Amou..
Rows
Budget Category
Filters
Dot Plot
Marks
All
WINDOW_...
AGG(Avg Gr...
Automatic
Color
Size
Label
Detail
Tooltip
Avg Grant Amount
Budget Category
$0
$10,000
$20,000
$30,000
$40,000
$50,000
Decentralization
Regional Economic Developme..
State & Local Partnership
Digitization
Capital Aid
Facilities
Capital Projects
Museum
Architecture + Design
Dance
Special Arts Services
Individual Artists
Arts In Education
Visual Arts
Architecture Planning & Design
Presenting
Music
Folk Arts
Stimulus
Electronic Media & Film
Stabilization
Arts Education
$50,491
$50,156
$40,320
$36,140
$35,909
$34,992
$31,424
$26,009
$21,522
$20,701
$19,613
$18,738
$17,513
$17,481
$15,478
$14,696
$14,462
$14,366
$14,282
$13,469
$13,336
$13,290
0K
5K
10K
15K
20K
25K
30K
35K
40K
45K
50K
55K
WINDOW_MAX([Avg Grant Amount])

Figure 3-11. A sneak peek into the Cleveland dot plot, prior to adjusting formatting.

Step 5: Adjust the circle size

Step 5a: Mark Type. Click on the right-most Avg Grant Amount on columns. Change the mark type from Automatic to Circle.

Step 5b: Size. Change the size of the circle. We prefer a smaller circle: that is, less than 50% of the distance between the lines that we created with WINDOW_MAX(). Sizing this mark must be done manually, so do your best to eye the size to something that seems appropriate.

Step 6: Finalize the visualization

Step 6a: Color. Adjust the color to something that stands out from the background lines.

Step 6b: Axes. Hide the axes.

Step 6c: Format. Remove all lines and dividers in the formatting pane.

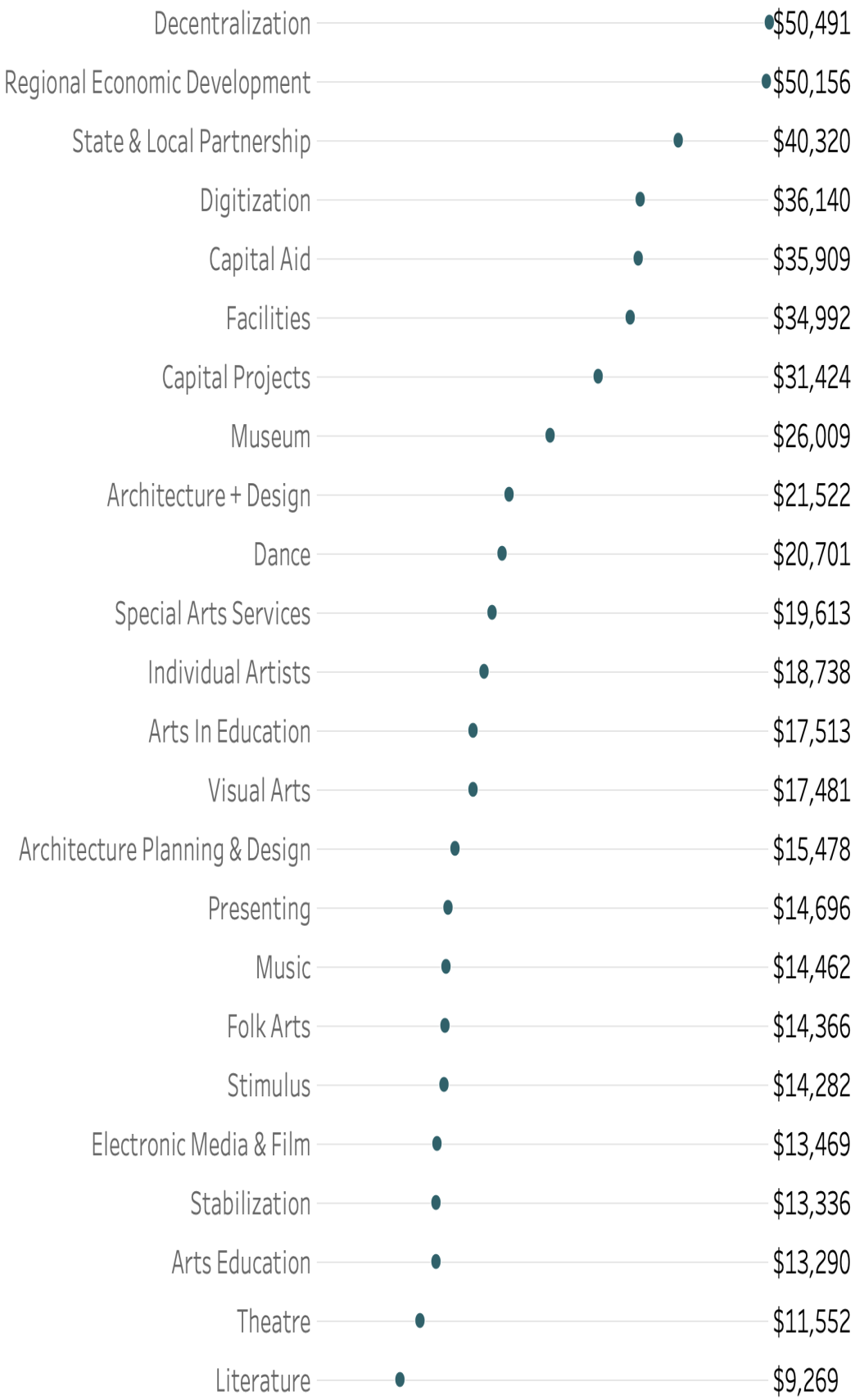

Decentralization $50,491
Regional Economic Development $50,156
State & Local Partnership $40,320
Digitization $36,140
Capital Aid $35,909
Facilities $34,992
Capital Projects $31,424
Museum $26,009
Architecture + Design $21,522
Dance $20,701
Special Arts Services $19,613
Individual Artists $18,738
Arts In Education $17,513
Visual Arts $17,481
Architecture Planning & Design $15,478
Presenting $14,696
Music $14,462
Folk Arts $14,366
Stimulus $14,282
Electronic Media & Film $13,469
Stabilization $13,336
Arts Education $13,290
Theatre $11,552
Literature $9,269

Figure 3-12. The final result: a Cleveland dot plot depicting the average grant amount by category.

Figure 3-11 shows the result.

What we like about the dot plot over the bar chart is that it provides us with the same information as a bar chart but increases the overall whitespace, maintains existing comparisons of the values, and aligns the text and the values in columns. We could get picky around right-aligning labels or increasing whitespace between the line and the labels, but for most audiences, the basic dot plot here will more than suffice.

Nonprofit Case Study: Showing Changes in Rank with a Bar Chart

Not all comparisons require you to examine magnitude across or within a measure. Sometimes your audience is merely interested in comparing how different groups change in rank over time.

There are many ways to show ranks of values. Here we will start with a very straightforward approach: showing the current rank in total grant dollars and the change in rank from one year to the next.

In this example we'll again display the total grant dollars for 2019. This time we'll display the rank relative to all other groups in 2019–but also the total change in rankings from 2018 to 2019 on the same metric (Figure 3-12).

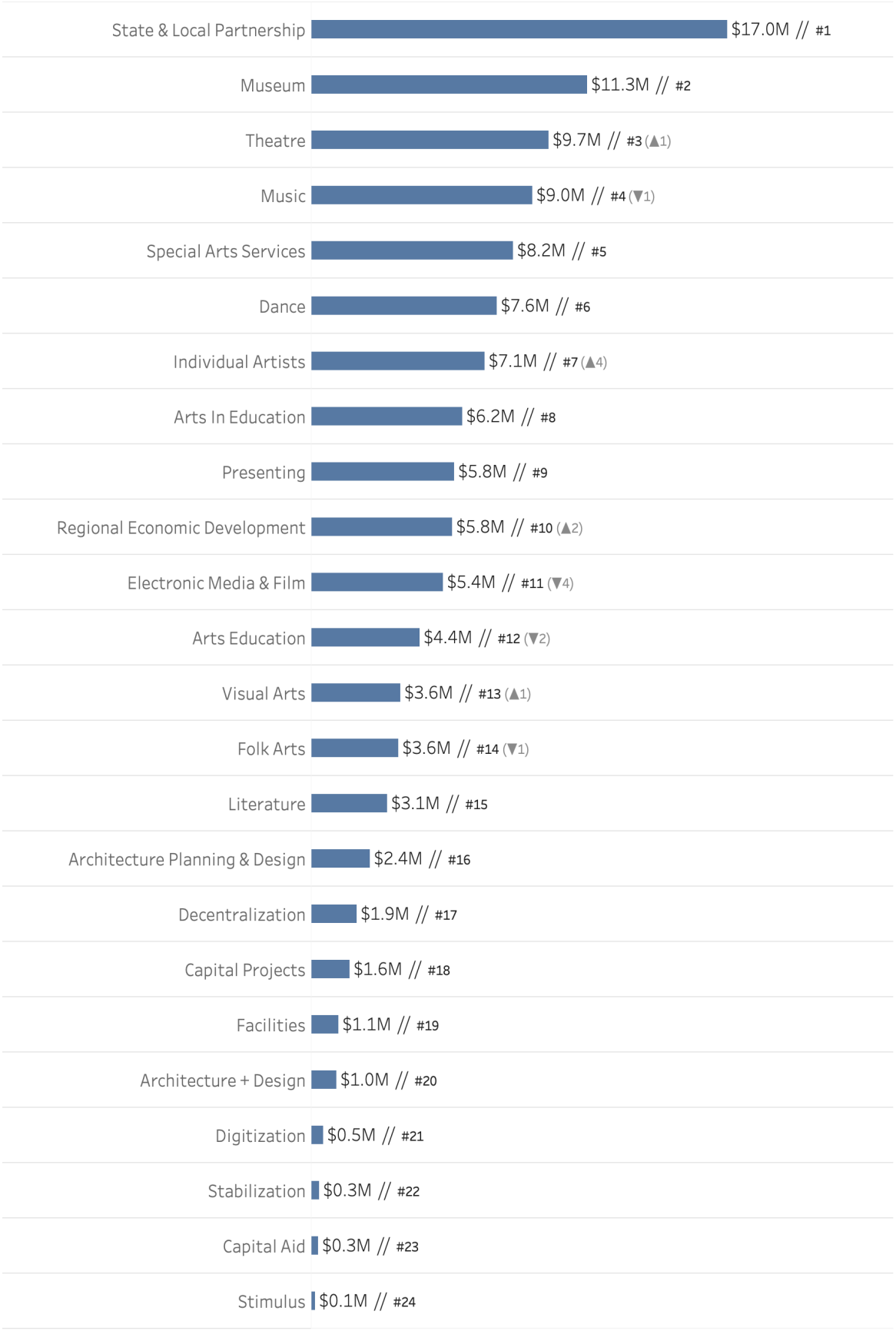

State & Local Partnership $17.0M // #1
Museum $11.3M // #2
Theatre $9.7M // #3 (▲1)
Music $9.0M // #4 (▼1)
Special Arts Services $8.2M // #5
Dance $7.6M // #6
Individual Artists $7.1M // #7 (▲4)
Arts In Education $6.2M // #8
Presenting $5.8M // #9
Regional Economic Development $5.8M // #10 (▲2)
Electronic Media & Film $5.4M // #11 (▼4)
Arts Education $4.4M // #12 (▼2)
Visual Arts $3.6M // #13 (▲1)
Folk Arts $3.6M // #14 (▼1)
Literature $3.1M // #15
Architecture Planning & Design $2.4M // #16
Decentralization $1.9M // #17
Capital Projects $1.6M // #18
Facilities $1.1M // #19
Architecture + Design $1.0M // #20
Digitization $0.5M // #21
Stabilization $0.3M // #22
Capital Aid $0.3M // #23
Stimulus $0.1M // #24

Figure 3-13. A snapshot of the visualization we will create for the next blueprint. Note the rank values–and the changes in rank, to the right side of the "//" dividers.

The reason for using rank in charts is simple: we–humans–can only process so much information. A change between two measurements is sometimes easier to understand.

Blueprint: Showing rank and change of rank on a bar chart

Step 1: Create the base bar chart.

Add Budget Category to rows and Grant Amount | 2019 to columns and to labels on the marks card (Figure 3-13).

Pages

Columns | AGG(Grant Amount | ..

Rows | Budget Category

Filters

Marks

Automatic

Color | Size | Label

Detail | Tooltip

AGG(Grant Am..

Change in Rank

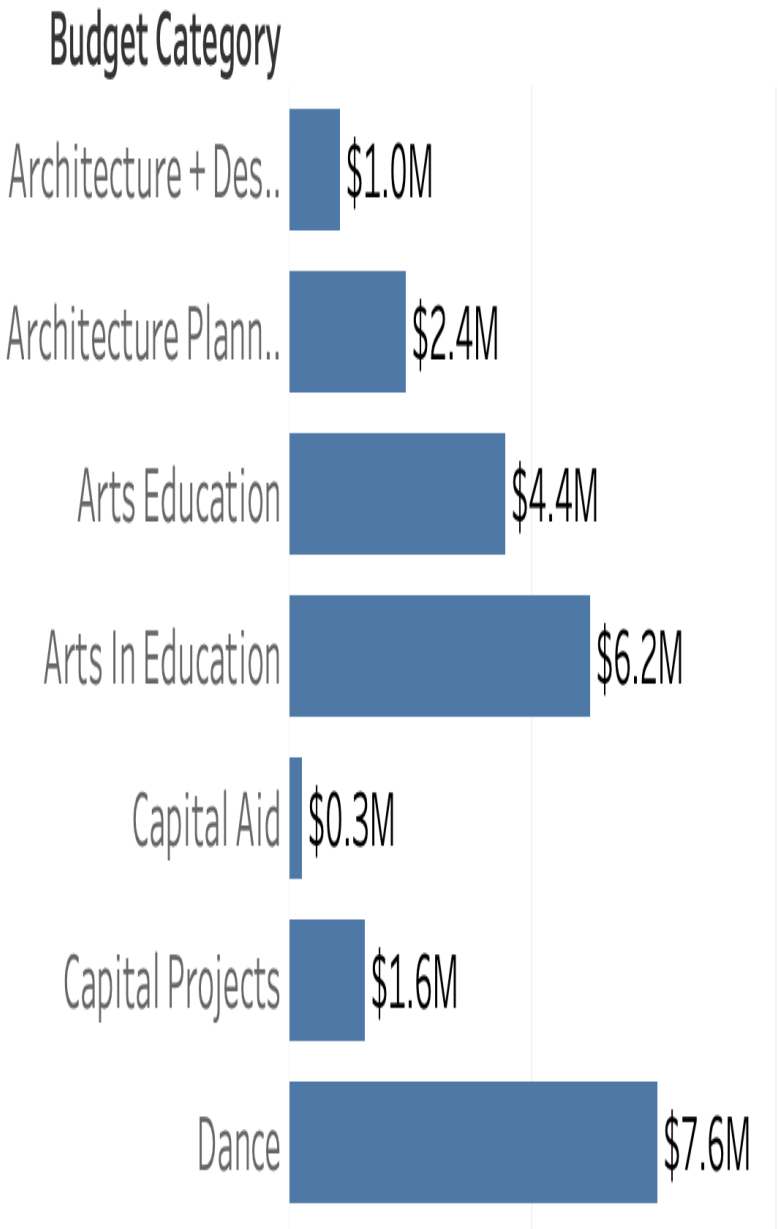


Figure 3-14. A horizontal bar chart depicting total grant dollars for 2019 by category.

Step 2: Create the rank calculation.

There are several different rank calculations built into Tableau including RANK(), RANK_DENSE(), RANK_MODIFIED(), RANK_PERCENTILE(), and RANK_UNIQUE(). While these calculations would make sense to use, we are going to use another method using the INDEX() function that tends to be more reliable.

Step 2a: Index. Create a new calculation called **Index | Grant Amount | 2019**:

// Index | Grant Amount | 2019

INDEX()

INDEX() creates a running count of values based on how you specify your table calculation. We'll set up how Index | Grant Amount | 2019 performs this running count by adjusting the table calculation on the view.

Step 2b: Marks card. Add Index | Grant Amount | 2019 to labels on the marks card (Figure 3-14).

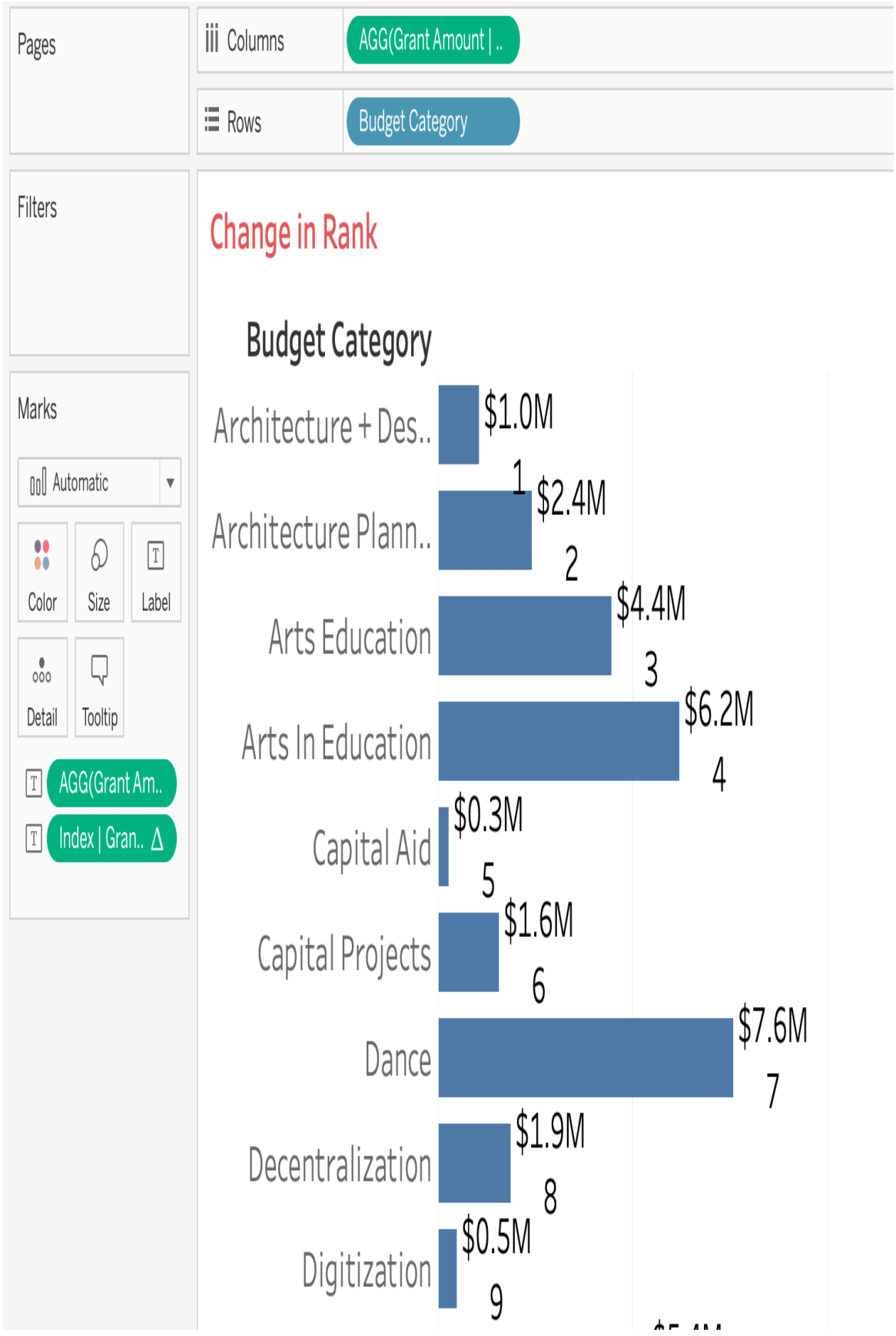

Pages
Columns
AGG(Grant Amount | ..
Rows
Budget Category
Filters
Change in Rank
Marks
Automatic
Color
Size
Label
Detail
Tooltip
AGG(Grant Am..
Index | Gran.. Δ
Budget Category
Architecture + Des..
$1.0M
1
Architecture Plann..
$2.4M
2
Arts Education
$4.4M
3
Arts In Education
$6.2M
4
Capital Aid
$0.3M
5
Capital Projects
$1.6M
6
Dance
$7.6M
7
Decentralization
$1.9M
8
Digitization
$0.5M
9

Figure 3-15. An unformatted horizontal bar chart showing total grant dollars in 2019 and an unsorted INDEX() value.

You'll notice that the function just adds a running value from top to bottom by mark on the view.

Step 2c: Table calculation. To make the calculation work like a rank, edit the table calculation by right clicking on the value on the marks card and editing the Table Calculation. Select Specific Dimensions and make sure Budget Category is selected. Now you can edit the sort order. Choose custom and sort descending on Grant Amount | 2019. The function should now look like a unique rank! (See Figure 3-15).

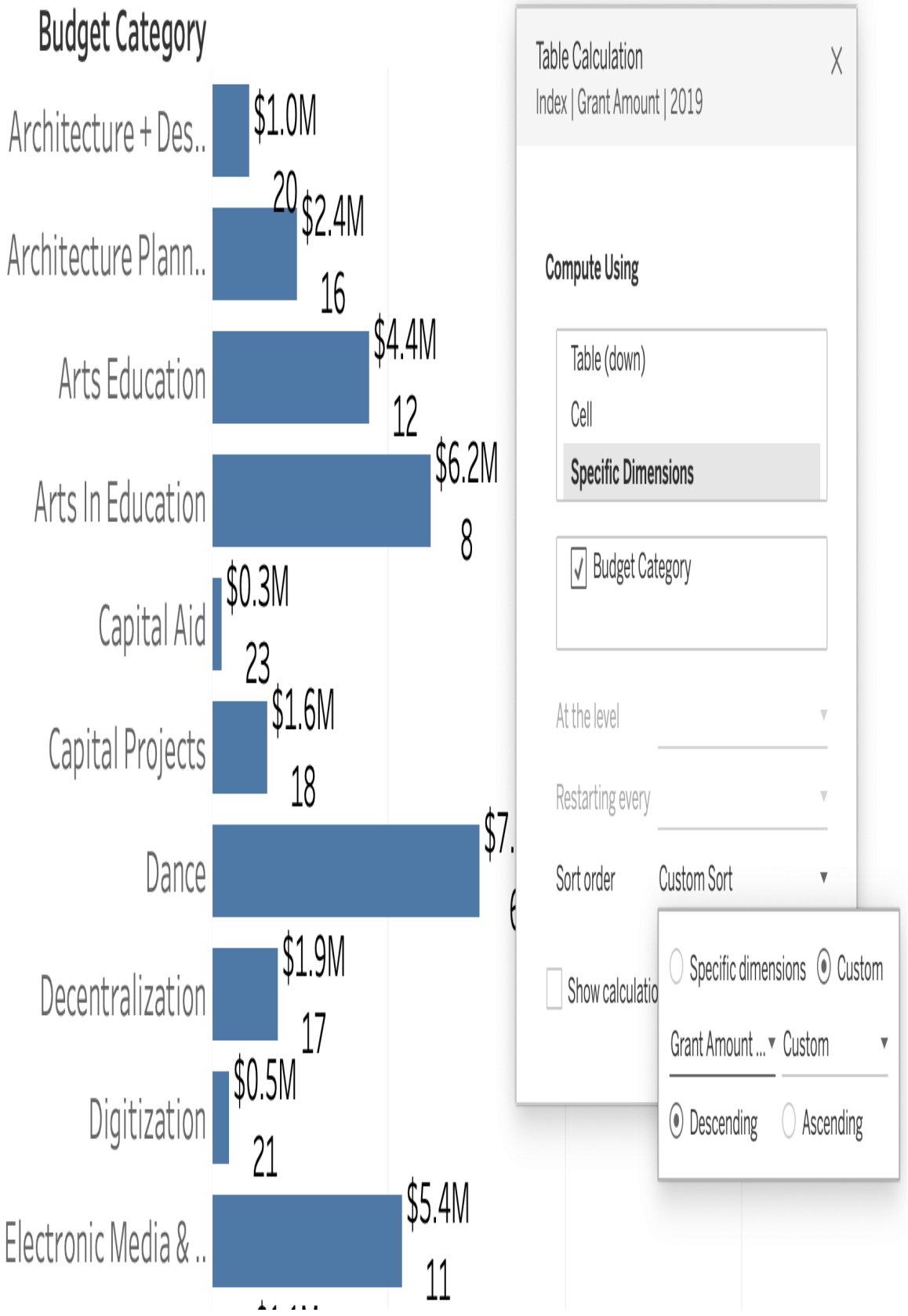

Budget Category
Architecture + Des..
$1.0M
20
Architecture Plann..
$2.4M
16
Arts Education
$4.4M
12
Arts In Education
$6.2M
8
Capital Aid
$0.3M
23
Capital Projects
$1.6M
18
Dance
Decentralization
$1.9M
17
Digitization
$0.5M
21
Electronic Media & ..
$5.4M
11
Table Calculation
Index | Grant Amount | 2019
Compute Using
Table (down)
Cell
Specific Dimensions
Budget Category
At the level
Restarting every
Sort order
Custom Sort
Show calculatio
Specific dimensions
Custom
Grant Amount ...
Custom
Descending
Ascending

Figure 3-16. An unformatted horizontal bar chart showing total grant dollars in 2019 and a sorted INDEX() value based on the total grant amounts in 2019.

Step 3: Create the difference-in-rank calculation

Step 3a: Index. Create a duplicate of the Index | Grant Amount | 2019 and call the new calculation Index | Grant Amount | 2018.

// Index | Grant Amount | 2018

INDEX()

Step 3b: Delta. For your next calculation you are going to take the difference between the two index calculations. Create a calculation called Index Change | Grant Amount where:

// Index | Grant Amount | 2018

[Index | Grant Amount | 2019] - [Index | Grant Amount | 2018]

Step 3c: Marks card and table calculation. Add this calculation to text on the marks card. Edit the table calculation. You will have the ability to customize both the Index | Grant Amount | 2018 and Index | Grant Amount | 2019.

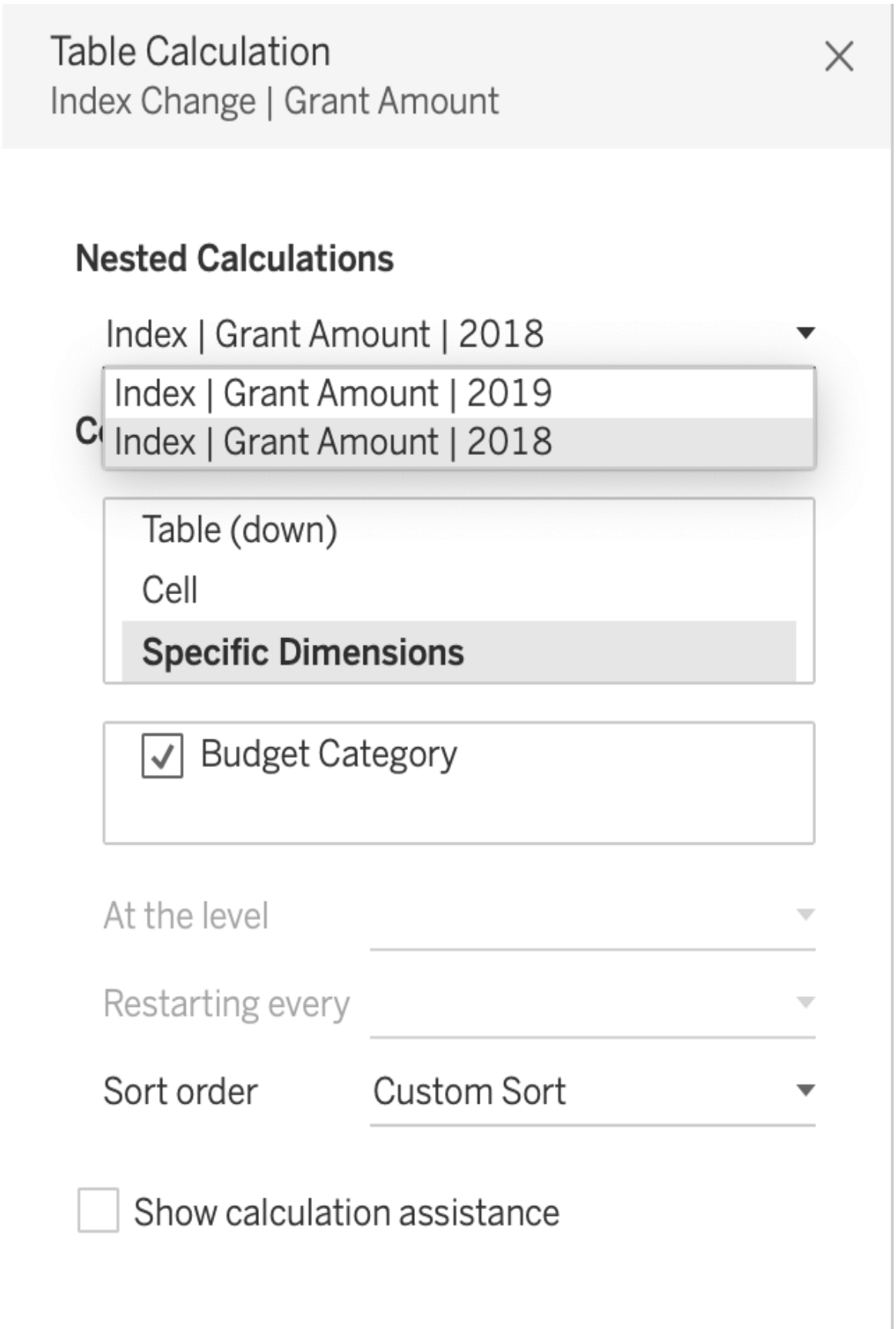


Table Calculation
Index Change | Grant Amount
Nested Calculations
Index | Grant Amount | 2018
Index | Grant Amount | 2019
Index | Grant Amount | 2018
Table (down)
Cell
Specific Dimensions
Budget Category
At the level
Restarting every
Sort order
Custom Sort
Show calculation assistance

Figure 3-17. Be sure to edit the table calculations for both 2018 and 2019. Use the dropdown to select the table calculation you would like to update.

This is an important step because the next calculation relies on a correct sort for both Index | Grant Amount | 2019 and Index | Grant Amount | 2018. (See Figure 3-16.)

Step 3d: Table calculation sorting. Be sure to do a custom sort descending for Index | Grant Amount | 2019 using Grant Amount | 2019 and a custom sort descending for Index | Grant Amount | 2019 using Grant Amount | 2019. This will give you a proper calculation showing the change in rank from 2018 to 2019.

Step 4: Add custom formatting

Once the calculation is displaying results properly, edit the text formatting of **Index Change | Grant Amount**. Use custom formatting and write (↑0);(↓0);"".

This custom formatting formats values one of three ways: when values are positive, when values are negative, or when values are zero. These are specified in an order of positive, negative, zero and are separated by the semicolon. Based on these rules the positive values will be in parentheses with an up arrow preceding the value, negative values will be in parentheses with a down arrow preceding the value, and if there is no change in rank nothing will show up because we are using empty quotations.

Once you've formatted the display of your labels, the visualization will look something like Figure 3-17.

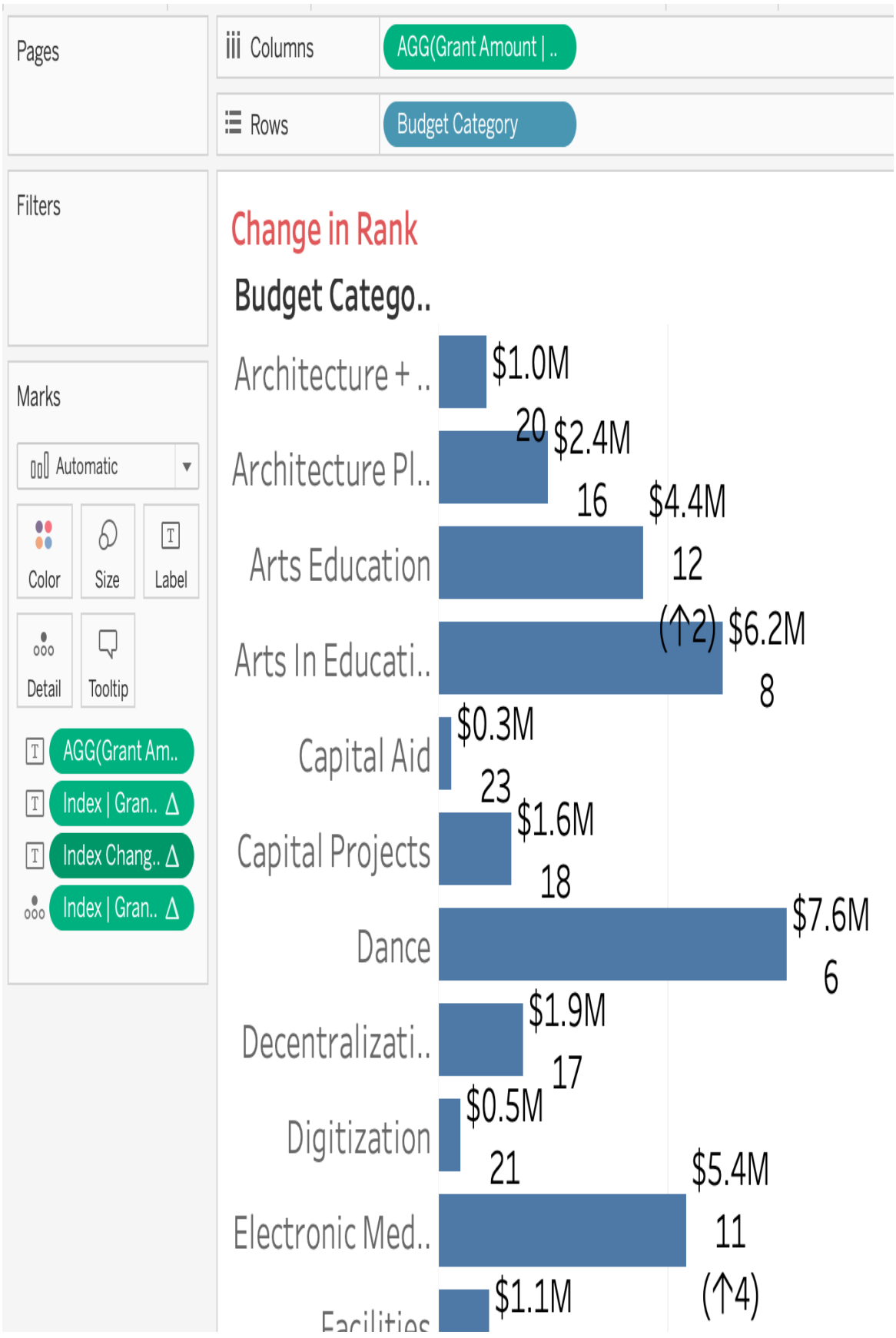

Pages
Columns
AGG(Grant Amount | ..
Rows
Budget Category
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
AGG(Grant Am..
Index | Gran.. Δ
Index Chang.. Δ
Index | Gran.. Δ
Change in Rank
Budget Catego..
Architecture + ..
$1.0M
Architecture Pl..
20
$2.4M
Arts Education
16
$4.4M
12
(↑2)
Arts In Educati..
$6.2M
8
Capital Aid
$0.3M
23
Capital Projects
$1.6M
18
Dance
$7.6M
6
Decentralizati..
$1.9M
17
Digitization
$0.5M
21
Electronic Med..
$5.4M
11
(↑4)
Facilities
$1.1M

Figure 3-18. An unformatted horizontal bar chart showing total grant dollars in 2019, a rank of total grant dollars in 2019 using an unsorted INDEX() value, and a change in rank using nested INDEX() functions in a single calculation.

Step 5: Finalize the visualization

Now that you've got all the components on the dashboard you need to finalize the view by formatting how the text is displayed on the visualization, formatting lines and dividers, and sorting the categories.

Start with editing the text label. For this blueprint we're going to place all the measures on a single line and add two slashes to help separate the ranks from the values (Figure 3-18).


Architecture + ..
$1.0M // 20
Architecture Pl..
$2.4M // 16
Arts Education
$4.4M // 12(↑2)
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Edit Label
Tableau Book
9
Insert
<AGG(Grant Amount | 2019)> // #<AGG(Index | Grant Amount | 2019)> <AGG(Index Change | Grant Amount)>
Reset
Preview
Apply
Cancel
OK
Folk Arts
$3.6M // 14(↑1)

Figure 3-19. A view of the text editor for the labels of the change-in-rank bar charts.

What’s great about this blueprint is the display of the change in rank. As you can see in Figure 3-19, the element allows us to see how change affects the order of categories from one period to the next.

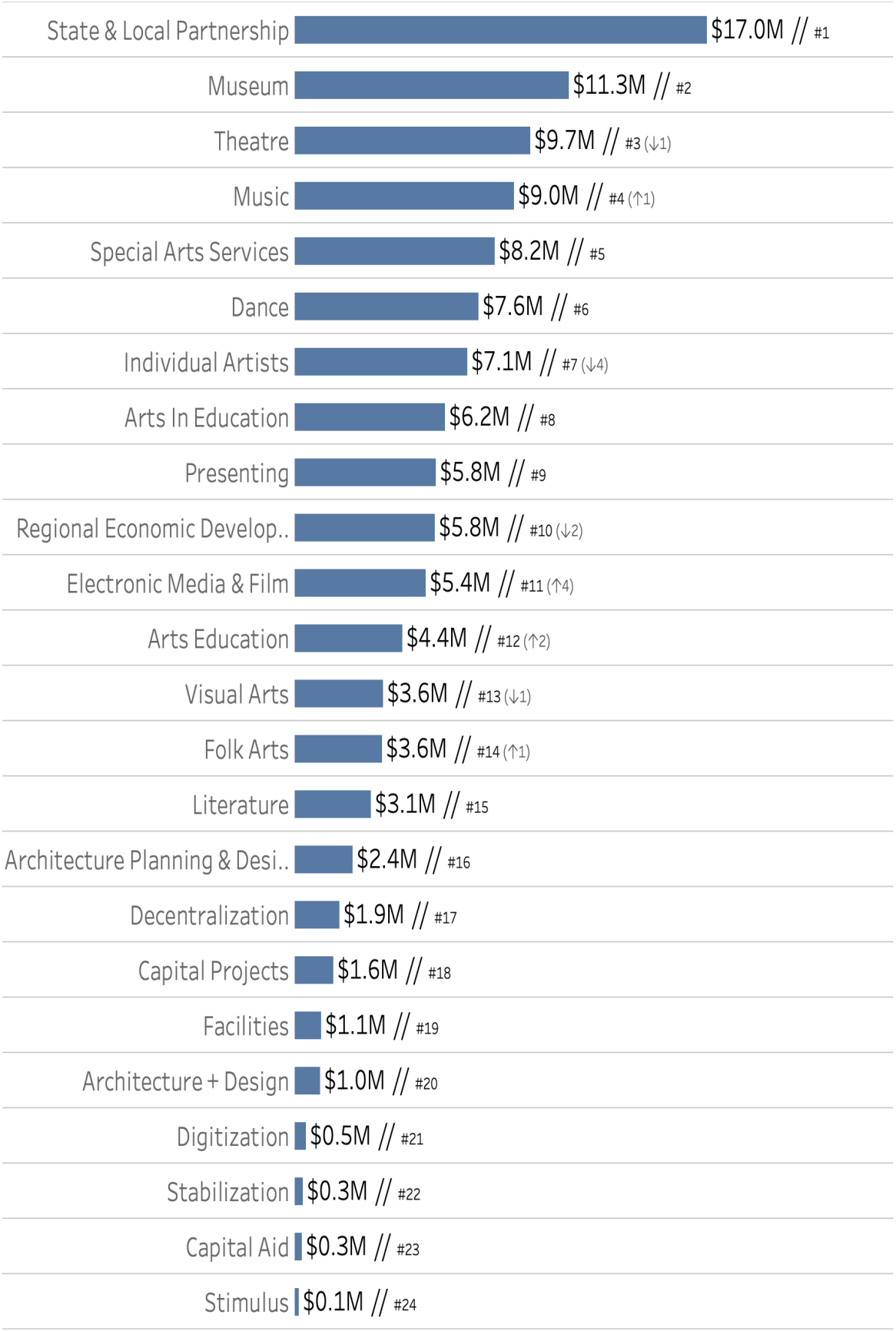

State & Local Partnership $17.0M // #1
Museum $11.3M // #2
Theatre $9.7M // #3 (↓1)
Music $9.0M // #4 (↑1)
Special Arts Services $8.2M // #5
Dance $7.6M // #6
Individual Artists $7.1M // #7 (↓4)
Arts In Education $6.2M // #8
Presenting $5.8M // #9
Regional Economic Develop.. $5.8M // #10 (↓2)
Electronic Media & Film $5.4M // #11 (↑4)
Arts Education $4.4M // #12 (↑2)
Visual Arts $3.6M // #13 (↓1)
Folk Arts $3.6M // #14 (↑1)
Literature $3.1M // #15
Architecture Planning & Desi.. $2.4M // #16
Decentralization $1.9M // #17
Capital Projects $1.6M // #18
Facilities $1.1M // #19
Architecture + Design $1.0M // #20
Digitization $0.5M // #21
Stabilization $0.3M // #22
Capital Aid $0.3M // #23
Stimulus $0.1M // #24

Figure 3-20. The final version showing total grant dollars for 2019, the rank compared across 24 categories and the change in rank from 2018 to 2019.

Bump Charts

The previous blueprint focused on the change in rank over just two periods. What if you're trying to track this period over multiple periods? Certainly text with arrows for several bar charts won't work. You'll need to use a different chart type.

Nonprofit Case Study: Tracking Changes in Rank over Multiple Periods

First, let's look at a Figure 3-20, a line chart showing total grant dollars by quarter, and consider it from the perspective of our boss and her audience:

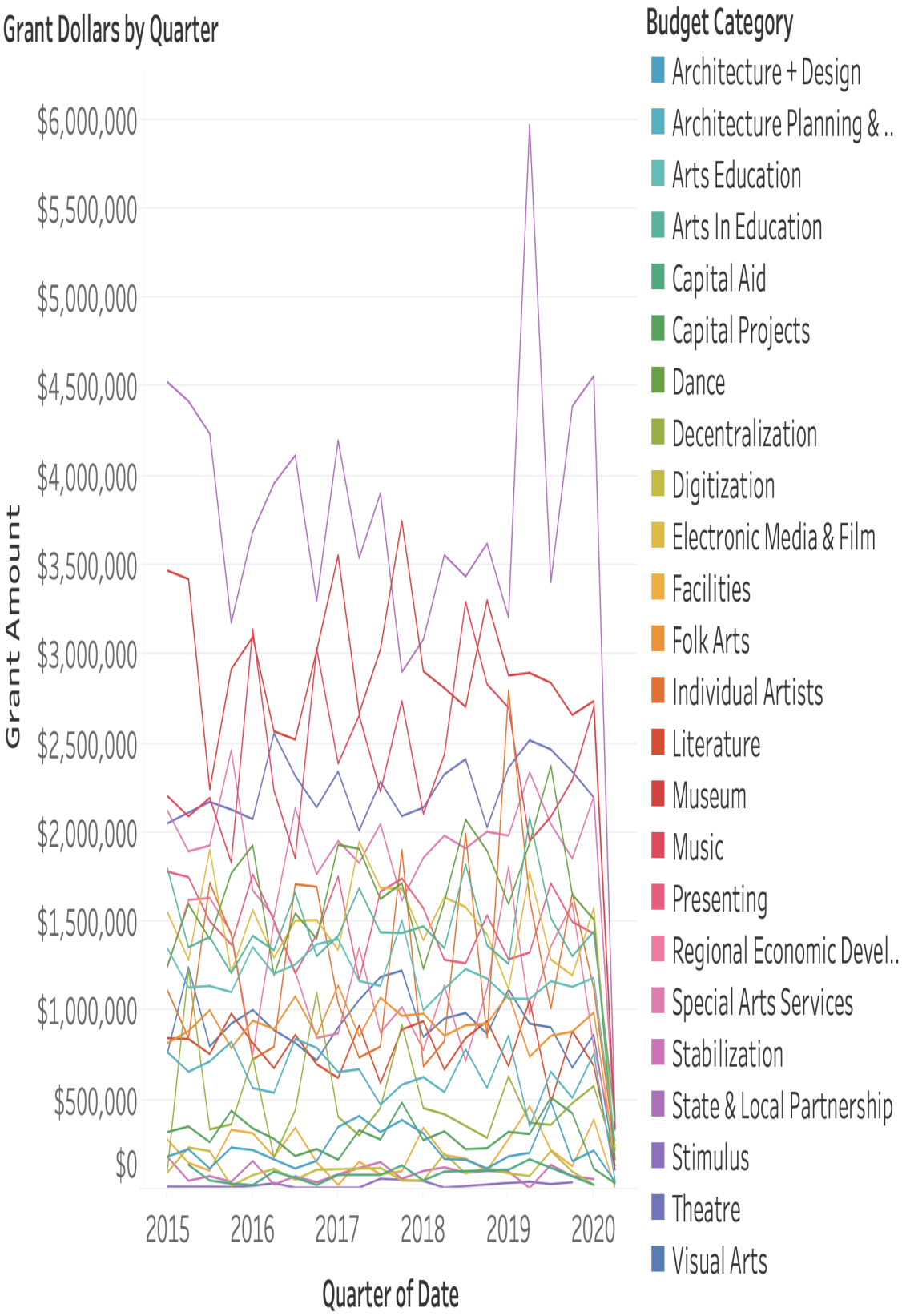

Grant Dollars by Quarter
Grant Amount
$6,000,000
$5,500,000
$5,000,000
$4,500,000
$4,000,000
$3,500,000
$3,000,000
$2,500,000
$2,000,000
$1,500,000
$1,000,000
$500,000
$0
2015
2016
2017
2018
2019
2020
Quarter of Date
Budget Category
Architecture + Design
Architecture Planning & ..
Arts Education
Arts In Education
Capital Aid
Capital Projects
Dance
Decentralization
Digitization
Electronic Media & Film
Facilities
Folk Arts
Individual Artists
Literature
Museum
Music
Presenting
Regional Economic Devel..
Special Arts Services
Stabilization
State & Local Partnership
Stimulus
Theatre
Visual Arts

Figure 3-21. A (less than ideal) line chart showing total grant dollars by quarter for each category.

There are a lot of lines here, but they are surprisingly not too difficult to track. Normally you should avoid more than eight lines on a line chart because too many tend to overlap. But when our audience follows any individual line, it's easier to see where increases or decreases are happening for a single line because there is decent spacing between categories. But our boss's audience is more interested in how any category compares to others, and this plot doesn't show that change over time. What's needed is a plot that describes this change in terms of rank.

One of the best chart types for change in rank over time is a bump chart. Using a bump chart allows you to see how changes in rank occur over time without getting completely lost in a chart compared to other members. Instead of building the above chart, let's build a bump chart that allows us to select a single category, as shown in Figure 3-21.

Ranking of Categories

Total Grant Dollars Per Biannual

Individual Artists

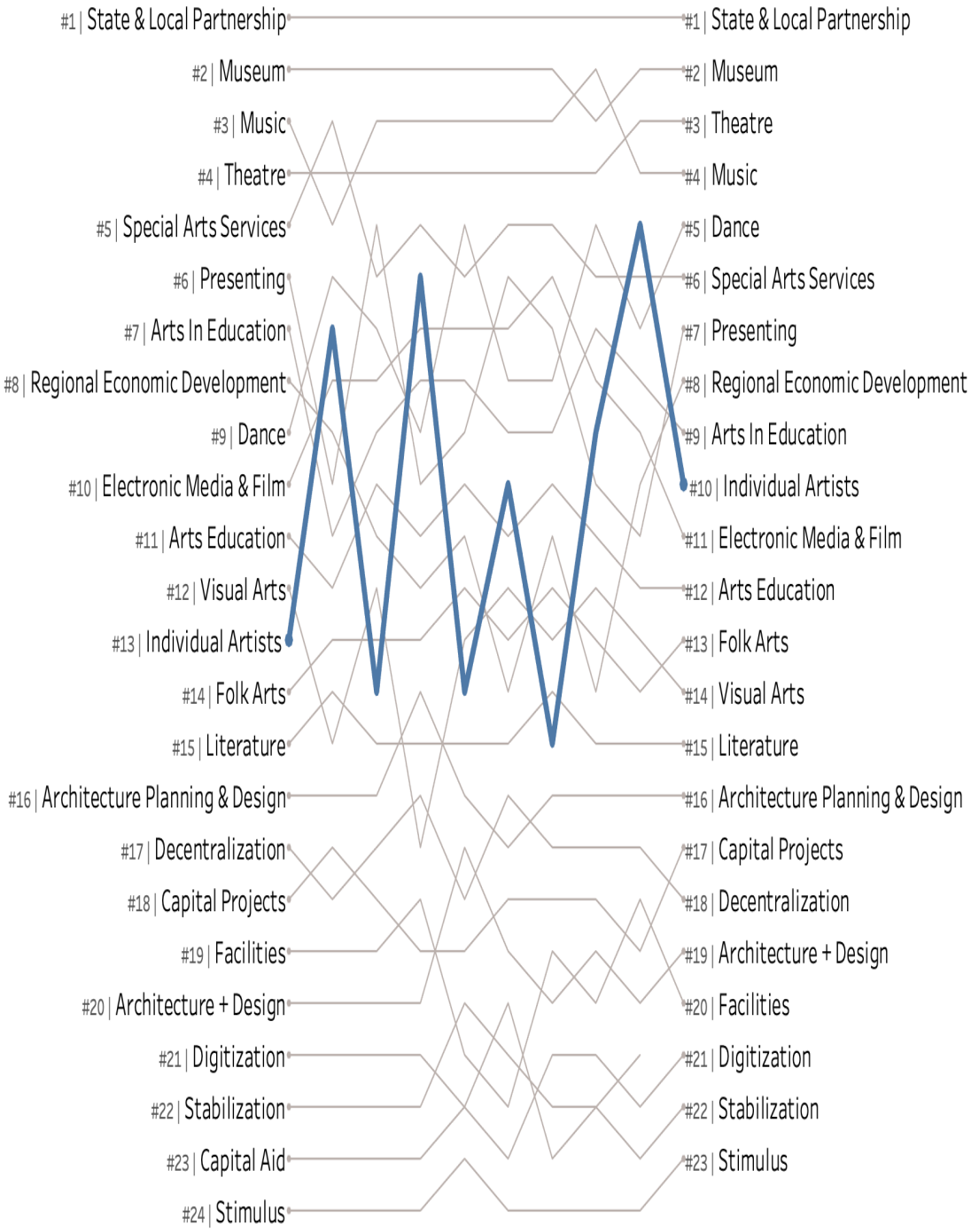


Figure 3-22. A bump chart showing the change in rank over time by quarter of the 24 different categories.

A bump chart is fundamentally different than a line chart. The next blueprint shows how the ranks of categories change biannually from 2015 through 2019. The information does not convey how close the underlying grant totals are, only the overall rank. This bump chart shows the cyclical nature of grant for individual artists relative to other categories.

Blueprint: Making a Bump Chart

Step 1: Create a line chart.

Step 1a: Marks Card Detail. Add SUM(Grant Amount) to rows and Budget Category to detail on the marks card.

Step 1b: Columns Shelf. Create a new calculated field called Date | Biannual that will return date dates to the nearest biannual–either January 1 or July 1.

```
// Date | Biannual

IF MONTH([Date]) > 6

THEN DATEADD('month', 6, DATETRUNC('year', [Date]))

ELSE DATETRUNC('year', [Date])

END
```

Place this calculation on columns as an exact date.

Step 1c: Filtering. Add a context filter using Date | Biannual and select all years but 2020.

Step 1d. Rows Shelf. Add a rank table calculation to SUM(Grant Amount) on rows. Set the rank to be unique and descending (Figure 3-22).

Compute the calculation on Budget Category.

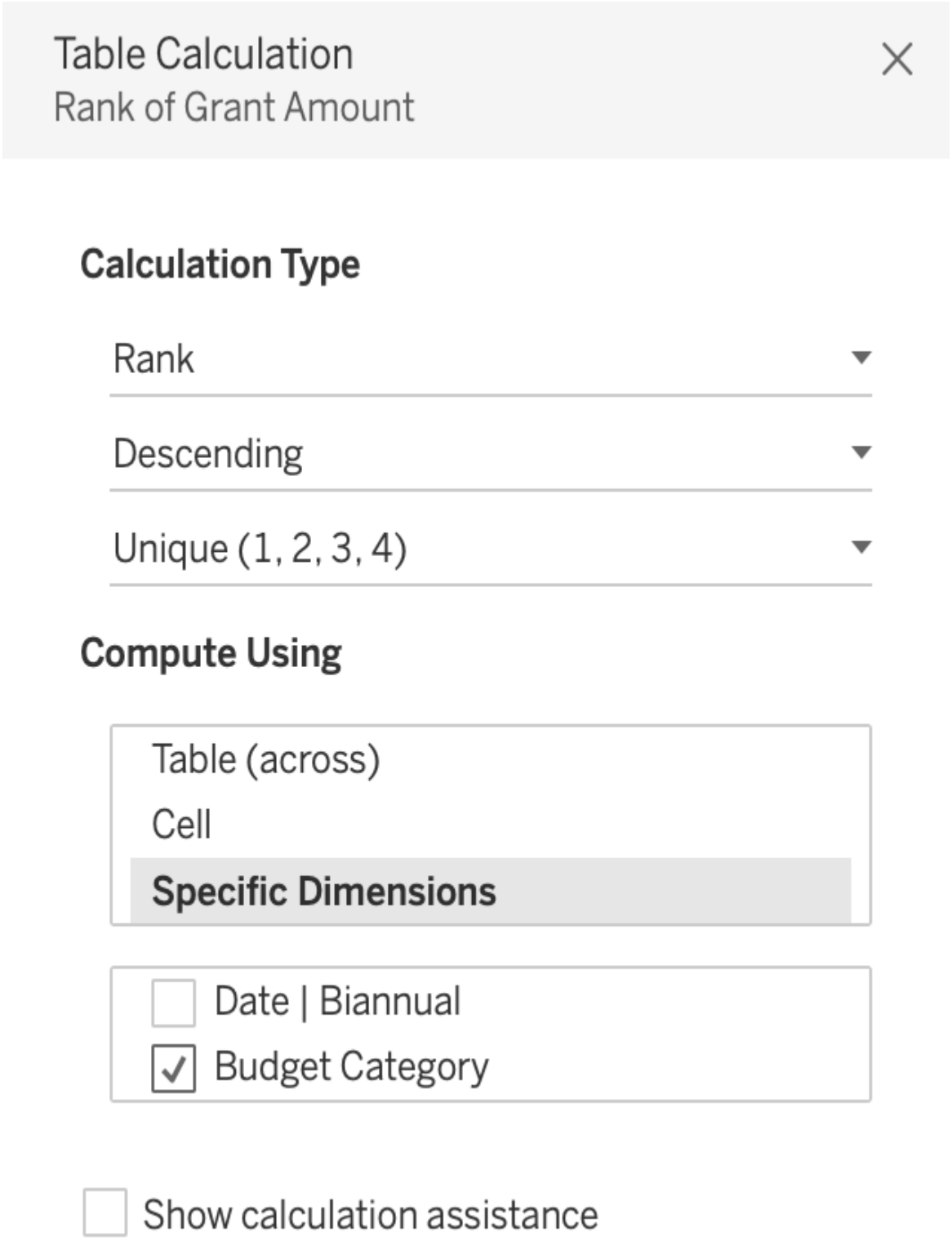


Figure 3-23. Creating a unique descending rank of total grant dollars for every date by budget category.

Drag the measure to the Tables area of the Data Pane. This will create a new saved calculation. Edit the name of the calculation to Grant Amount | Rank.

Once you’ve created the calculation, your visualization will look like Figure 3-23.

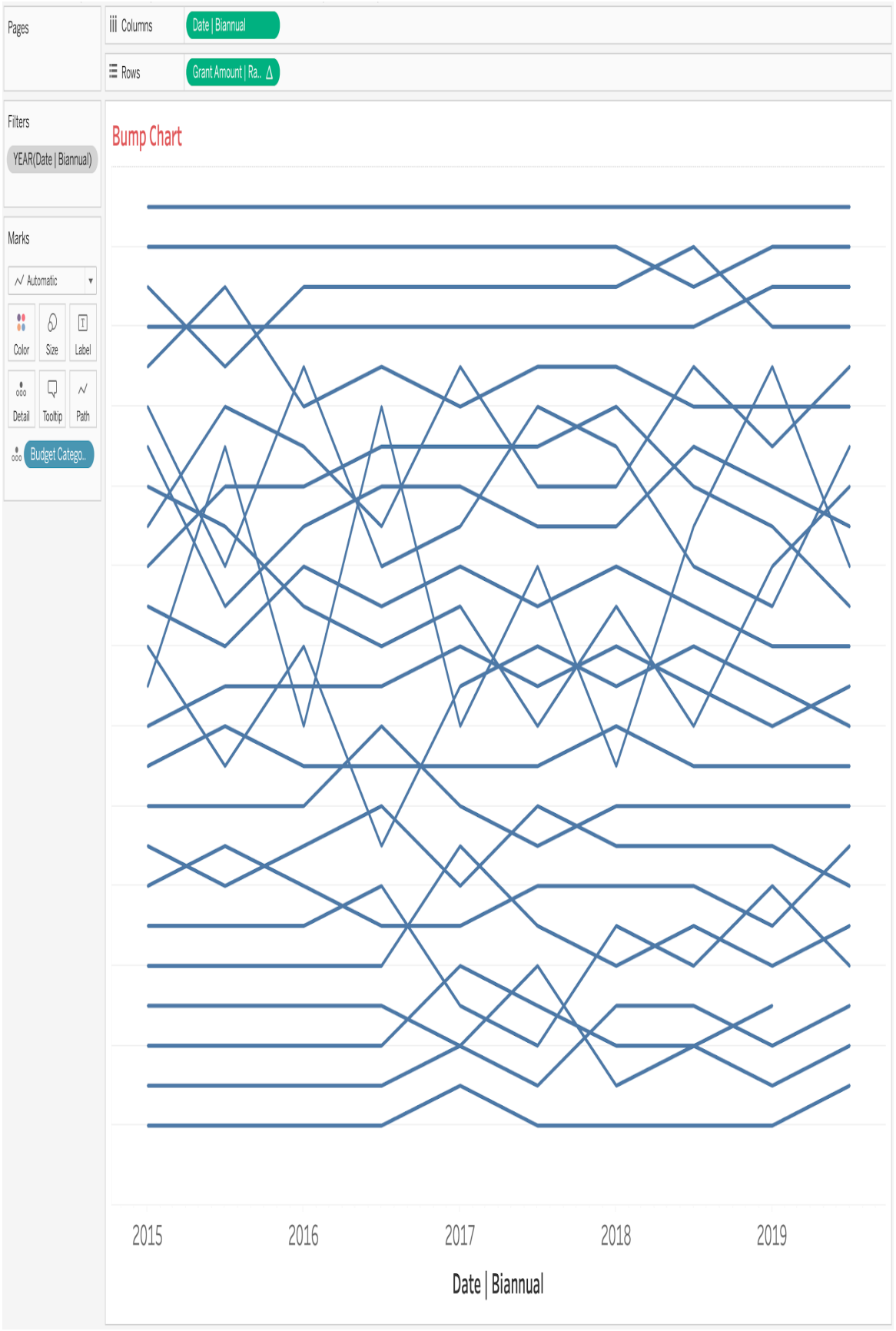

Pages
Columns
Date | Biannual
Rows
Grant Amount | Ra..
Filters
YEAR(Date | Biannual)
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Path
Budget Catego..
Bump Chart
2015
2016
2017
2018
2019
Date | Biannual

Figure 3-24. A look at the work-in-progress after adding the rank calculation.

Step 2: Organize and add labels

Step 2a: Prepare your worksheet. Start by editing the axis and reverse the axis. Hide the rank axis.

Step 2b: Calculations. To add the labels, create two calculations. The first should be called Bump Category Label:

```
// Bump Category Label
 IF [Date | Biannual] = {MIN([Date | Biannual])}
 OR [Date | Biannual] = {MAX([Date | Biannual])}
 THEN [Budget Category]
 END
```

Call the second Bump Rank Label.

```
// Bump Rank label
 IF MAX([Date | Biannual]) = MIN({MAX([Date | Biannual])})
 OR MIN([Date | Biannual]) = MAX({MIN([Date | Biannual])})
 THEN [Grant Amount | Rank]
 END
```

We will discuss parts of these calculations in Chapter 4. Just know for now that both calculations will show labels on the start and ending dates for each category and corresponding rank.

Step 2c: Build the labels. (See Figure 3-24.)

Add Bump Category Label as an attribute to labels.

Add Bump Rank Label and change the table calculation to specific dimensions on Budget Category.

Edit the axis of Date | Biannual to range from 1/1/2013 to 1/1/2022, then hide the axis.

Right-click and format Bump Rank Label so that there is a “#” in front of the rank values.

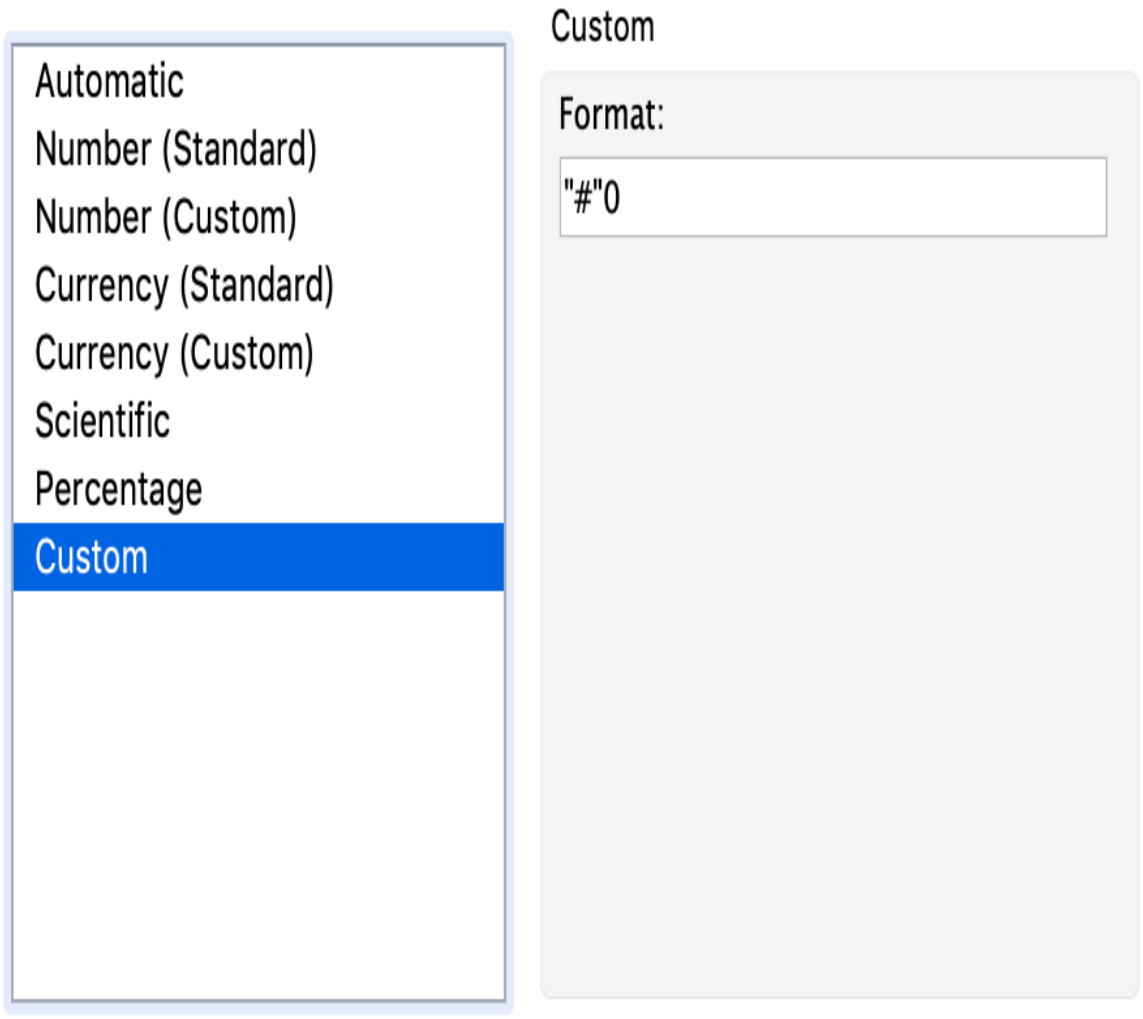


Figure 3-25. Custom formatting of the text labels for the bump chart.

After formatting the text, align the labels to the middle. It should look something like Figure 3-25.

Pages

Columns: Date | Biannual

Rows: Grant Amount | Ra.. Δ

Filters

YEAR(Date | Biannual)

Marks

Automatic

Color | Size | Label | Detail | Tooltip | Path

ATTR(Bump C..

Bump Rank.. Δ

Budget Catego..

Bump Chart

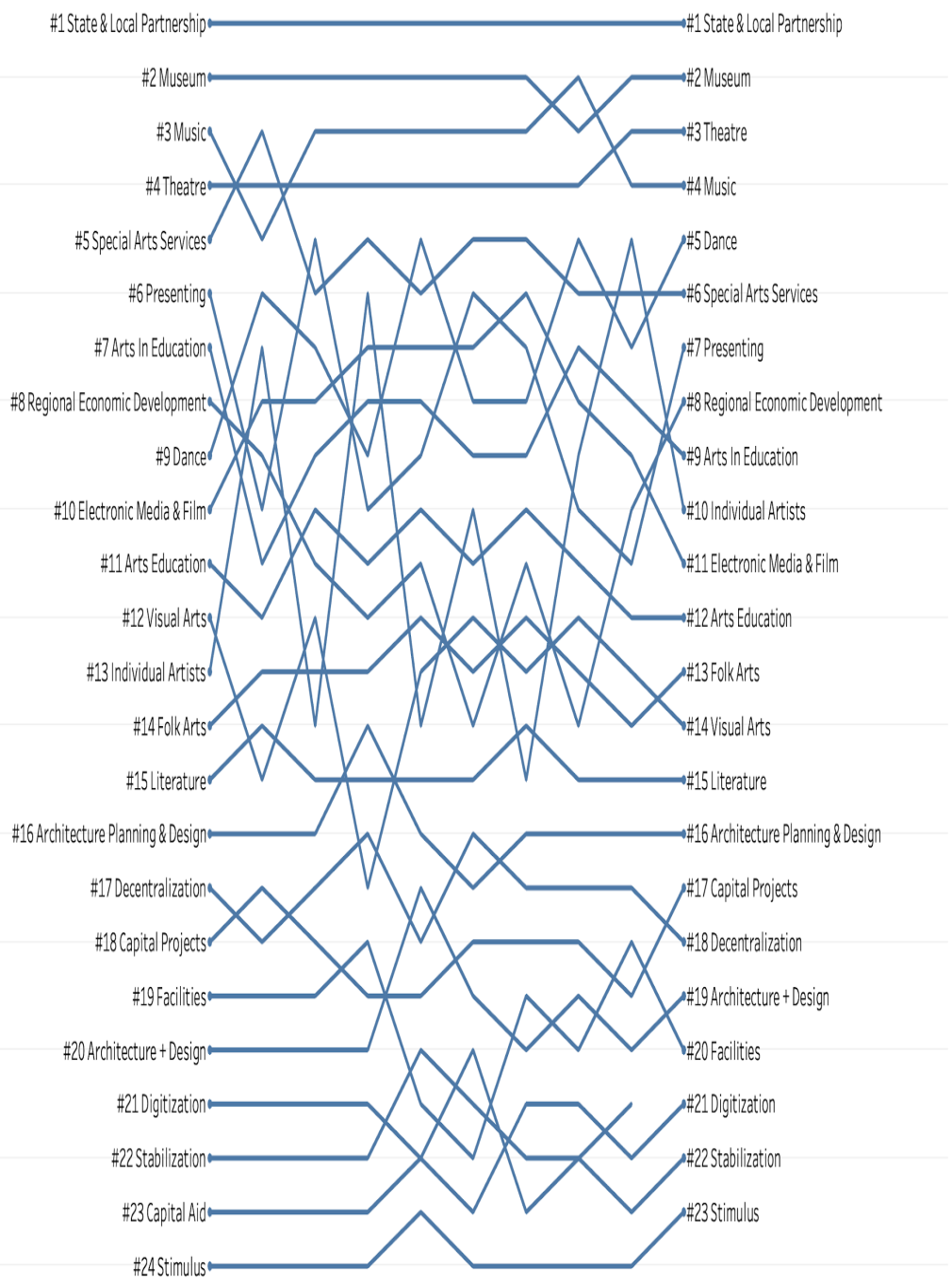


Figure 3-26. A look at the work-in-progress for the bump chart, prior to adding highlighting.

Step 3: Highlight a single category

Step 3a: Parameter. Create a parameter from Budget Category called Budget Category Parameter.

Step 3b: Calculation. Create a calculation from the dimension and parameter called Budget Category | TF:

// Budget Category | TF

[Budget Category] = [Budget Category Parameter]

Step 3c: Marks card. Place this new dimension on size and color.

Step 3d: Edit colors. Edit the colors so false values are gray and true values are blue. Place the true values in front of the false values and make the size of the true values larger than the false values.

Step 3e: table calculations. Edit the table calculation for both Grant Amount | Rank and Bump Rank Label to include both Budget Category and Budget Category | TF as the specific dimensions used to compute the table calculations (Figure 3-26).

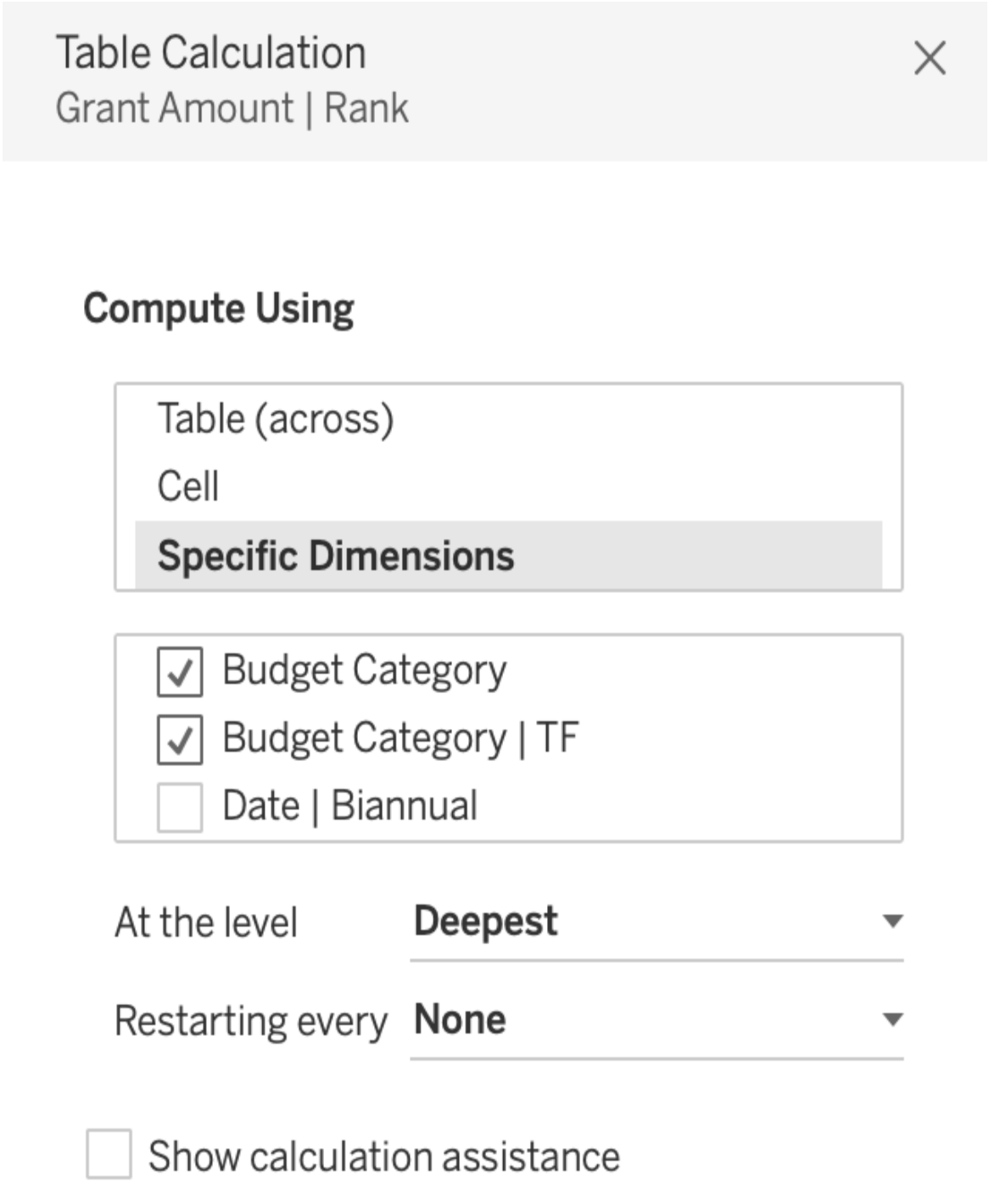


Figure 3-27. Adjust the table calculation of your rank calculation to include Budget Category | TF as a selected dimension.

After completing this step, your visualization will look like Figure 3-28.

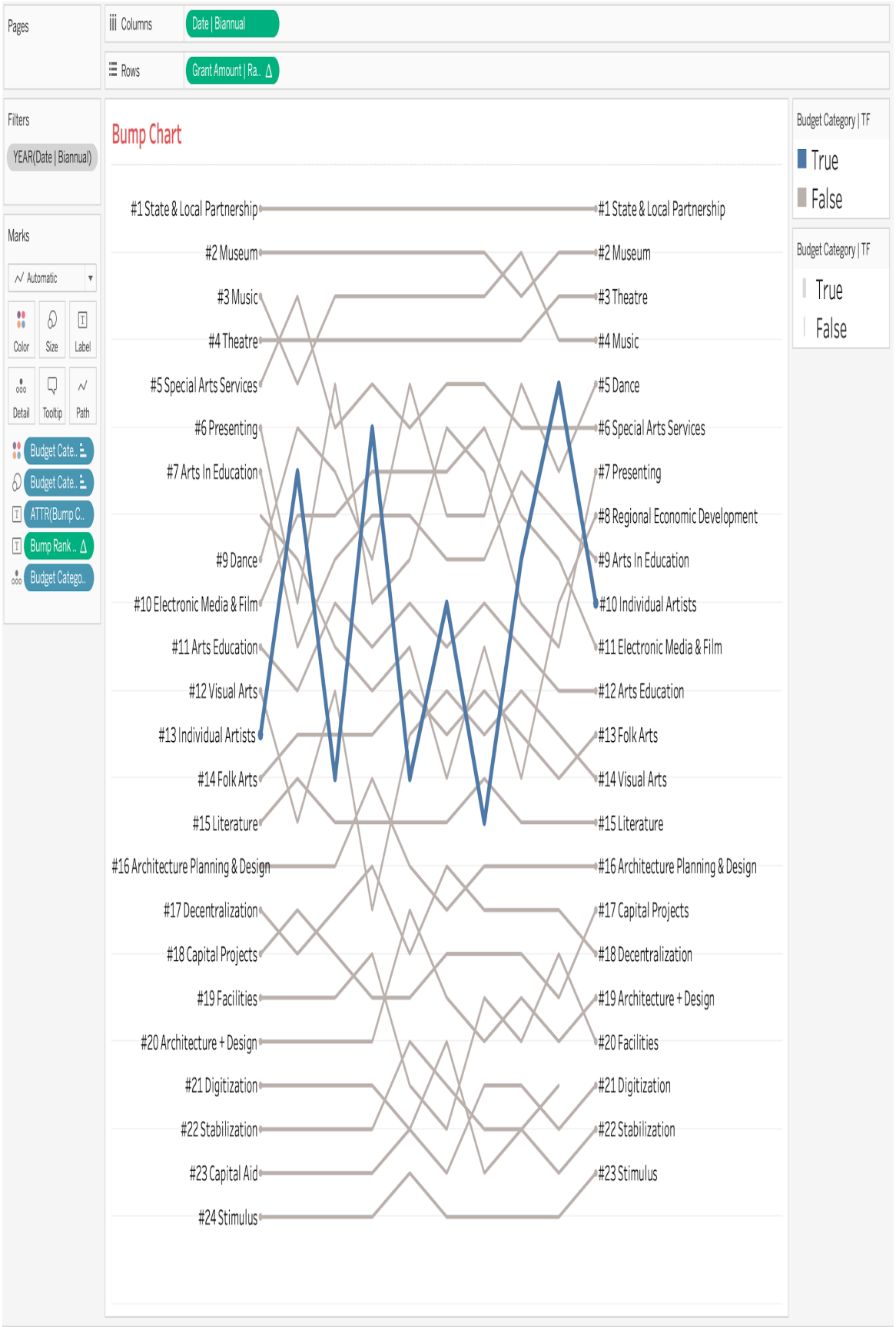

Pages
Columns
Date | Biannual
Rows
Grant Amount | Ra.. Δ
Filters
YEAR(Date | Biannual)
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Path
Budget Cate..
Budget Cate..
ATTR(Bump C..
Bump Rank .. Δ
Budget Catego..
Bump Chart
#1 State & Local Partnership
#2 Museum
#3 Music
#4 Theatre
#5 Special Arts Services
#6 Presenting
#7 Arts In Education
#9 Dance
#10 Electronic Media & Film
#11 Arts Education
#12 Visual Arts
#13 Individual Artists
#14 Folk Arts
#15 Literature
#16 Architecture Planning & Design
#17 Decentralization
#18 Capital Projects
#19 Facilities
#20 Architecture + Design
#21 Digitization
#22 Stabilization
#23 Capital Aid
#24 Stimulus
#1 State & Local Partnership
#2 Museum
#3 Theatre
#4 Music
#5 Dance
#6 Special Arts Services
#7 Presenting
#8 Regional Economic Development
#9 Arts In Education
#10 Individual Artists
#11 Electronic Media & Film
#12 Arts Education
#13 Folk Arts
#14 Visual Arts
#15 Literature
#16 Architecture Planning & Design
#17 Capital Projects
#18 Decentralization
#19 Architecture + Design
#20 Facilities
#21 Digitization
#22 Stabilization
#23 Stimulus
Budget Category | TF
True
False
Budget Category | TF
True
False

Figure 3-28. : A minimally formatted bump chart highlighting a single category.

Step 4: Format the bump chart

Remove all lines and dividers.

It’s also worth formatting your tooltips to include the SUM(Grant Amount). We prefer to go beyond the default formatting of tooltips. The result for this example is the bump chart in Figure 3-28.

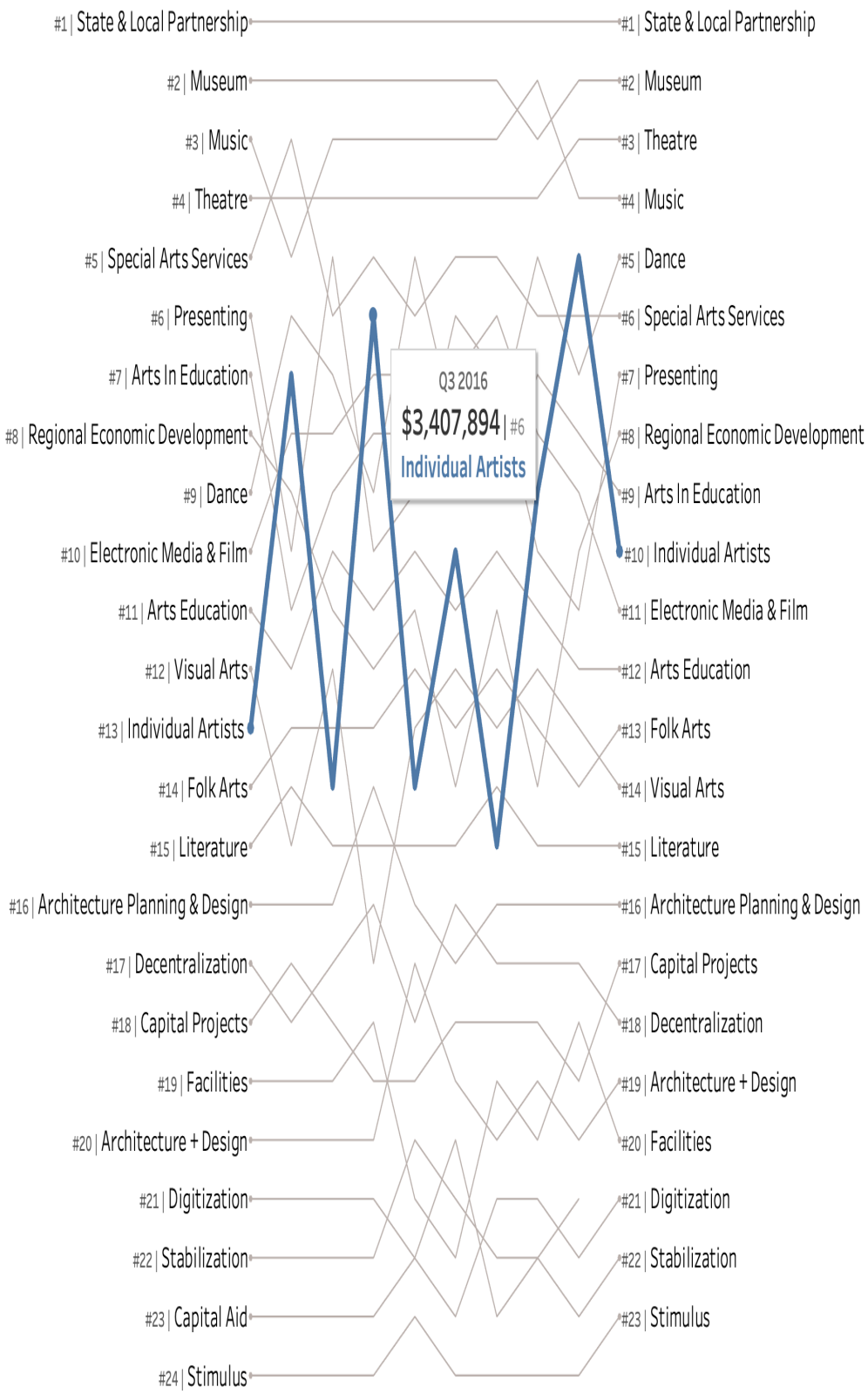

#1 | State & Local Partnership
#2 | Museum
#3 | Music
#4 | Theatre
#5 | Special Arts Services
#6 | Presenting
#7 | Arts In Education
#8 | Regional Economic Development
#9 | Dance
#10 | Electronic Media & Film
#11 | Arts Education
#12 | Visual Arts
#13 | Individual Artists
#14 | Folk Arts
#15 | Literature
#16 | Architecture Planning & Design
#17 | Decentralization
#18 | Capital Projects
#19 | Facilities
#20 | Architecture + Design
#21 | Digitization
#22 | Stabilization
#23 | Capital Aid
#24 | Stimulus
#1 | State & Local Partnership
#2 | Museum
#3 | Theatre
#4 | Music
#5 | Dance
#6 | Special Arts Services
#7 | Presenting
#8 | Regional Economic Development
#9 | Arts In Education
#10 | Individual Artists
#11 | Electronic Media & Film
#12 | Arts Education
#13 | Folk Arts
#14 | Visual Arts
#15 | Literature
#16 | Architecture Planning & Design
#17 | Capital Projects
#18 | Decentralization
#19 | Architecture + Design
#20 | Facilities
#21 | Digitization
#22 | Stabilization
#23 | Stimulus
Q3 2016
$3,407,894 | #6
Individual Artists

Figure 3-29. A fully formatted bump chart displaying the final visualization.

Barbell Charts

Comparisons go beyond bar charts and rankings. Sometimes you have to compare hierarchical data. One of the most common approaches for hierarchical comparisons is a barbell chart. (Surprise!)

The comparisons we will make for the rest of the chapter could be used in any industry. However, for this blueprint–and the remaining blueprints in the chapter–we're going to switch from community grant data to retail data.

Retail Case Study: Comparing Sales by Region

Remember our office-supply store from Chapter 2? We're going to go back there. This time we need to compare how well the company's main product subcategories sell in different regions. This retail data has 17 different subcategories of product lines and four sales regions. Our job is to examine the percent margin SUM(Profit)/SUM(Sales) within subcategory.

One way you might look at this data would be as a single column of bar charts, as in Figure 3-30.

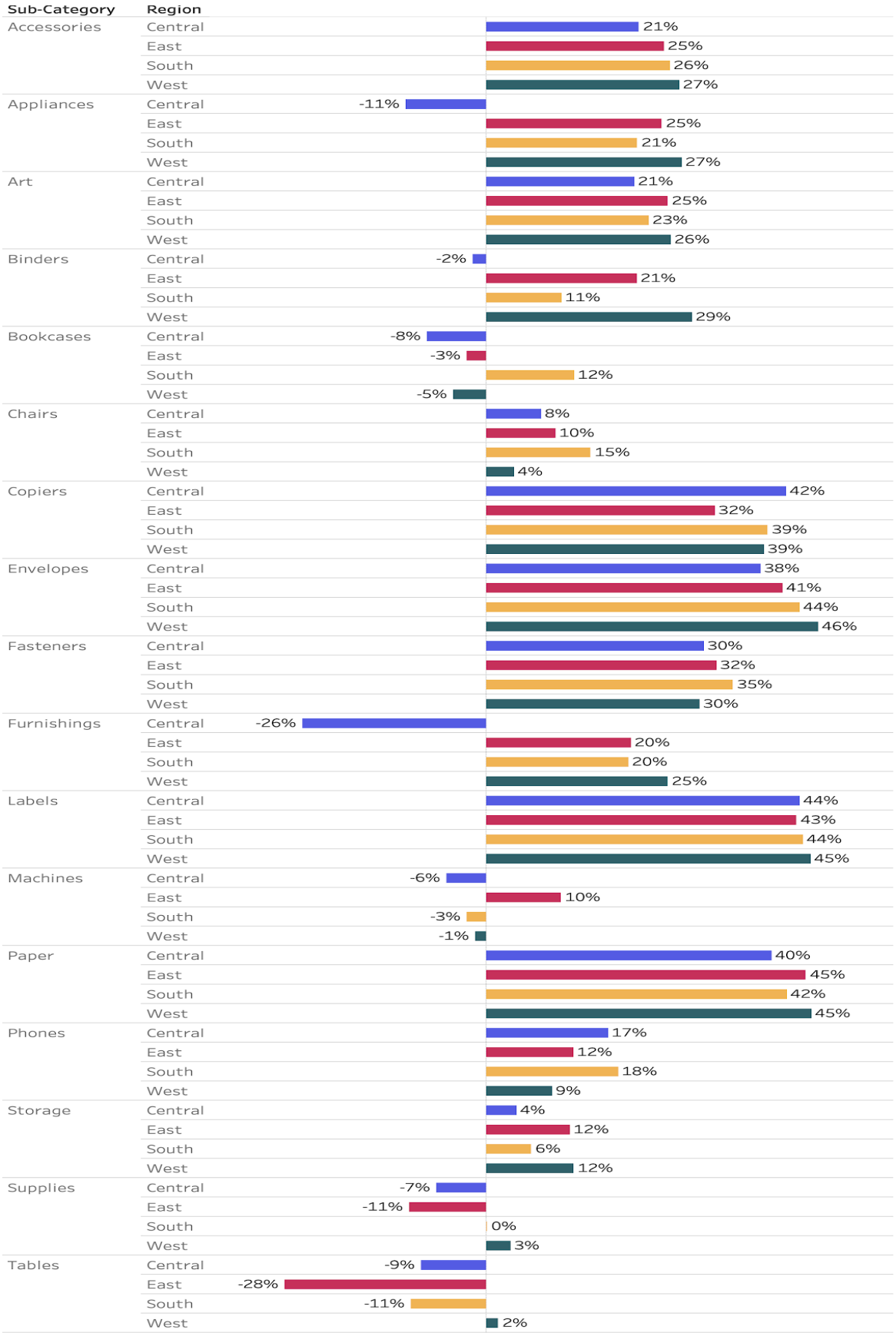

Sub-Category
Region
Accessories
Central 21%
East 25%
South 26%
West 27%
Appliances
Central -11%
East 25%
South 21%
West 27%
Art
Central 21%
East 25%
South 23%
West 26%
Binders
Central -2%
East 21%
South 11%
West 29%
Bookcases
Central -8%
East -3%
South 12%
West -5%
Chairs
Central 8%
East 10%
South 15%
West 4%
Copiers
Central 42%
East 32%
South 39%
West 39%
Envelopes
Central 38%
East 41%
South 44%
West 46%
Fasteners
Central 30%
East 32%
South 35%
West 30%
Furnishings
Central -26%
East 20%
South 20%
West 25%
Labels
Central 44%
East 43%
South 44%
West 45%
Machines
Central -6%
East 10%
South -3%
West -1%
Paper
Central 40%
East 45%
South 42%
West 45%
Phones
Central 17%
East 12%
South 18%
West 9%
Storage
Central 4%
East 12%
South 6%
West 12%
Supplies
Central -7%
East -11%
South 0%
West 3%
Tables
Central -9%
East -28%
South -11%
West 2%

Figure 3-30. : Side-by-side bar charts for subcategory and region. This chart is not ideal because of the space it consumes.

We can’t say this is a terrible solution: if our audience looks at any subcategory, they can easily compare regions. The problem is that if they are interested in the storage subcategory, they have to go through dozens of data points to find the information they are interested in. When they do get to it, they have to re-orient themselves to the chart.

An alternative option would be to move region to columns and percent of margin to rows, to create vertical bar charts within each subcategory. This decreases the cognitive load on the audience: they’ll be able to process the information more quickly because each row is now one subcategory (not a combination of subcategory and region). Now, when they scan from left to right, they’ll be able to quickly compare regions within a subcategory, as Figure 3-30 shows.

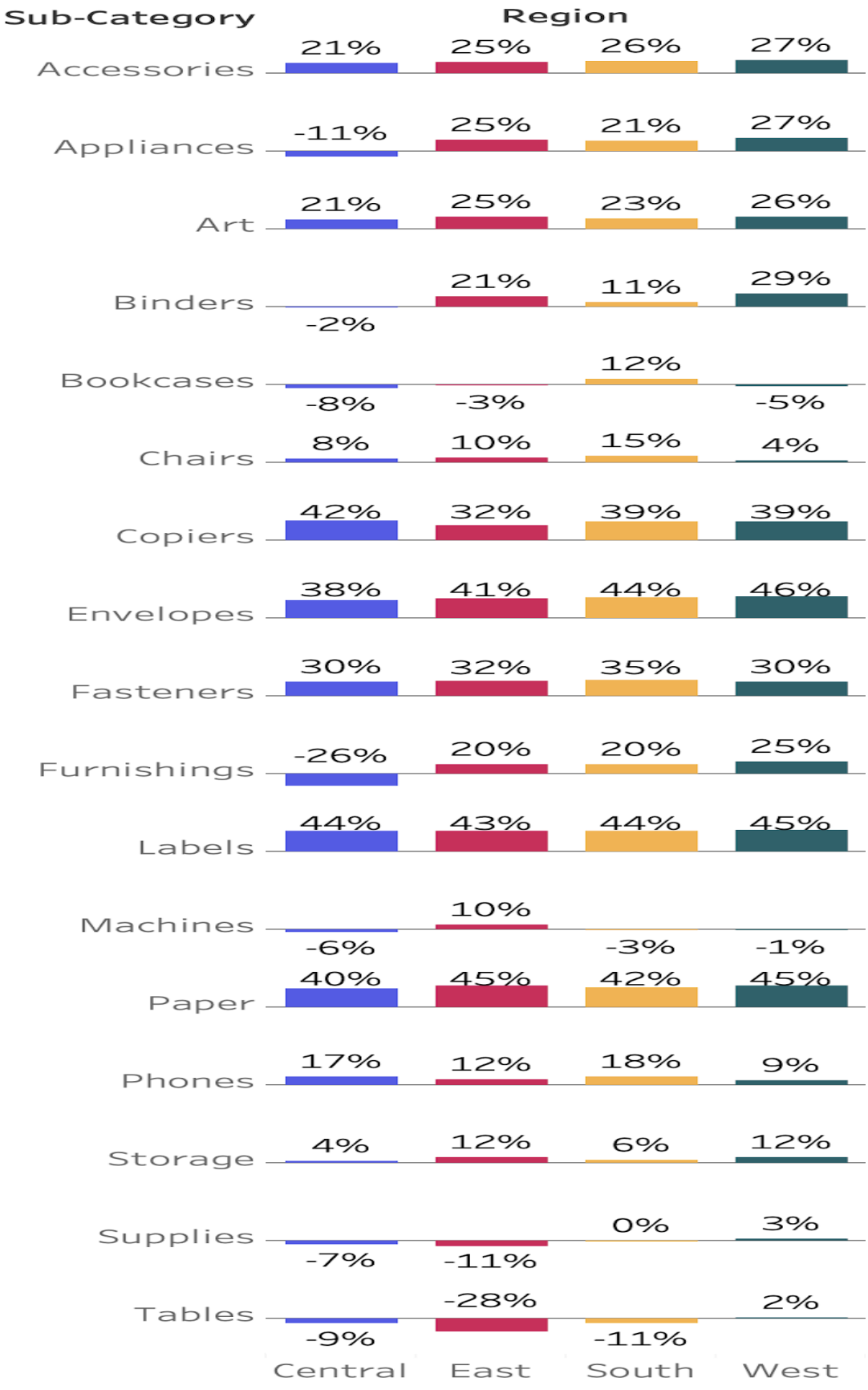

Sub-Category
Region
Accessories
21%
25%
26%
27%
Appliances
-11%
25%
21%
27%
Art
21%
25%
23%
26%
Binders
-2%
21%
11%
29%
Bookcases
-8%
-3%
12%
-5%
Chairs
8%
10%
15%
4%
Copiers
42%
32%
39%
39%
Envelopes
38%
41%
44%
46%
Fasteners
30%
32%
35%
30%
Furnishings
-26%
20%
20%
25%
Labels
44%
43%
44%
45%
Machines
-6%
10%
-3%
-1%
Paper
40%
45%
42%
45%
Phones
17%
12%
18%
9%
Storage
4%
12%
6%
12%
Supplies
-7%
-11%
0%
3%
Tables
-9%
-28%
-11%
2%
Central
East
South
West

Figure 3-31. Even this side-by-side bar chart consumes too much space.

While this chart provides clarity within each region, it's long and it limits the reader's ability to make comparisons across subcategory. An alternative that might work well here is the barbell plot–a sibling to the dot plot.

The barbell plot places dots–each representing one dimension–along several lines where each line is the second dimension and the measure of interest is the continuum of the line (Figure 3-31).

Barbell Plot

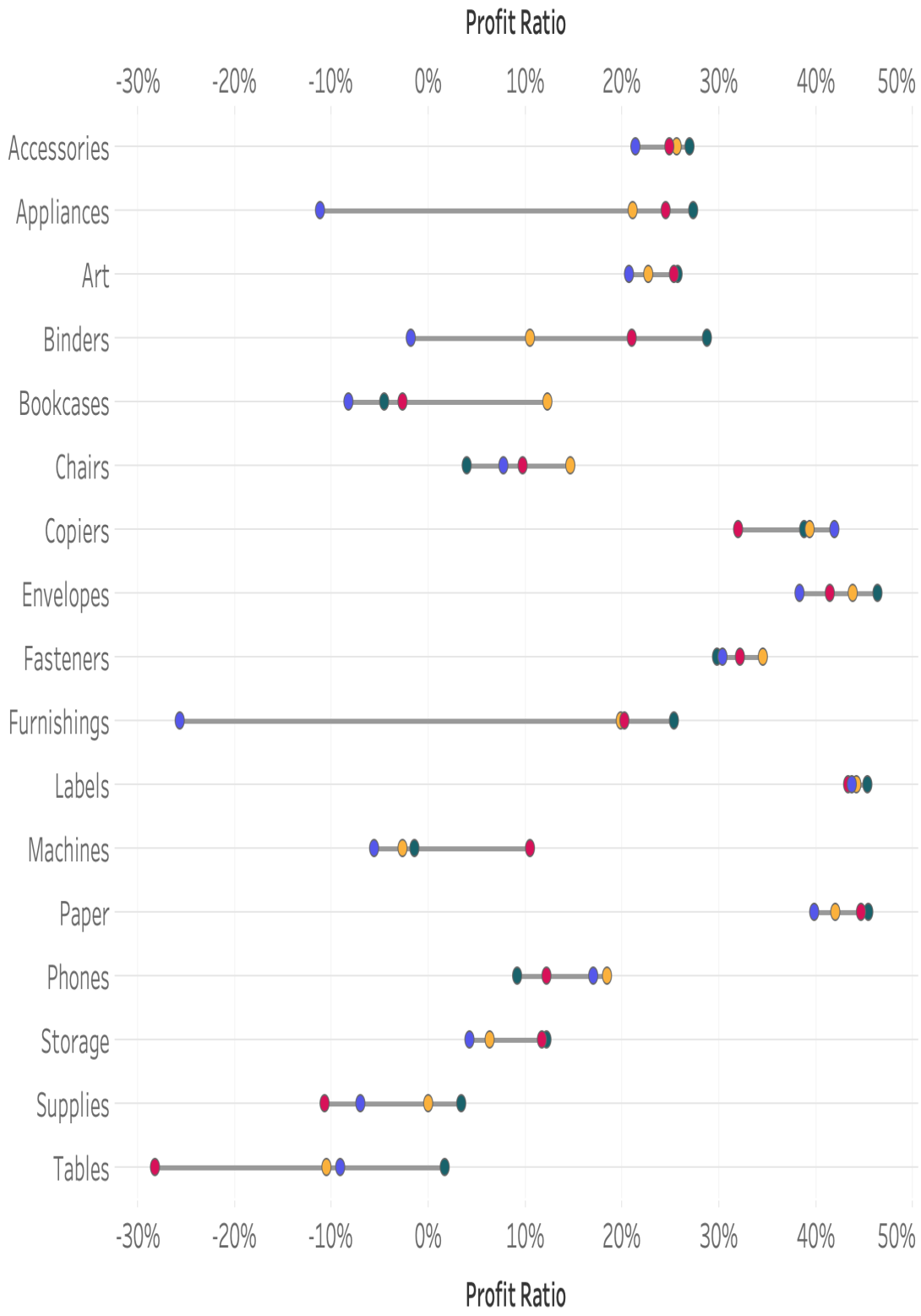


Figure 3-32. Barbell plots can significantly consolidate a visualization that might have otherwise been side-by-side bar charts.

The result is a concise visual representation that allows your audience to quickly understand the performance within a measure while still grasping overall performance. The success of the barbell chart is not just in the plot design but the formatting. With the chart in Figure 3-31, your audience has a guideline that allows them to follow an individual subcategory from left to right or quickly scan up or down to compare values.

So how do we build this visualization?

Blueprint: Building a Barbell Chart

Step 1: Create the base visualization

Step 1a: Rows. Connect to the Superstore dataset. Add Subcategory to rows.

Step 1b: Marks card. Add Region to color. For this example we've customized the colors.

Step 1c: Calculation. Create a calculation called % Margin where:

// % Margin

SUM([Profit]) / SUM([Sales])

Step 1d: Columns. Add % Margin calculation twice to the columns shelf (Figure 3-32).

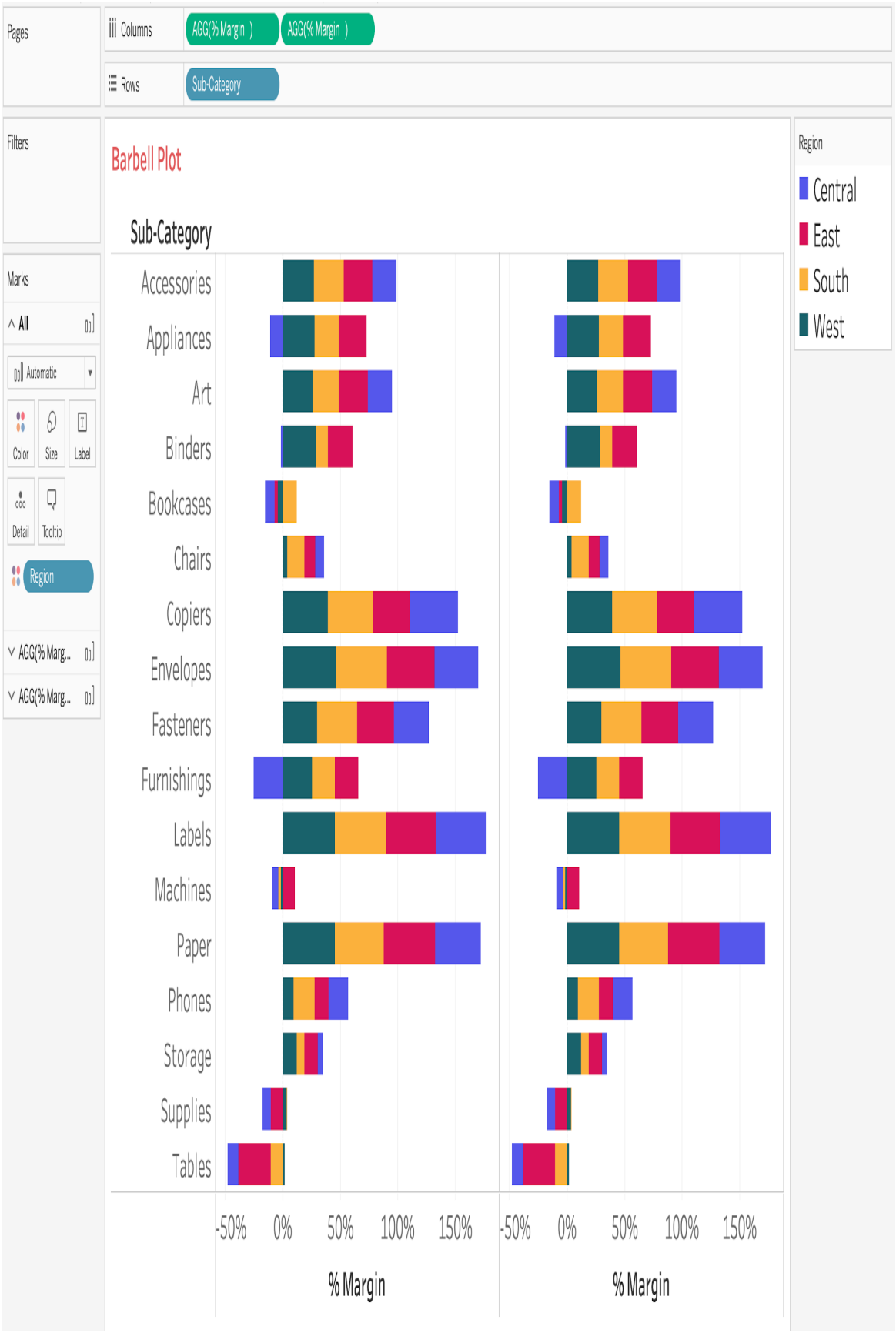

Pages
Columns
AGG(% Margin)
AGG(% Margin)
Rows
Sub-Category
Filters
Barbell Plot
Region
Central
East
South
West
Marks
All
Automatic
Color
Size
Label
Detail
Tooltip
Region
AGG(% Marg...
AGG(% Marg...
Sub-Category
Accessories
Appliances
Art
Binders
Bookcases
Chairs
Copiers
Envelopes
Fasteners
Furnishings
Labels
Machines
Paper
Phones
Storage
Supplies
Tables
-50%
0%
50%
100%
150%
% Margin
-50%
0%
50%
100%
150%
% Margin

Figure 3-33. To create a barbell plot start with stacked bar charts.

Step 2: Customize the marks card

With the left-most % Margin marks card, change the mark type from Automatic to Line. Move Region from color to path.

With the right-most % Margin marks card change the mark type from Automatic to Circle. Format the color by adding a border to the circle marks.

Create a synchronized dual axis chart.

Step 3: Perfect the formatting

Step 3a: Rows Shelf. You are going to add an in-line calculation to the rows shelf. Double-click to the right of Subcategory on the rows shelf and type MIN(0.0). This will create the ruler that audiences will use to move from subcategory to values on the axis. Hit enter. (See Figure 3-33.)

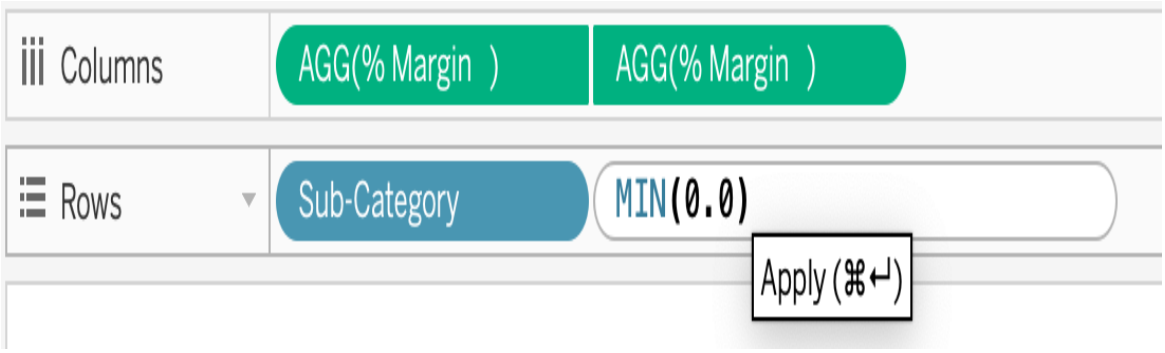


Figure 3-34. How to add MIN(0.0) as an ad-hoc calculation.

This will give you a new vertical axis. Right-click and hide the new axis (Figure 3-34).

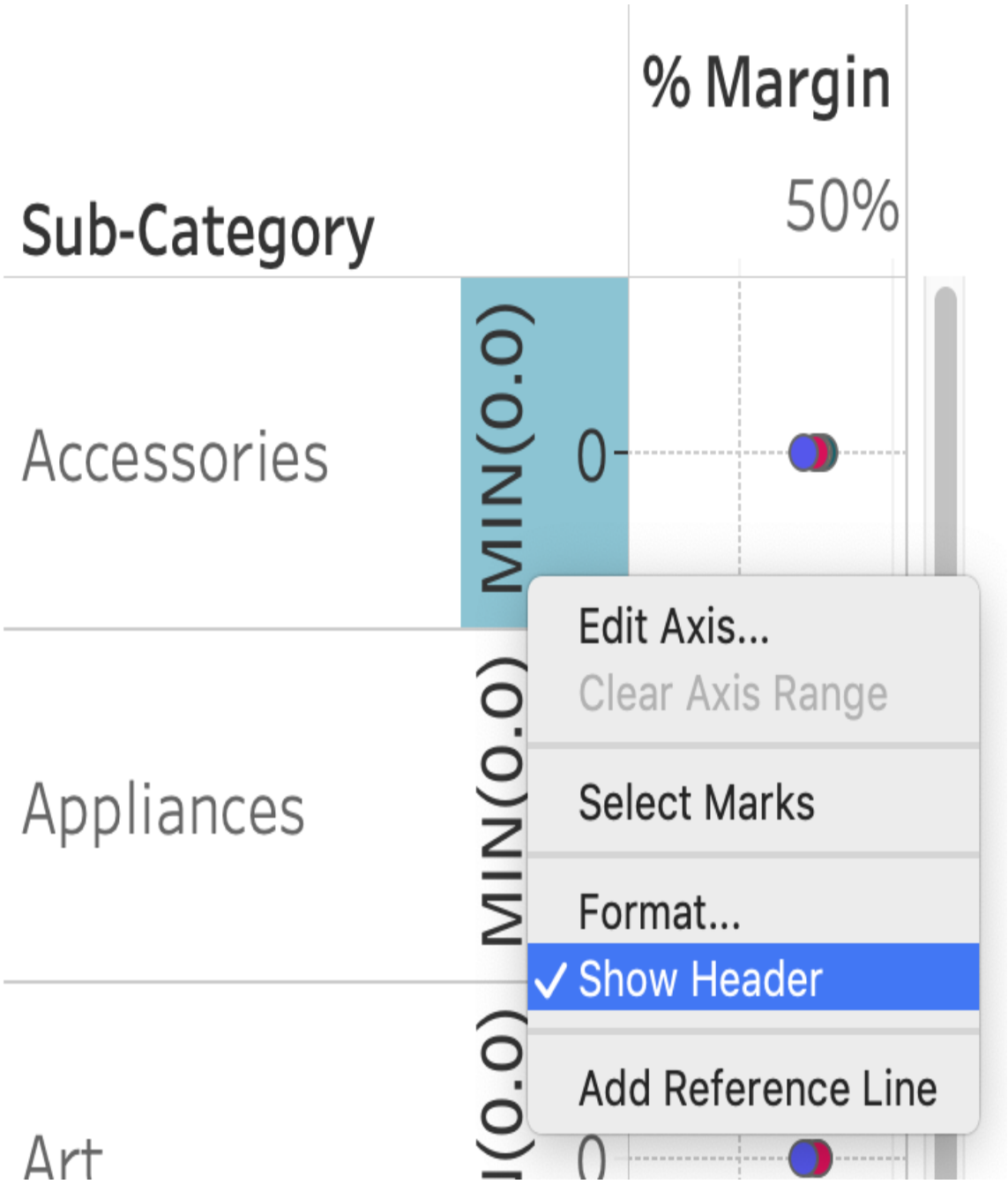


Figure 3-35. Uncheck show header to hide the axis.

Step 3b: Formatting. Resize the subcategories so they belong to a concise view.

Format and remove all border lines except for column grid lines and rows zero axis.

Format the line type and color of these to be identical.

On the header where it says Subcategory, right-click and select Hide Field Labels for Rows (we'll do that a lot through the book).

Right-click on any subcategory and format the header to the members that are right-aligned.

Right-click on the MIN(0.0) header and uncheck Show Header.

Finally, size the circles so they are noticeable but not distracting for your audience. Keep the line smaller, but don't use the smallest (to distinguish it from the guide). Your result should look something like Figure 3-36.

Barbell Plot

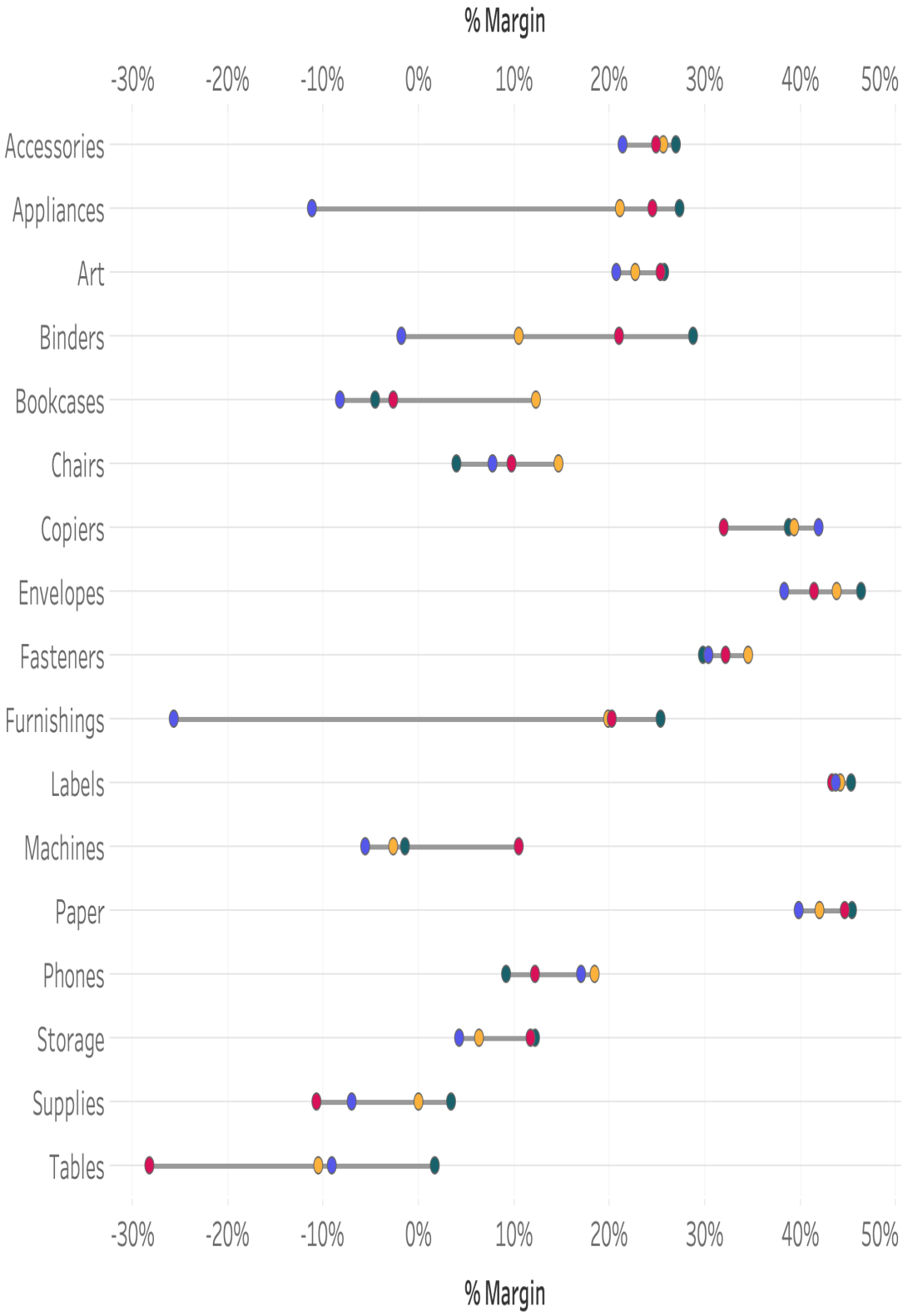


Figure 3-36. : The final version of the barbell plot.

Another alternative is to make the lines slightly larger than the circles (Figure 3-36). Whether you go with wide or narrow lines, you can use the border color of the circles to help distinguish the color and shape of the circles from the color of the lines.

Barbell Plot

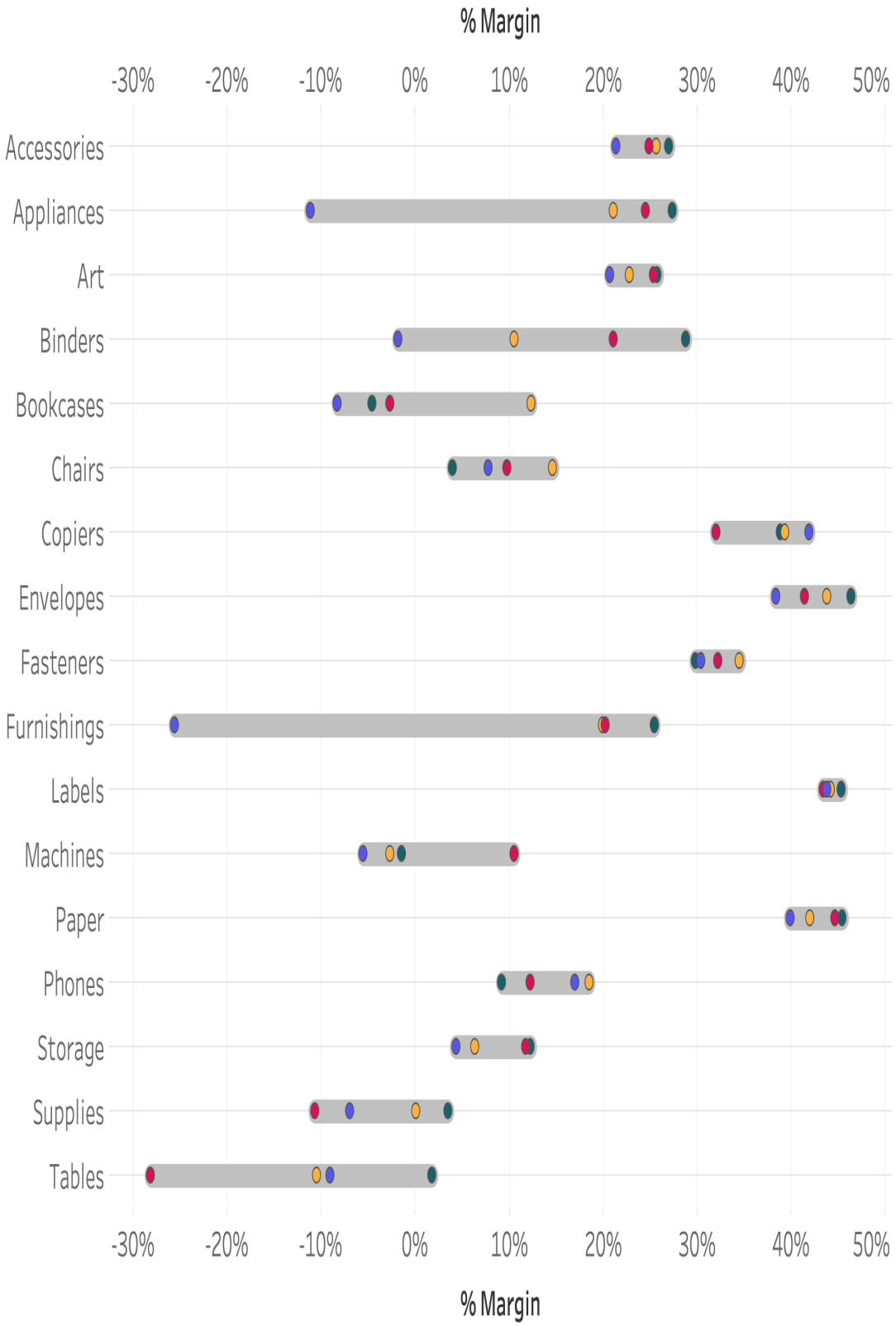


Figure 3-37. An alternative barbell plot where the lines between the dots are much larger.

The result of the barbell plot is a concise chart that lets your audience quickly compare values within a group–or even across groups.

Trellis Charts

Consider the line chart in Figure 3-37, which shows total sales by quarter over four years and is broken down by the 17 subcategories.

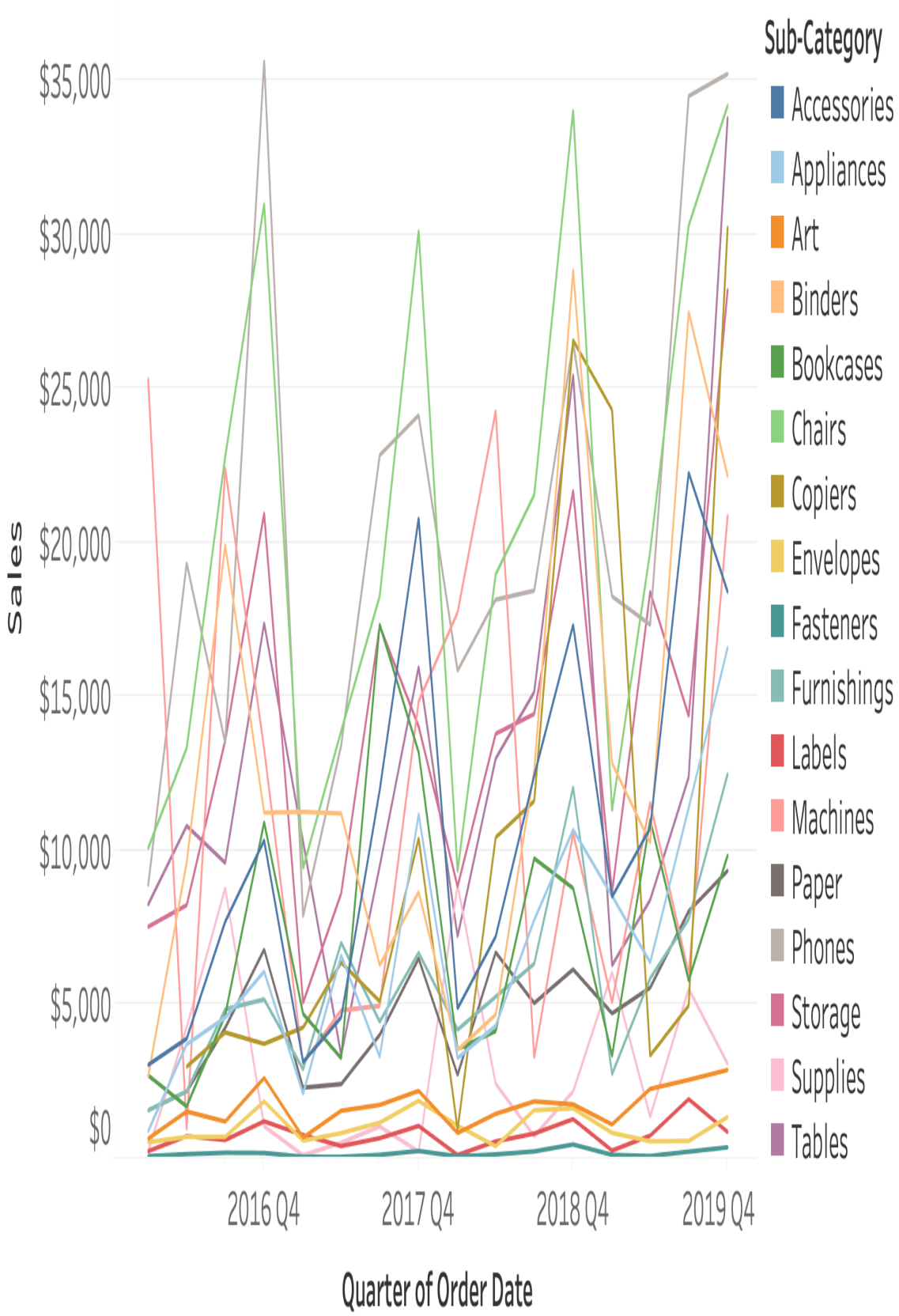

Sub-Category
Accessories
Appliances
Art
Binders
Bookcases
Chairs
Copiers
Envelopes
Fasteners
Furnishings
Labels
Machines
Paper
Phones
Storage
Supplies
Tables
$35,000
$30,000
$25,000
$20,000
$15,000
$10,000
$5,000
$0
Sales
2016 Q4
2017 Q4
2018 Q4
2019 Q4
Quarter of Order Date

Figure 3-38. Another less-than-ideal line chart showing sales by subcategory.

Imagine showing this plot to the board of your office-supply company. Their responses would probably be something along the lines of “What the heck is this crap? It’s too messy!”

Almost no one wants to see a line plot with more than three lines. The board will want to trace a single line, and it’s impossible to follow just one of the lines in this plot–even with unique color encoding. You’ve got to find a better way to communicate this information. One we prefer is a small multiple, or trellis, chart.

A trellis chart is actually a series of charts that use the exact same axes, most often ordered in a grid. Rather than showing a single line plot with seventeen different lines, here you can choose to show 17 individual charts–in Figure 3-38, bar charts–to show the pattern in the data.

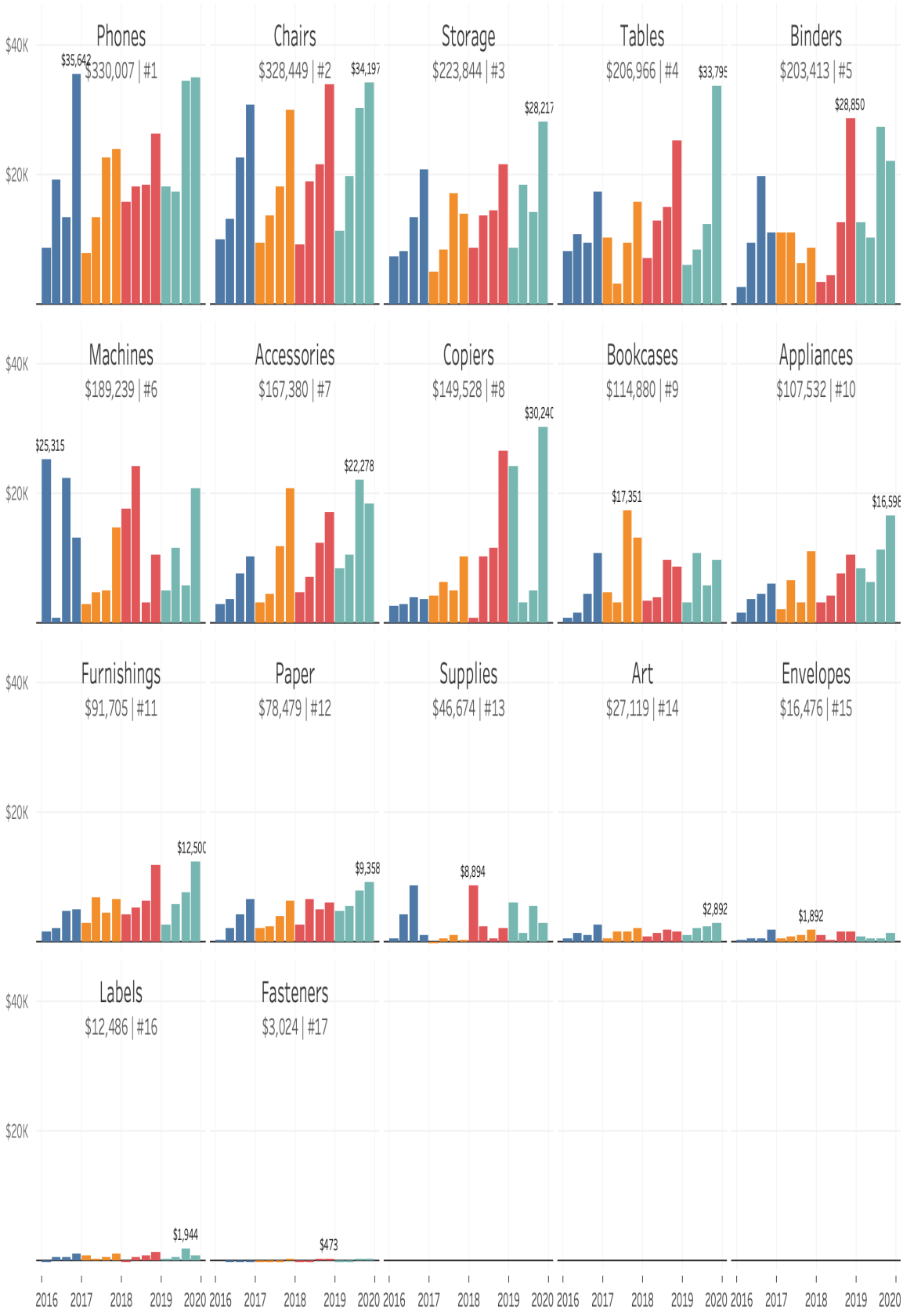

Phones
$330,007 | #1
$35,642
Chairs
$328,449 | #2
$34,197
Storage
$223,844 | #3
$28,217
Tables
$206,966 | #4
$33,795
Binders
$203,413 | #5
$28,850
Machines
$189,239 | #6
$25,315
Accessories
$167,380 | #7
$22,278
Copiers
$149,528 | #8
$30,240
Bookcases
$114,880 | #9
$17,351
Appliances
$107,532 | #10
$16,598
Furnishings
$91,705 | #11
$12,500
Paper
$78,479 | #12
$9,358
Supplies
$46,674 | #13
$8,894
Art
$27,119 | #14
$2,892
Envelopes
$16,476 | #15
$1,892
Labels
$12,486 | #16
$1,944
Fasteners
$3,024 | #17
$473
$40K
$20K
2016
2017
2018
2019
2020

Figure 3-39. Small multiples can be used to show patterns within a group across time.

Retail case study: Trellis Charts (Small Multiples)

The plot in Figure 3-38 shows total sales by quarter for the 17 subcategories. Each year has a distinct color to aid with analysis. We also added labels to the maximum value within each subcategory to provide additional context.

With small multiples your audience can track patterns in the data. In this case they will be able to see the seasonality of the data as well as the overall performance of subcategories. They can see how Phone and Chairs are the top sellers. They can note the consistent seasonality in Chairs, Tables, Accessories, and Paper. And they can note that in Q2 and Q3 of 2019, copier sales dipped significantly before picking up to normal values in Q4. Now your audience can track individual values.

One detractor to this chart type is not knowing the actual values for any particular bar–but this is the struggle of almost every visualization. Additionally, it can be very difficult to compare values across subcategory.

But that’s the battle with any chart–there are always trade-offs. Your job is to weigh the options and select the best chart type.

The difficulty of building a trellis chart in Tableau varies based on the desired result. You could create a simple trellis by examining sales region and business segment, as in Figure 3-39.

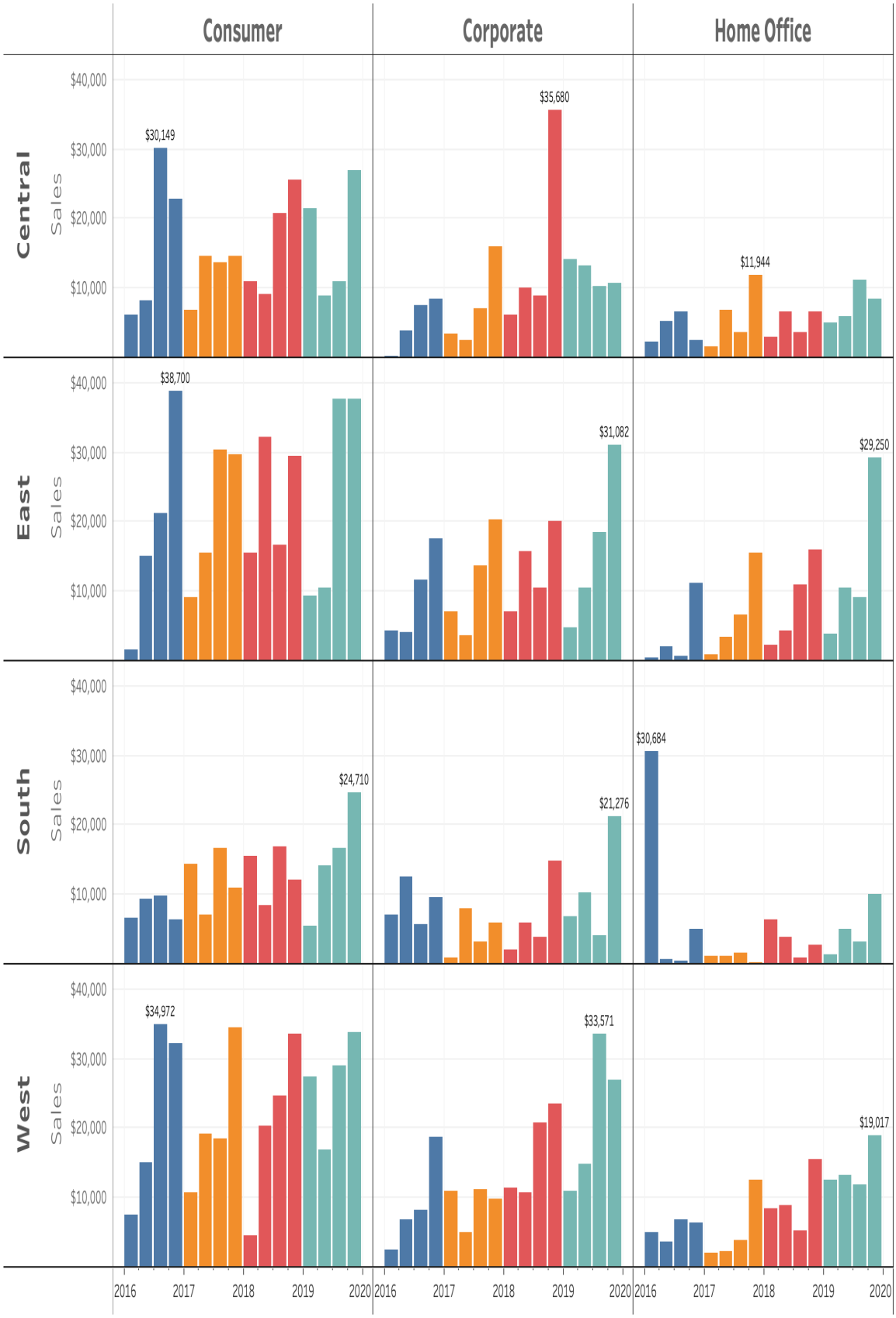
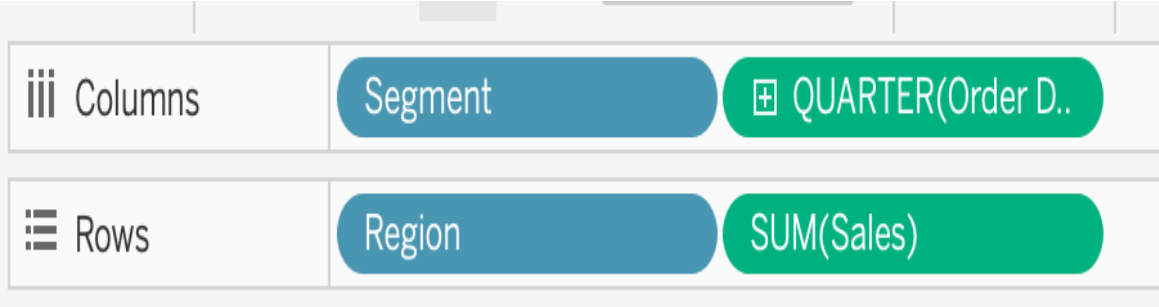

Consumer
Corporate
Home Office
Central
East
South
West
Sales
$40,000
$30,000
$20,000
$10,000
$30,149
$35,680
$11,944
$38,700
$31,082
$29,250
$24,710
$21,276
$30,684
$34,972
$33,571
$19,017
2016
2017
2018
2019
2020

Figure 3-40. A simple small multiple where two dimensions are used to define rows and columns.

This type of trellis is very easy to complete in Tableau: it requires at least one dimension on rows and another dimension on columns (Figure 3-40).



A closeup of Segment on columns and Region on rows.

The real challenge is creating a grid of small multiples using a single dimension. In the next blueprint, we ask you to make a trellis with a grid from just a single dimension. Creating this chart is not out-of-the-box by any means. It requires you to think carefully about how you'll design and label your grid.

Let's do this!

Blueprint: Creating a trellis chart for a single dimension

Using the Sample – Superstore dataset, create a new sheet and follow the steps below.

Step 1: Create the grid.

Creating the grid requires you to use table calculations. You will write three calculations and create one parameter in this step.

Step 1a: Calculate Index. First, let's create a calculation that we will use to sort the members that will be displayed on the grid. Create a new calculation called Index where the calculation is:

// Index

INDEX()

You are going to use this calculation inside two other calculations that will form the columns and rows.

Step 1b: Create a parameter. Create a new integer parameter called Total Columns. This will provide us a way to dynamically change the number of columns in our trellis chart. Set the value to 5.

Step 1c: Calculation for columns. Create a calculation called Columns this will encode each member to the appropriate column:

// Columns

([Index] - 1) % [Total Columns]

This calculation takes the index function, a calculation that creates a running count of values from 1 to any number and applies the modulo function. If you are unfamiliar with the modulo operator it calculates the remainder to any division problem. Remember back in grade school when you calculated the remainder of a division problem and not the decimal? I bet you never thought you'd be using it as an adult!

The output from this calculation provides you with members of a dimension grouped into different groups based on whatever number you specified in the Total Columns parameter: here, five different groups.

Figure 3-41 shows the mapping you'd expect to see:

Calculation	Result											
Index	1	2	3	4	5	6	7	8	9	10	11	12
Columns	0	1	2	3	4	0	1	2	3	4	0	1

Figure 3-41: Output of the Index and Columns calculations

Step 1d: Calculate rows. Now that you have the Columns calculation, all you need is the Rows calculation:

// Rows

((([Index] - 1) - [Rows]) - 1) / [Total Columns]

Add Subcategory and Index to detail. Add Columns to the columns shelf and Rows to the rows shelf. Change Columns and Rows to discrete.

Step 1e: Build the grid on the sheet. Edit the table calculations for Column, Rows, Index and select Specific Dimensions and choose Subcategory. Right-click on Subcategory on the marks card and create a custom sort using sum of sales, descending. For the Rows calculation you will have two table calculations to edit. Make sure the table calculations follow the same compute using and sort order (Figure 3-42).

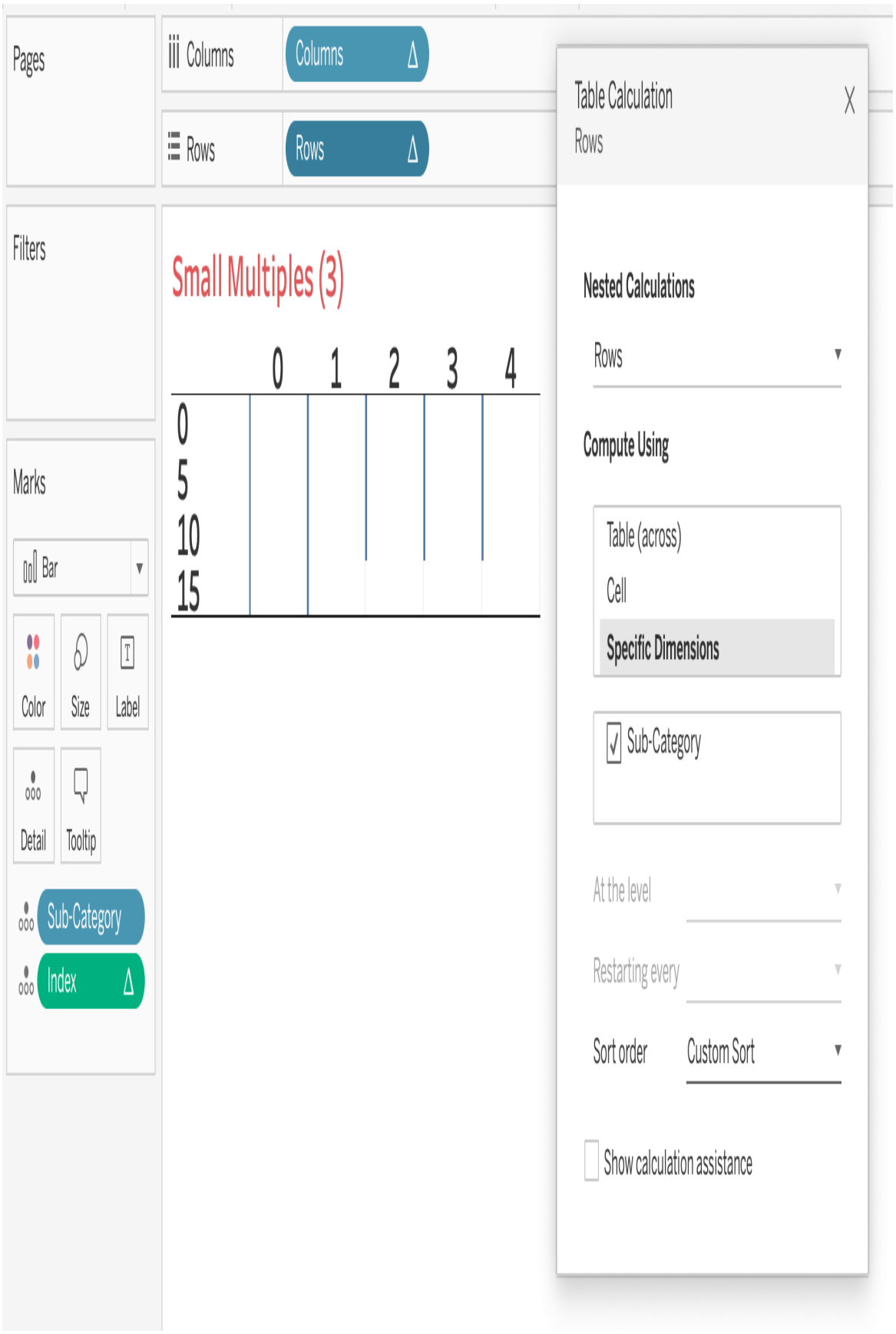

Pages
Columns
Columns
Rows
Rows
Filters
Marks
Bar
Color
Size
Label
Detail
Tooltip
Sub-Category
Index
Small Multiples (3)
0 1 2 3 4
0
5
10
15
Table Calculation
Rows
Nested Calculations
Rows
Compute Using
Table (across)
Cell
Specific Dimensions
Sub-Category
At the level
Restarting every
Sort order
Custom Sort
Show calculation assistance

Figure 3-42. Use table calculations to build the grid for the small multiples.

Step 2: Build the base visualization

Once you have established the grid for your visualization you can begin to build the visualization. You will look at sales based on order date.

Step 2a: Date value for columns: Click-and-drag Order Date onto columns, then right-click and select a continuous quarter date type (Figure 3-43).

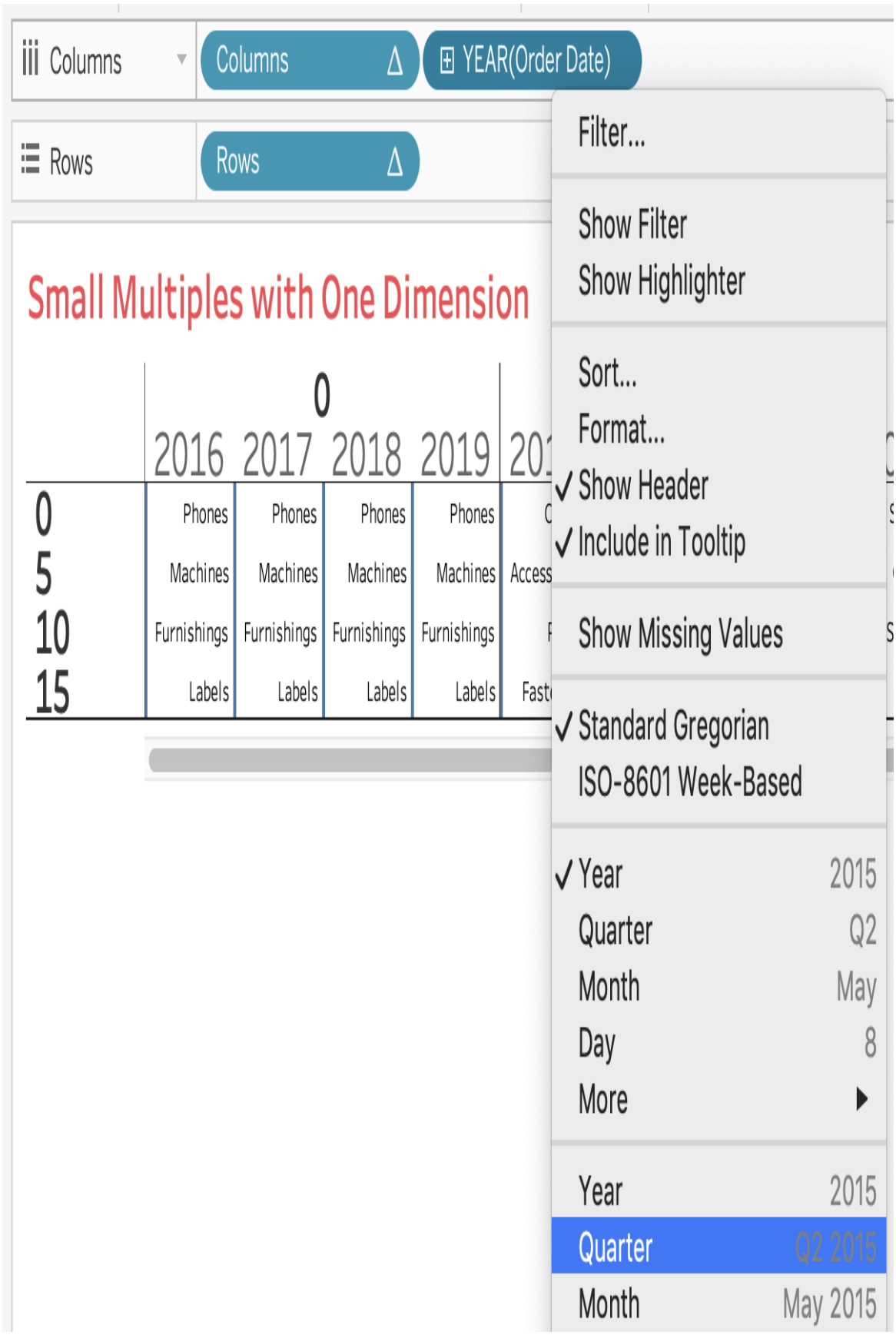

Columns
Columns
YEAR(Order Date)
Rows
Rows
Small Multiples with One Dimension
0
2016 2017 2018 2019 201
0
5
10
15
Phones
Machines
Furnishings
Labels
Access
Filter...
Show Filter
Show Highlighter
Sort...
Format...
✓ Show Header
✓ Include in Tooltip
Show Missing Values
✓ Standard Gregorian
ISO-8601 Week-Based
✓ Year 2015
Quarter Q2
Month May
Day 8
More
Year 2015
Quarter Q2 2015
Month May 2015

Figure 3-43. Change order date to a date value by quarter.

Step 2b: Measure for rows. Add SUM(Sales) to rows. Add Year(Order Date) to color. This will leave you with the following visualization:


Pages
Columns
Columns
QUARTER(Order D..
Rows
Rows
SUM(Sales)
Filters
Marks
Bar
Color
Size
Label
Detail
Tooltip
YEAR(Order..
Index
Sub-Category
Small Multiples with One Dimension
0
1
2
3
4
0
5
10
15
Sales
$30,000
$20,000
$10,000
2017
2019
Quarter of Order ..
YEAR(Order Date)
2016
2017
2018
2019

Figure 3-44. A small multiple chart without subcategory labels for each section of the grid.

You've got the base visualization complete. What would be nice would be labels for each box. To do this, you are going to create a dual axis chart with two custom calculations.

Step 3: Add labels to each multiple

To add a label that is centered above each multiple, create a custom date calculation that identifies the most middle date.

Step 3a: Calculate label location. We can do this by identifying the ends of our dates and then taking the average of two. Write the following calculation, called Order Date | SM Label.

```
// Order Date | SM Label

{MIN([Order Date])} + (({MAX([Order Date])} - {MIN([Order Date])})/2)
```

This finds the first and last order date and then finds the middle of the two dates. (More about this in chapter 4.)

Step 3b: Columns. Add this calculation as an exact date to the columns shelf. You'll notice a single stacked bar chart that corresponds to the new date calculation (Figure 3-45).


Pages
Columns
Columns
QUARTER(Order D..
Order Date | SM Label
Rows
Rows
SUM(Sales)
Filters
Small Multiples with One Dimension
0
1
2
3
4
Marks
All
Bar
Color
Size
Label
Detail
Tooltip
YEAR(Order..
Index
Sub-Category
QUARTER(Or...
Order Date | ...
$300,000
$200,000
$100,000
Sales
0
5

Figure 3-45. A work-in-progress for obtaining proper labels.

Step 4c: Text. Edit the marks card for Order Date | SM Label. Change the mark type to text and remove YEAR(Order Date) from color.

Step 4d: Axes. Finally, create a synchronized dual axis. On this same marks card, move Subcategory and Index from detail to text.

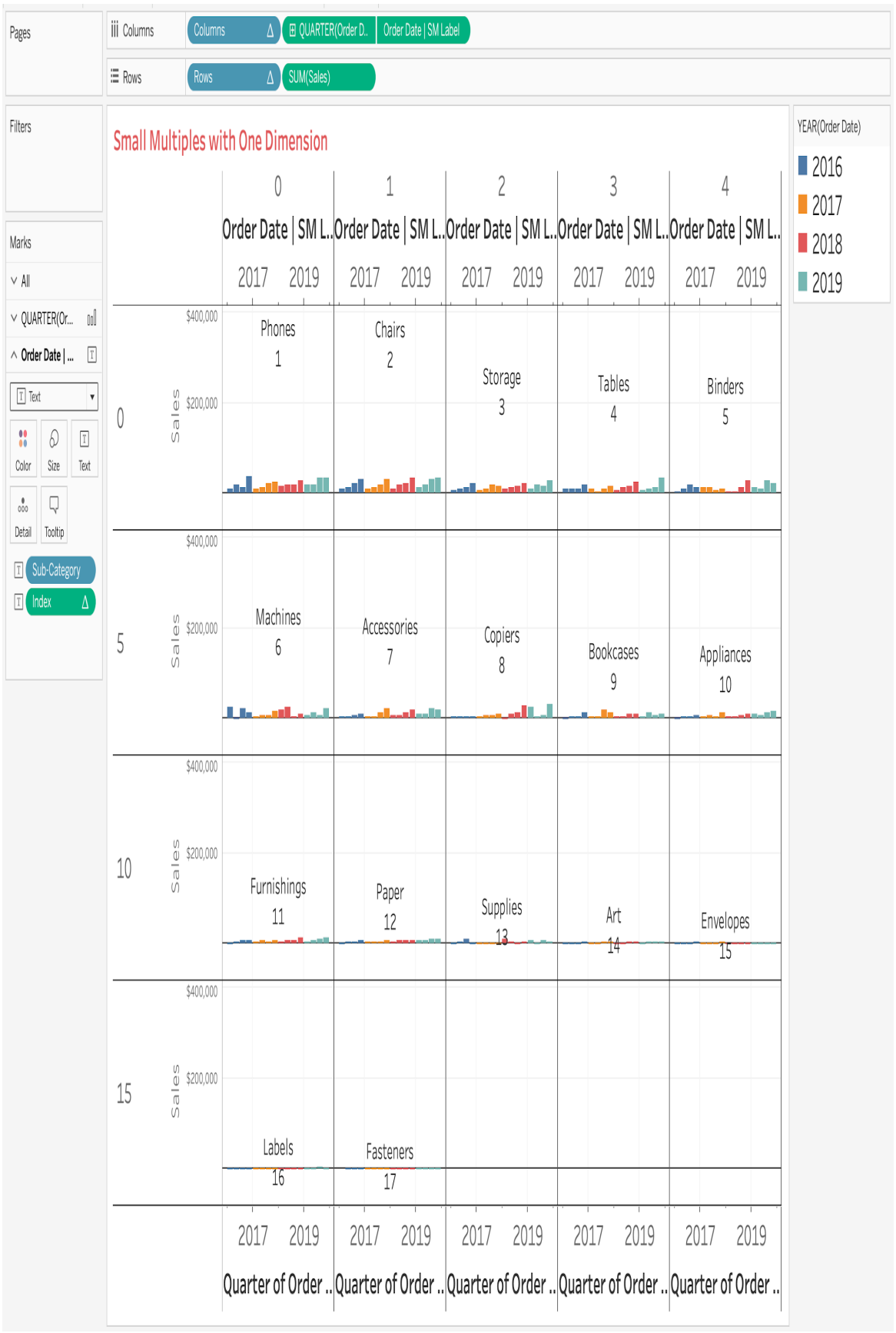

Pages
Columns
Columns
QUARTER(Order D..
Order Date | SM Label
Rows
Rows
SUM(Sales)
Filters
Marks
All
QUARTER(Or...
Order Date | ...
Text
Color
Size
Text
Detail
Tooltip
Sub-Category
Index
Small Multiples with One Dimension
YEAR(Order Date)
2016
2017
2018
2019
0
1
2
3
4
Order Date | SM L..
2017
2019
$400,000
$200,000
Sales
Phones
1
Chairs
2
Storage
3
Tables
4
Binders
5
Machines
6
Accessories
7
Copiers
8
Bookcases
9
Appliances
10
Furnishings
11
Paper
12
Supplies
13
Art
14
Envelopes
15
Labels
16
Fasteners
17
Quarter of Order ..

Figure 3-46. Labels are now showing for the small multiples but the height varies by cell of the small multiples grid.

One issue you'll see in Figure 3-46 is that the text doesn't align across each subcategory. The reason is for this is because, for the Order Date | SM axis, each label is positioned at the overall total for that subcategory. We need to find some magic (or a nifty way) to align these labels.

Step 5. Align labels.

Find the single largest quarterly total for a subcategory, then just adjust the label to be some multiple higher than that value. This means that if the highest total sales for any category-by-quarter came, hypothetically, from Copiers in Q4 of 2018, and that value was $100–then you'd want the labels for all the subcategories to be slightly higher on the chart–perhaps centered at 120.

But here's the thing: you need text to be centered, *and* you need the bars to appear at the same height. And you can only do this with a single calculation! How do you accomplish it? By figuring out how many marks are occurring per calculation using the SIZE() function!

Step 5a: Create calculation. Create a calculation called Bars + Label.

```
// Bars + Label

IF SIZE() > 1

THEN SUM([Sales])

ELSE 1.1 * AVG({

MAX(

{FIXED [Subcategory], DATETRUNC("quarter", [Order Date]) :

SUM([Sales])

})

})

END
```

In this calculation, SIZE(), first count the marks based on how you set the table calculation. For the text marks this will return a value of 1. For the bar charts SIZE() will return the number of quarters displaying for each subcategory. For example this is 16. Since the value is 16 for the bars the calculation will return the total sales. For the text it returns 1.1 times more than the highest total for a single subcategory-by-quarter.

Step 5b: Rows. Take this Bars + Label and replace SUM(Sales) and rows with the calculation. Edit the table calculation and select by Quarter of Order Date. Add SUM(Sales) to label on the Order | SM Label marks card. This will leave you with the visualization in Figure 3-47.


Pages
Columns
Columns
QUARTER(Order D..
Order Date | SM Label
Rows
Rows
Bars + Label
Filters
Marks
All
QUARTER(Or...
Order Date | ...
Text
Color
Size
Text
Detail
Tooltip
Sub-Category
Index
SUM(Sales)
Small Multiples with One Dimension
YEAR(Order Date)
2016
2017
2018
2019
0
1
2
3
4
Order Date | SM L..
2017
2019
Phones
1
$330,007
Chairs
2
$328,449
Storage
3
$223,844
Tables
4
$206,966
Binders
5
$203,413
Machines
6
$189,239
Accessories
7
$167,380
Copiers
8
$149,528
Bookcases
9
$114,880
Appliances
10
$107,532
Furnishings
11
$91,705
Paper
12
$78,479
Supplies
13
$46,674
Art
14
$27,119
Envelopes
15
$16,476
Labels
16
$12,486
Fasteners
17
$3,024
Bars + Label
$40K
$20K
0
5
10
15
Quarter of Order ..

Figure 3-47. The trellis chart now has labels that are located in the same position, but just need formatting.

Step 6: Finalize the visualization.

For the last steps, you'll want to format the text showing on the Order Date | SM Label.

Step 6a: Edit text. You can format this text how you prefer; we are putting subcategory on the first line and the Index (which acts like a rank) and SUM(Sales) on the second line (Figure 3-48).

Figure 3-48. An example of how the text might be formatted for the visualization.

Step 6b: Labels. Add SUM(Sales) to the label of the quarter of order date marks card. We're setting the font to size 7, and then showing the top value per subcategory, which can be done by selecting Pane in the scope section and then selecting label maximum value (Figure 3-49).


Show mark labels
Label Appearance
Text: <SUM(Sales)>
...
Font: Tableau Book, 7pt, ..
Alignment: Automatic
Marks to Label
All
Selected
Min/Max
Highlighted
Most Recent
Scope
Table
Cell
Pane
Line/Pie
Field
Automatic
Options
Allow labels to overlap other marks
Label minimum value
Label maximum value
Accessories
$167,380 | #7
$22,278
Paper
$78,479 | #12
$12,500
$9,358
Fasteners
$3,024 | #17

Figure 3-49. Add labels the maximum value on the bars by selecting min/max, then pane, then checking label maximum value.

Step 6c: Format. Adjust the lines and borders to your preference. For this example, we prefer to keep gridlines and include row and column dividers that are set to the background color. The final result in Figure 3-50 is a 5x4 grid of small multiples.

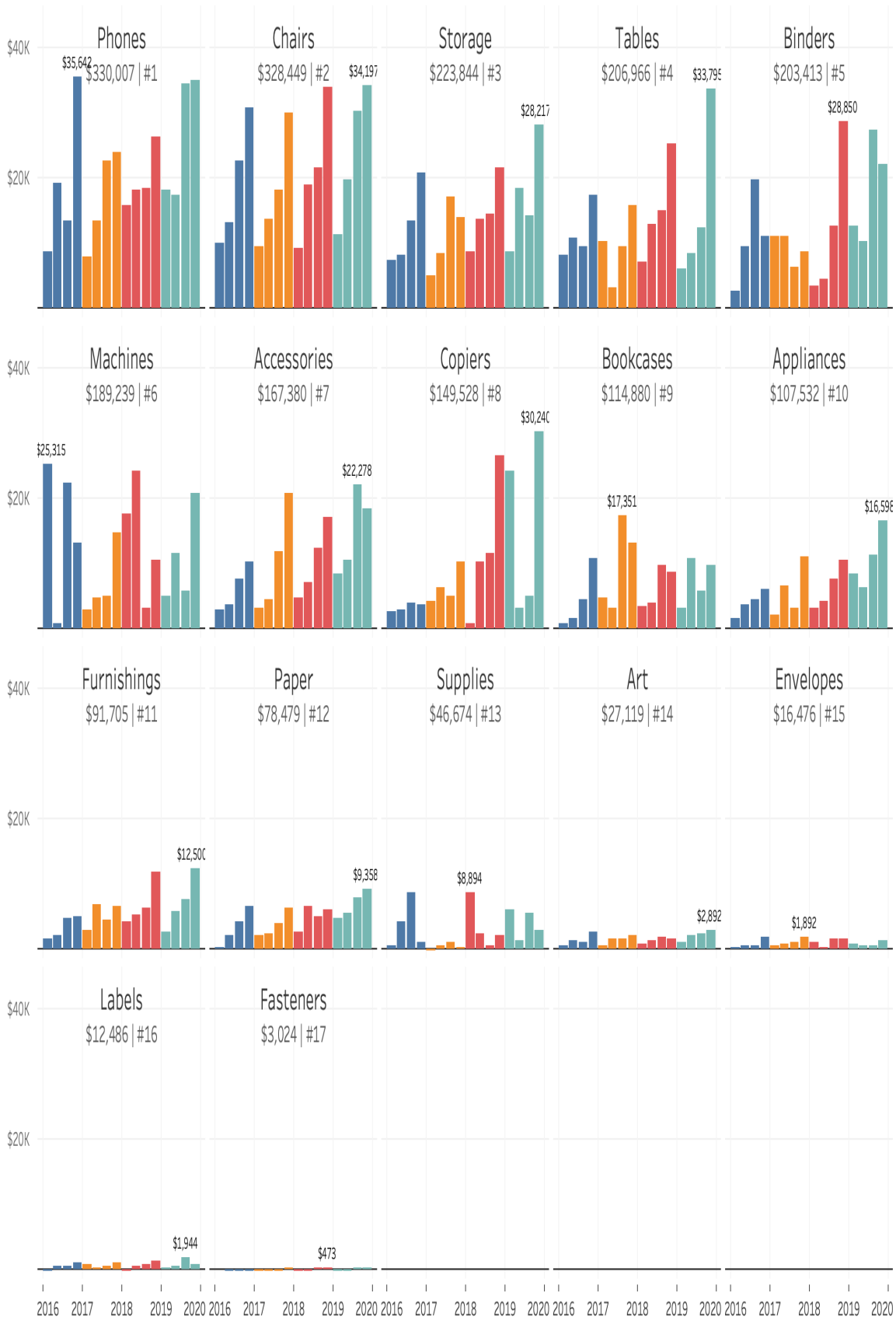

Phones
$330,007 | #1
$35,642
Chairs
$328,449 | #2
$34,197
Storage
$223,844 | #3
$28,217
Tables
$206,966 | #4
$33,795
Binders
$203,413 | #5
$28,850
Machines
$189,239 | #6
$25,315
Accessories
$167,380 | #7
$22,278
Copiers
$149,528 | #8
$30,240
Bookcases
$114,880 | #9
$17,351
Appliances
$107,532 | #10
$16,596
Furnishings
$91,705 | #11
$12,500
Paper
$78,479 | #12
$9,358
Supplies
$46,674 | #13
$8,894
Art
$27,119 | #14
$2,892
Envelopes
$16,476 | #15
$1,892
Labels
$12,486 | #16
$1,944
Fasteners
$3,024 | #17
$473
$40K
$20K
2016
2017
2018
2019
2020

Figure 3-50. The final result is an organized trellis chart that allows your audience to examine patterns within subcategory.

Step 6d: Filter. If you are showing hundreds of multiples, you may want to filter to, perhaps, the top 20. You can do this by filtering using your Index calculation, and filtering to a range between 1 and 20. Just be sure your table calculation on the filter is set up correctly.

With a trellis, you can create multiple plots with the same base framework and examine how subsets of your data look relative to each other. This allows for comparisons within a subset and across subsets. You can create a trellis chart in Tableau with multiple dimensions, or with a single dimension and a few handy table calculations. This particular example tackled a single dimension trellis with dates. This is perhaps the most difficult to create!

Parallel Coordinates Plots

For our last blueprint of the chapter, let’s tackle a comparison that analysts are often forced to make: comparing multiple members of a single dimension across multiple measures. The most common way this is done is with a bar chart, as in Figure 3-51.

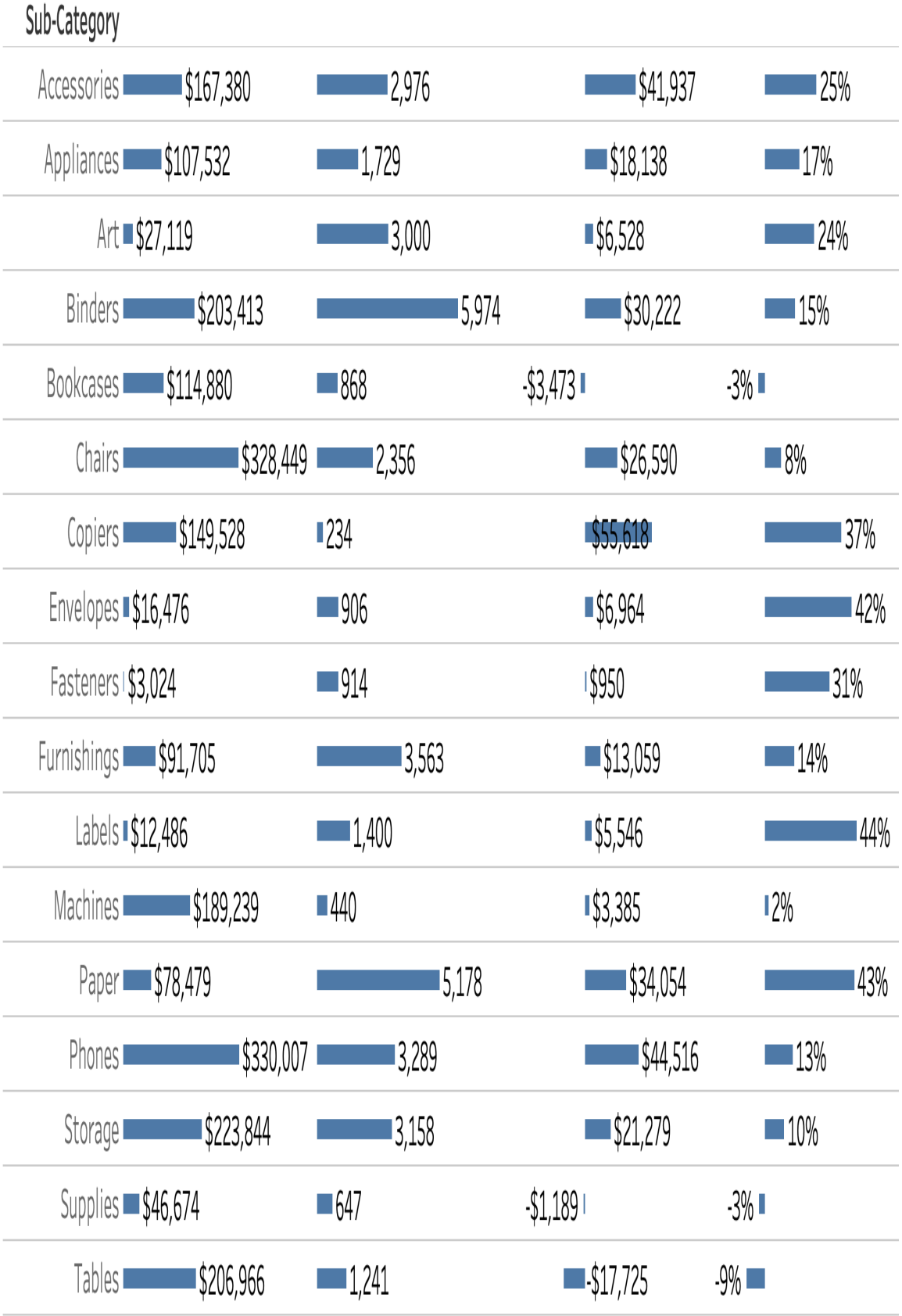

Sub-Category
Accessories $167,380 2,976 $41,937 25%
Appliances $107,532 1,729 $18,138 17%
Art $27,119 3,000 $6,528 24%
Binders $203,413 5,974 $30,222 15%
Bookcases $114,880 868 -$3,473 -3%
Chairs $328,449 2,356 $26,590 8%
Copiers $149,528 234 $55,618 37%
Envelopes $16,476 906 $6,964 42%
Fasteners $3,024 914 $950 31%
Furnishings $91,705 3,563 $13,059 14%
Labels $12,486 1,400 $5,546 44%
Machines $189,239 440 $3,385 2%
Paper $78,479 5,178 $34,054 43%
Phones $330,007 3,289 $44,516 13%
Storage $223,844 3,158 $21,279 10%
Supplies $46,674 647 -$1,189 -3%
Tables $206,966 1,241 -$17,725 -9%

Figure 3-51. Multiple bar charts showing sales, units sold, margin, and percent of margin. This chart works, but there are alternatives.

The one pitfall here is that you could have too many members/rows to manage; you may prefer to summarize your data in a more concise format.

You can do this with a parallel coordinates plot. Parallel coordinates are used for plotting data with multiple dimensions, and comparing the relationships across each dimension. With this chart type, each variable has its own axis. These axes are often on different units of measurement and are normalized for each measure. This allows the scales to remain uniform. A line is then used to connect each member of a dimension across the measures and their corresponding axes.

Because many members are often plotted at a single time, the plot can become cluttered very quickly. We recommend that you either highlight a single member relative to the entire group or minimize the number of members you compare. We prefer highlighting a single member of a dimension, as shown in Figure 3-52.

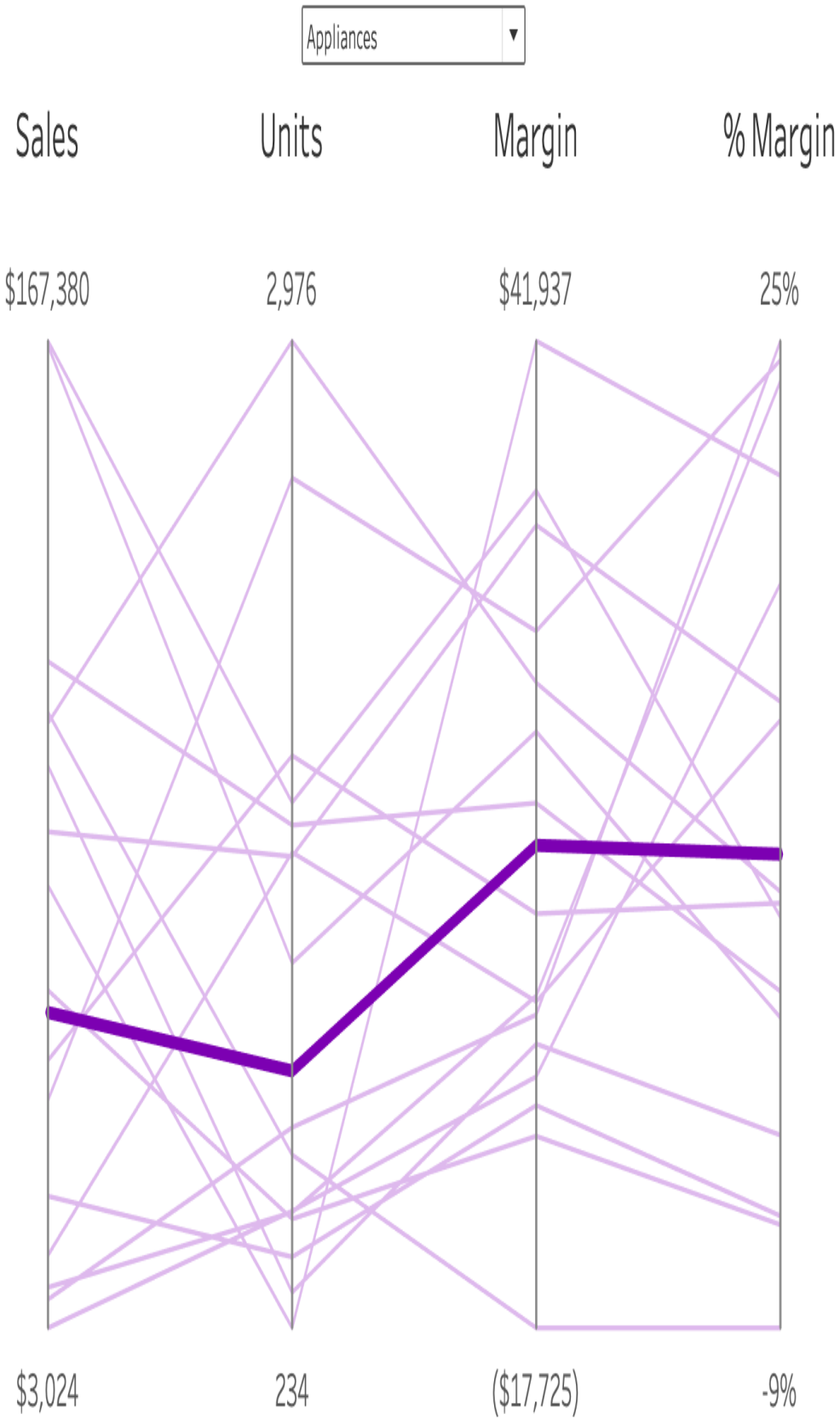

Appliances
Sales
Units
Margin
% Margin
$167,380
2,976
$41,937
25%
$3,024
234
($17,725)
-9%

Figure 3-52. A parallel coordinates plot can allow audiences to compare any member across multiple dimensions very quickly.

A parallel coordinates plot can quickly compare any member of a dimension across multiple measures, providing context faster than a bar chart often can. Additionally, the plot isn't best for comparing the relationship between any two variables–this is better left for a scatterplot.

Retail case study: Comparing across multiple measure s

You've been tasked with comparing the seventeen subcategories of a retailer across four different metrics: total sales, total units solid, the margin or profit in dollars, and the margin as a percent of sales. You want to compare performance of each subcategory relative to the others.

The best way to do this is with a parallel coordinates chart.

Blueprint: Building a parallel coordinates chart

Step 1: Build normalized metrics

First you need to build *normalized metrics*. This means taking any subset of metrics and transforming them from their existing scale to a scale that goes from zero to one. The format for this calculation is simple:

value - lowest value

---

highest value - lowest value

Since you are working with four metrics, you have four calculations that need to be normalized. These calculations are Sales | Normalized, Margin | Normalized, % Margin | Normalized, and Units | Normalized.

// Sales | Normalized

(SUM([Sales]) - WINDOW_MIN(SUM([Sales])))

/

```
(WINDOW_MAX(SUM([Sales])) - WINDOW_MIN(SUM([Sales])))

// Units | Normalized

(SUM([Quantity]) - WINDOW_MIN(SUM([Quantity])))

/

(WINDOW_MAX(SUM([Quantity])) -
WINDOW_MIN(SUM([Quantity])))

// Margin | Normalized

(SUM([Profit]) - WINDOW_MIN(SUM([Profit])))

/

(WINDOW_MAX(SUM([Profit])) - WINDOW_MIN(SUM([Profit])))

// % Margin | Normalized

((SUM([Profit])/SUM([Sales])) -
WINDOW_MIN(SUM([Profit])/SUM([Sales])))

/

(WINDOW_MAX(SUM([Profit])/SUM([Sales])) -
WINDOW_MIN(SUM([Profit])/SUM([Sales])))
```

You'll notice each calculation has a similar structure.

Step 2, Build the plot.

Step 2a: Detail. Place Subcategory on detail. Add Measure Names to columns and Measure Values rows. Add Sales | Normalized, Units | Normalized, Margin | Normalized, and % Margin | Normalized to the Measure Values card.

Step 2b: Mark type. Edit each table calculation so that the table calculations are computed across subcategory. Change the mark type to line. (See Figure 3-53.)

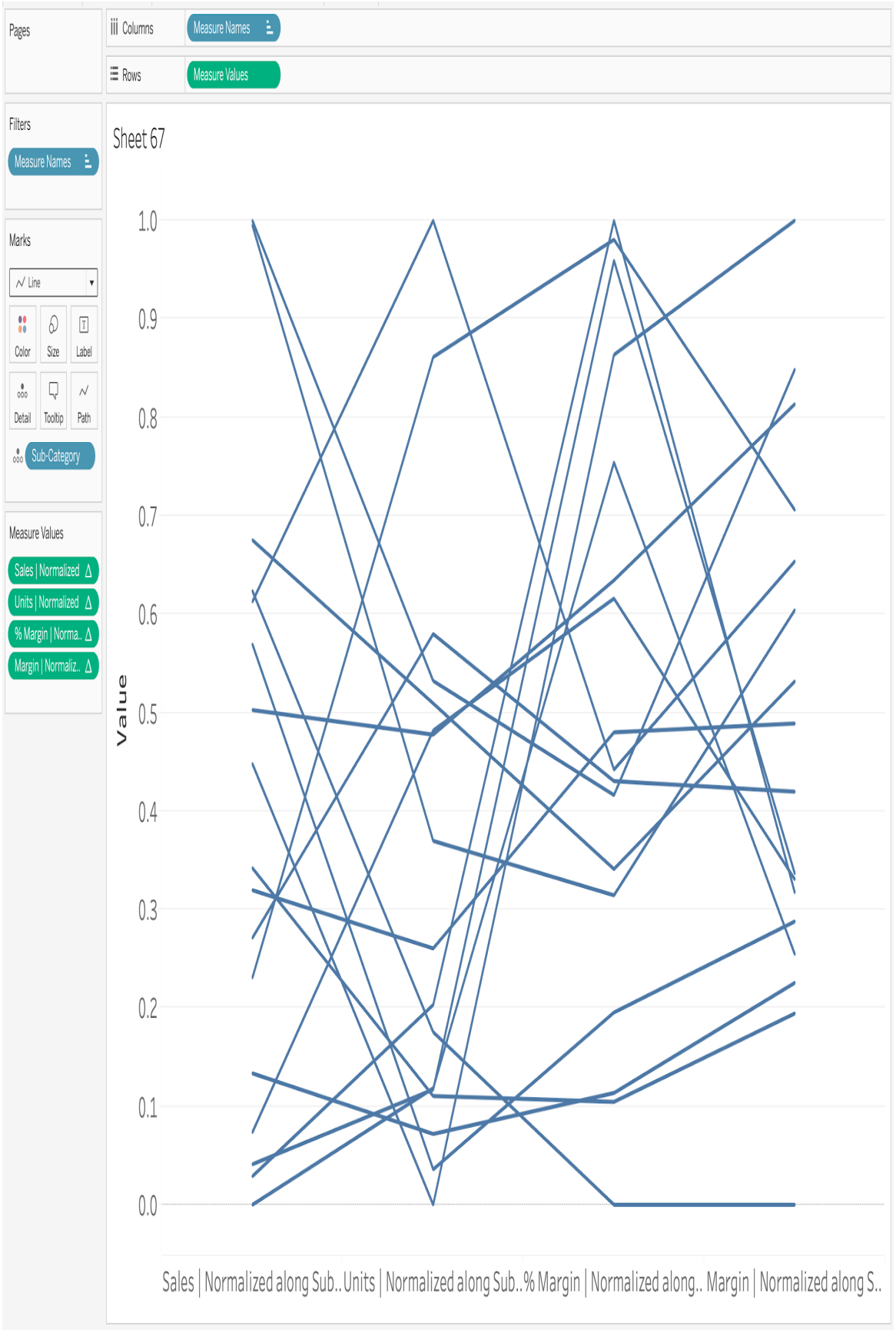

Pages
Columns
Measure Names
Rows
Measure Values
Filters
Measure Names
Marks
Line
Color
Size
Label
Detail
Tooltip
Path
Sub-Category
Measure Values
Sales | Normalized
Units | Normalized
% Margin | Norma..
Margin | Normaliz..
Sheet 67
Value
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Sales | Normalized along Sub..
Units | Normalized along Sub..
% Margin | Normalized along..
Margin | Normalized along S..

Figure 3-53. A sneak peek into the beginning of the parallel coordinates plot.

Step 2c: Rows. Add a second Measure Values to rows. Edit the left-most Measure Values marks card and move Subcategory from detail to path.

Step 2d: Size. Change the size of the lines to be narrower and change the color to be a medium gray.

Step 2e: Axes. Create a synchronized dual axis chart. Format and remove all lines and borders. Then hide the headers (Figure 3-54).

Pages

Columns: Measure Names

Rows: Measure Values | Measure Values

Filters

Measure Names

Marks

All

Measure Valu...

Measure Val...

Line

Color | Size | Label

Detail | Tooltip | Path

Sub-Category

Measure Values

Sales | Normalized Δ

Units | Normalized Δ

% Margin | Norma.. Δ

Margin | Normaliz.. Δ

Parallel Coordinates

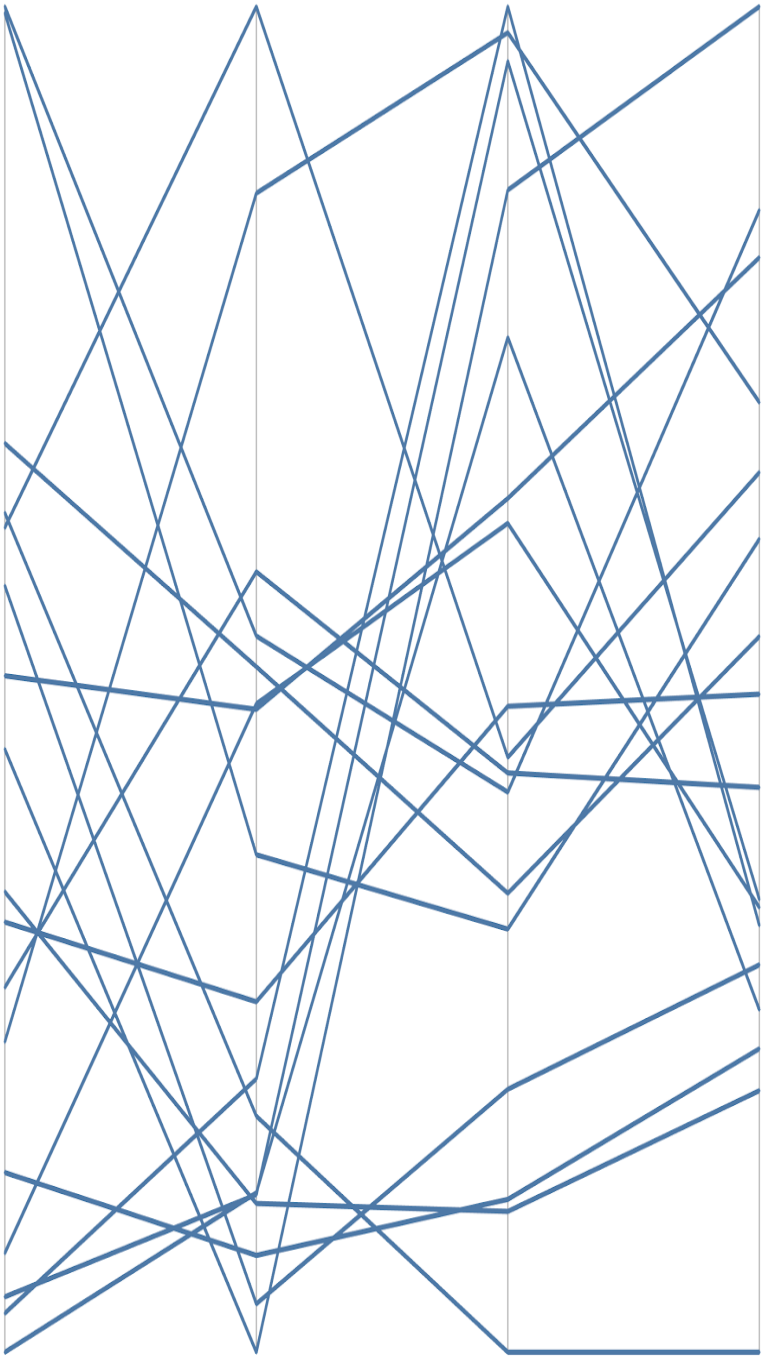


Figure 3-54. Adding the vertical lines is useful for our audience as it delineates where change occurs.

Step 3: highlight an individual member of the subcategory dimension.

Step 3a: Create parameter. Create a parameter from subcategory and call the new parameter Subcategory Parameter.

Step 3b: Create calculation. Create a new calculation called Subcategory | TF:

// Subcategory | TF

[Subcategory] = [Subcategory Parameter]

Step 3c: Marks card. On the right-most Measure Values marks card, add Subcategory | TF to size and color. On the left-most Measure Values marks card add Subcategory | TF to detail.

Step 3d: Table calculations. You will need to edit your table calculations again to show the proper values with color and size. When you edit your table calculations in the measure values pane be sure to compute using both Subcategory and Subcategory | TF.

Step 3e: Color. Set a color to the True and False values. Set the True value to have greater luminosity relative to the false values. For this example we've set the values to #7C00B2 for True and #CEB8D8 to False. Make sure the true value is layered on top of the false values. Make the true values slightly wider than the false values.

Step 3f: Tooltips. Add the actual values to the tooltips of the second marks card and format the tooltip. Remove the tooltip from the first marks card by deleting all the values inside the tooltip prompt.

Step 3g: Axes. Edit the axis so that the values run from exactly 0 to 1.

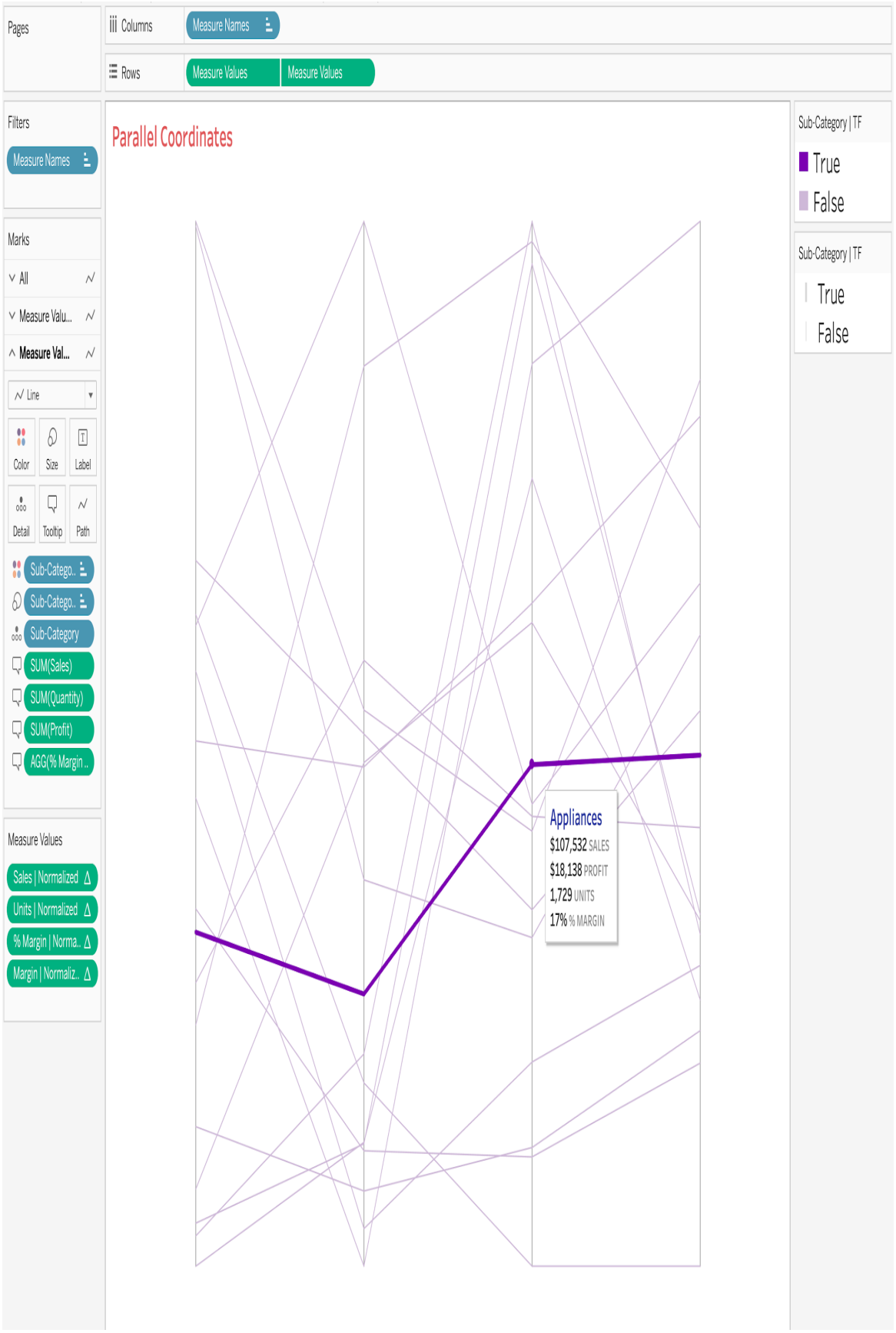

Pages
Columns
Measure Names
Rows
Measure Values
Measure Values
Filters
Measure Names
Marks
All
Measure Valu...
Measure Val...
Line
Color
Size
Label
Detail
Tooltip
Path
Sub-Catego..
Sub-Catego..
Sub-Category
SUM(Sales)
SUM(Quantity)
SUM(Profit)
AGG(% Margin ..
Measure Values
Sales | Normalized
Units | Normalized
% Margin | Norma..
Margin | Normaliz..
Parallel Coordinates
Appliances
$107,532 SALES
$18,138 PROFIT
1,729 UNITS
17% % MARGIN
Sub-Category | TF
True
False
Sub-Category | TF
True
False

Figure 3-55. A formatted parallel coordinates plot—but you just can't add labels to the top and bottom of the chart. Instead, you'll create separate sheets.

The core work required for this example is done.

If we want simple labels for this chart, we could edit the left-most Measure Values marks card and add Measure Names to label, then show the values on the minimum and maximum values. However, we're going to show you a multi-sheet approach that will give more context to the data.

Step 4: Create top sheet

At the top of each axis, you want to display the maximum value. This will give context to your audience. To do this you actually need to create a separate sheet.

Step 4a: Columns. Start by creating four identical ad-hoc calculations on columns. Double-click and type MIN(0.0). Do this again 3 more times. This will create 4 separate marks cards for us to place labels in each (Figure 3-56).

Figure 3-56. Add MIN(0.0) as many times as necessary to build the top labels of the parallel coordinates.

Step 4b: Mark type. Change all of the marks cards to text marks. On each marks card you are going to add a label for the measure type and the maximum value on the axis.

Step 4c: Marks card #1. On the first marks card create a new calculation called Sales | Window Max and write

// Sales | Window Max

WINDOW_MAX(SUM([Sales]))

Add this calculation to text. Edit the text. Write Sales on the first line and place the calculation on the second line (Figure 3-57).

Figure 3-57. Add text describing the dimension and the maximum value in the window.

Step 4d: Marks card #2-#4. Repeat this task on the second MIN(0.0) marks card for Units | Window Max. And Margin | Window Max and % Margin | Window Max on the 3rd and 4th marks cards, respectively.

// Units | Window Max

WINDOW_MAX(SUM([Quantity]))

// Margin | Window Max

WINDOW_MAX(SUM([Profit]))

// % Margin | Window Max

WINDOW_MAX(SUM([Profit])/SUM([Sales]))

Step 4e: Detail. Add Subcategory to detail of each of the marks cards. Then remove all lines and borders. Finally, hide the headers. This will leave you with the messy visualization in Figure 3-58.

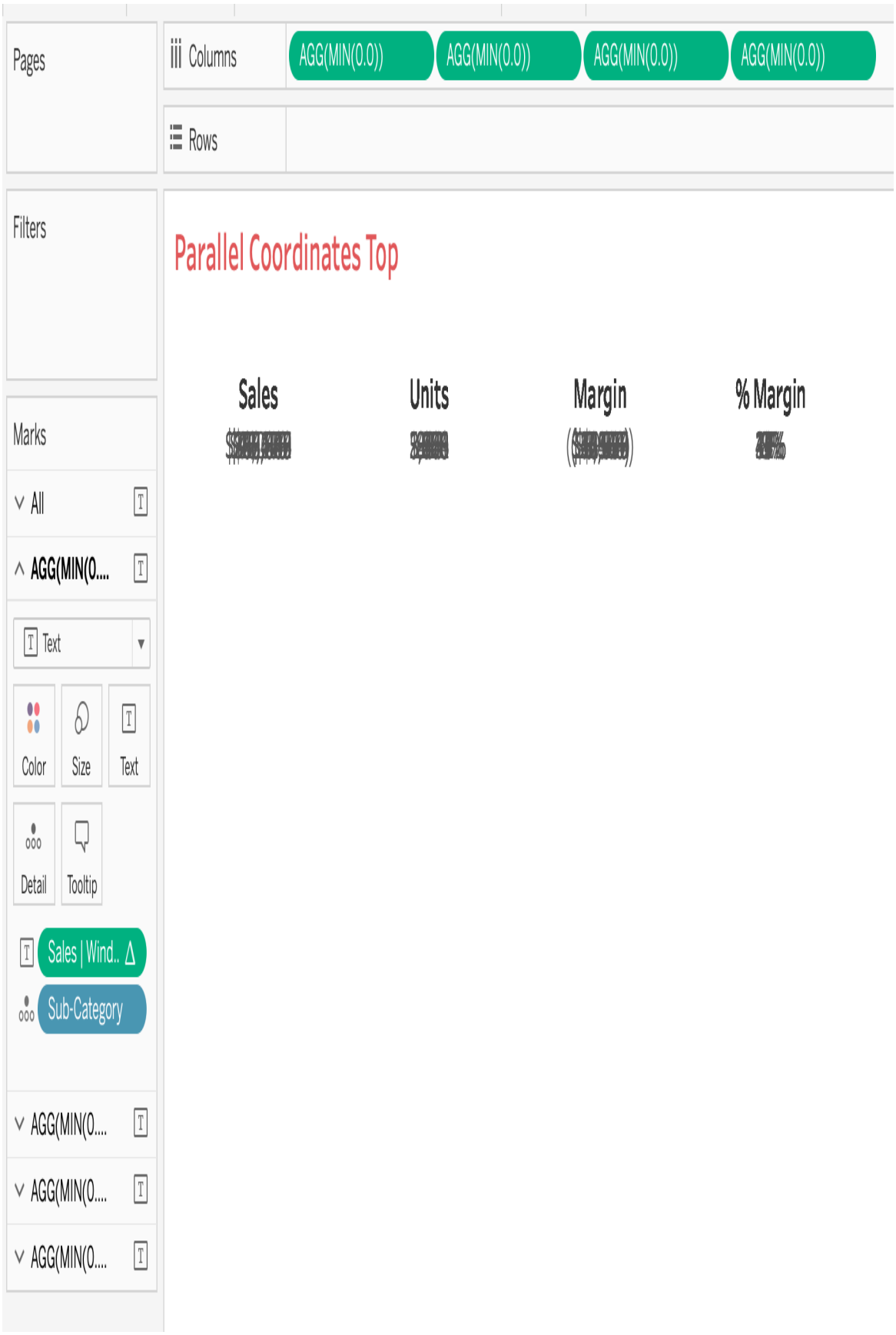

Pages
Columns
AGG(MIN(0.0))
AGG(MIN(0.0))
AGG(MIN(0.0))
AGG(MIN(0.0))
Rows
Filters
Parallel Coordinates Top
Sales
Units
Margin
% Margin
Marks
All
AGG(MIN(0....
Text
Color
Size
Text
Detail
Tooltip
Sales | Wind.. Δ
Sub-Category
AGG(MIN(0....
AGG(MIN(0....
AGG(MIN(0....

Figure 3-58. A sneak peek at the work-in-progress for the top labels before adding a filter to show a single value.

Step 4f: Filter. To clean this up, you'll need to create and add a single calculation to your filters. Create a calculation called First, where

// First

FIRST() = 0

Then place the calculation on filters. Don't worry about selecting True or False right away. You'll need to edit your table calculation before anything works anyway.

Step 4g: Table calculation on filters. Set your table calculation to compute across subcategory and then edit your filter for when values are True. Finally, turn off the tooltips.

Now you'll have a header that's worth sharing (Figure 3-59)!


Pages
Columns
AGG(MIN(0.0))
AGG(MIN(0.0))
AGG(MIN(0.0))
AGG(MIN(0.0))
Rows
Filters
First: True
Parallel Coordinates Top
Sales
$167,380
Units
2,976
Margin
$41,937
% Margin
25%
Marks
All
AGG(MIN(0....
Text
Color
Size
Text
Detail
Tooltip
Sales | Wind..
Sub-Category
AGG(MIN(0....
AGG(MIN(0....
AGG(MIN(0....

Figure 3-59. A sheet that will be used as a header for the parallel coordinates.

Step 5: Create bottom sheet

The last sheet we need to create is a sheet that will display the lowest values for each of the measures and will look like its labelling the bottom axis.

Step 5a: Columns. Repeat the same process from the previous step, adding for custom MIN(0.0) calculations to columns.

Step 5b: Details Marks Cards #1-#4. On the first marks card you'll need to calculate the window minimum for sales, then the window minimum for units, then the minimum for margin and percent of margin. Each of these will be placed on text.

// Units | Window Min

WINDOW_MIN(SUM([Sales]))

// Units | Window Min

WINDOW_MIN(SUM([Quantity]))

// Margin | Window Min

WINDOW_MIN(SUM([Profit]))

// % Margin | Window Min

WINDOW_MIN(SUM([Profit])/SUM([Sales]))

Step 5c: Detail. Add Subcategory to detail for each. Then format your sheet.

Step 5d: Filter. Add First as a filter, repeating the steps we discussed in the previous step.

The result is four values representing the lowest values possible on each of our axes in our parallel coordinates (Figure 3-60).


Pages
Columns
AGG(MIN(0.0))
AGG(MIN(0.0))
AGG(MIN(0.0))
AGG(MIN(0.0))
Rows
Filters
FIRST() = 0: True
Parallel Coordinates Bottom
$3,024
234
($17,725)
-9%
Marks
All
AGG(MIN(0....
Text
Color
Size
Text
Detail
Tooltip
Sales | Wind..
Sub-Category
AGG(MIN(0....
AGG(MIN(0....
AGG(MIN(0....

Figure 3-60. A sheet that will be used as a footer for the parallel coordinates.

Step 6: Add to dashboard

Now that you have three sheets with the key data, you need to stitch these together on a dashboard. The best way to do this is by using a vertical container on a dashboard.

Ideally this visualization would be paired with a table (yes, a table) or a series of bar charts that show the individual values, but for the sake of this blueprint we are going to have a dashboard with a single visualization.

Step 6a: Size. Create a new dashboard and set the size to be 800px by 500px.

Step 6b: Containers. Add a tiled vertical container.

Step 6c: Add visuals. In the container, add the top of the parallel coordinates, then the parallel coordinates themselves, and then the footer. For each of the sheet make sure they use the entire view. Also hide the titles.

Step 6d: Remove legends. Neither of the legends provides a ton of value, so remove those from the dashboard.

Step 6e: Formatting. Adjust the spacing for the header and the footer. I recommend fixing the height for the header to 60 and the footer to 30.

Step 6f: View Parameter. Add the parameter for subcategory onto the dashboard.

The result is the very clean-looking parallel coordinates plot in Figure 3-61.

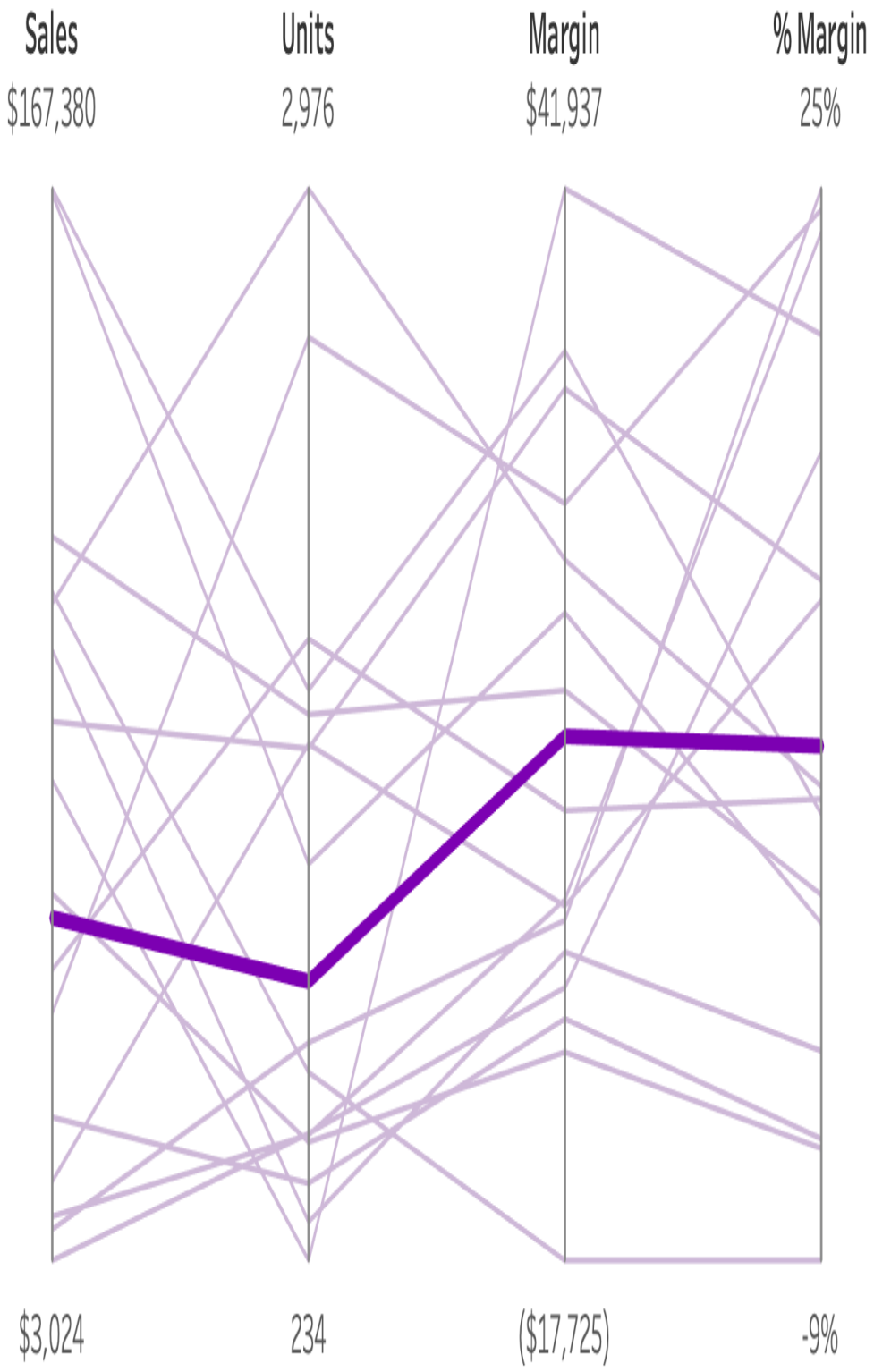

Sales
Units
Margin
% Margin
$167,380
2,976
$41,937
25%
$3,024
234
($17,725)
-9%
Appliances

Figure 3-61. : A single dashboard displaying the header, parallel coordinates, and footer in a single dashboard.

As mentioned earlier, a parallel coordinates plot is designed to summarize how a specific group performs across multiple measurements relative to other groups.

It does not provide specifics of the relationship between two or more metrics like a scatterplot. It also does not describe the performance of other groups on those same metrics. The real purpose is to examine how one group performs across these multiple measurements.

In our example, if we looked at Appliances, we'd probably note how average of a category this is. It doesn't stand out on any particular metric! If we looked at a different subcategory, copiers, we'd see a different pattern. Copiers, like appliances, are average for sales (Figure 3-62).

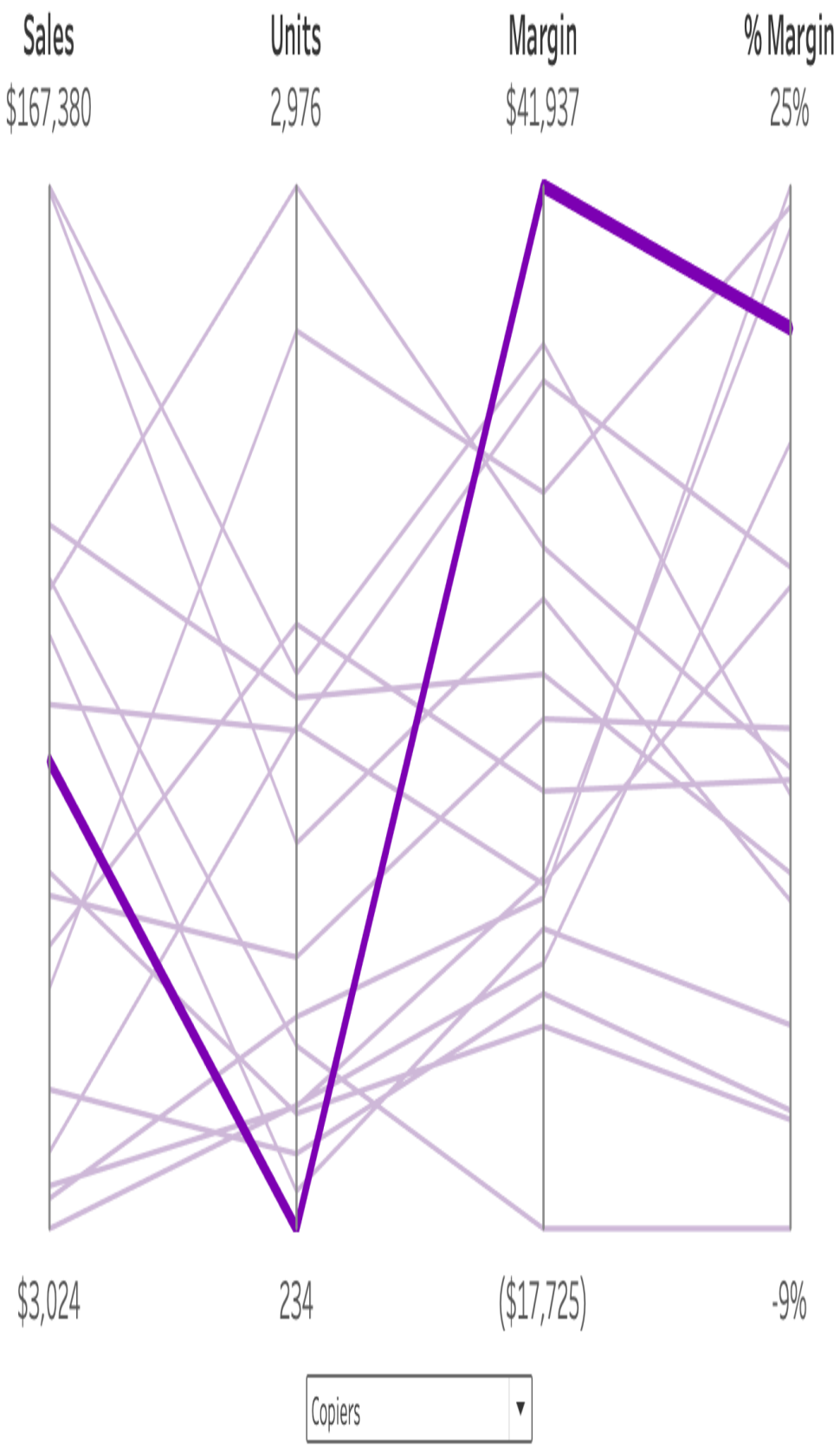

Sales
Units
Margin
% Margin
$167,380
2,976
$41,937
25%
$3,024
234
($17,725)
-9%
Copiers

Figure 3-62. An updated view of the parallel coordinates with the Copiers subcategory selected.

But that's where things begin to differ. Copiers are the lowest for total units, but they produce the highest total margin and a very high percent of margin. The result is a plot that concisely tells two different stories for the two subcategories. While there is a lot of effort that goes into creating a parallel coordinates plot, the payoff can be grand! If you're not sure you want a parallel coordinates plot, you can always use a bar chart.

Conclusion

In this chapter you explored different ways to compare data. For the most part, when we make comparisons with members of a single dimension across a single measure. For this type of comparison, we suggested you use a bar chart.

If your audience gets tired of bar charts, there are other alternatives that, while not technically bar charts, serve the same purpose, such as the lollipop chart and the Cleveland dot plot. The reality is that you don't even need to change the chart type to be something different, sometimes you just need to format your bar chart to look a little different.

In the next section, you explored how to calculate rankings without explicitly using the ranking calculation. Using INDEX() to sort is more reliable—particularly when it comes to rendering on server—and is a faster way to calculate unique rankings.

By using the index function for separate ranking calculations, you learned how to show the change in rank over time and display that change in rank either up or down.

You also learned how to create a bump chart for when you are interested in the order of multiple members over a time series. With the bump chart, you had to reverse the axis and use a level of detail calculation to identify the start and end of the lines to provide labels. You also focused on highlighting a single member from the bump chart to emphasize insight.

For the final section of the chapter, you tackled comparing multiple dimensions or multiple measures. Some comparisons are straightforward. With the barbell plot, you learned that you can use a dual axis chart and some formatting to really highlight differences across multiple measures.

You also learned about trellis or small multiple charts. Sometimes creating these is easy, when you have unique dimensions on rows and columns. Other times you might be faced with the challenge of creating a grid using just a single dimension. With this example, you learned that some simple arithmetic and well-designed calculations can create a grid with ease. You also learned that adding labels to a grid is not always as easy as it seems. But once again, we can use the strength of Tableau to determine the number of marks on a part of the view and provide dynamic calculations based on the detail on the marks card.

Finally, you tackled multidimensional data by creating a parallel coordinates plot. This required you to normalize your data. It also required a few extra sheets, to create values that look like they are on the ends of each of the axes of the parallel coordinate systems. With this example you conquered WINDOW_MAX(), WINDOW_MIN(), and FIRST(). The final result was a visualization that allows for a concise comparison of one group versus others on multiple dimensions.

Remember our last piece of advice from our parallel coordinates blueprint: bar charts would work for *all* of the examples in this chapter. Bar charts are extremely useful and versatile. However, when you are looking to communicate information in a more concise form, consider some of the alternative chart types discussed in this chapter.

In the next chapter, you'll utilize some of the chart types you've just learned; however, our focus will be on really understanding how you can get the most out of dates (and the calculations you can derive from dates). It will also focus on how you can automate insights, rather than updating filters and parameters as data are updated.

Chapter 4. Working with Time

> **A NOTE FOR EARLY RELEASE READERS**
>
> With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.
>
> This will be the 4th chapter of the final book. Please note that the GitHub repo will be made active later on.
>
> If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the authors at ann@jacksontwo.com and luke.stanke@tessellationconsulting.com.

When we were debating the content of this book, no chapter contained more examples than this chapter on time. We had so many examples that we practically added them in all chapters–including earlier chapters.

What makes time interesting is that a single time-related column can make a set of data extremely flexible to analyze. This is because time data is naturally hierarchical. Whether seconds, minutes, hours, days, weeks, months, or years, a column including time data gives you flexibility no other column of data will have.

In Chapter 3, you learned about categorical data, which is ideal for making comparisons. One underlying assumption about all comparisons is consistency. When we, as developers, visualize two non-time-based values, our audience assumes that we are making appropriate comparisons—in other words, that we're comparing data from the same time period.

Whether you're comparing year-to-date with the current year, or one month's performance with the same month a year ago, audiences assume

consistency with time periods–it’s practically an unspoken contract.

The goal of this chapter is to discuss the natural hierarchies of dates and datetimes and to help you develop calculations that can standardize and automate your visualizations. We’ll walk you through the foundational date calculations, then transition to blueprints to show you some specific challenges we’ve faced regularly in working with dates data. We’ll be putting that knowledge to work in the remainder of the book, too, with plenty of examples that utilize dates.

What you'll learn in this chapter

- How to navigate the hierarchies of date and time fields in Tableau, including using the DATEPART(), DATENAME(), and DATETRUNC() fields.
- How to plot points to the nearest hour and minute.
- How to plot points to the nearest 15 seconds and 15 minutes (or whatever time interval you choose).
- How to build effective heatmaps
- How to create continuous time calculations when the dates may be different.
- How to use table calculations to analyze sales data month-over-month and year-over-year.
- How to automatically filter data to the most recent 13 months using date calculations.
- How to work with non-standard fiscal calendars like mid-year fiscal starts, ISO-8601, and 4-5-4 retail calendars.

What You'll Need

For this chapter you will need two data sources: the Call Center dataset and the Sample – Superstore dataset. You will use a custom color palette.

You can add the custom color palette to your custom color palette by adding this chunk of code to the preferences.tps file in your My Tableau Repository:

```
<workbook>
            <preferences>
            <color-palette name="blackbody" type = "ordered-diverging">
            <color>#000000</color>
            <color>#8f0000</color>
            <color>#e63200</color>
            <color>#e6b900</color>
            <color>#eee154</color>
```

```
<color>#f9f4bf</color>
<color>#f0f6ff</color>
<color>#c8dfff</color>
<color>#a0c8ff</color>
</color-palette>
</preferences>
</workbook>
```

If you already have custom color palettes, then you can add everything from the <color-palette> code sections.

Understanding Dates and Time

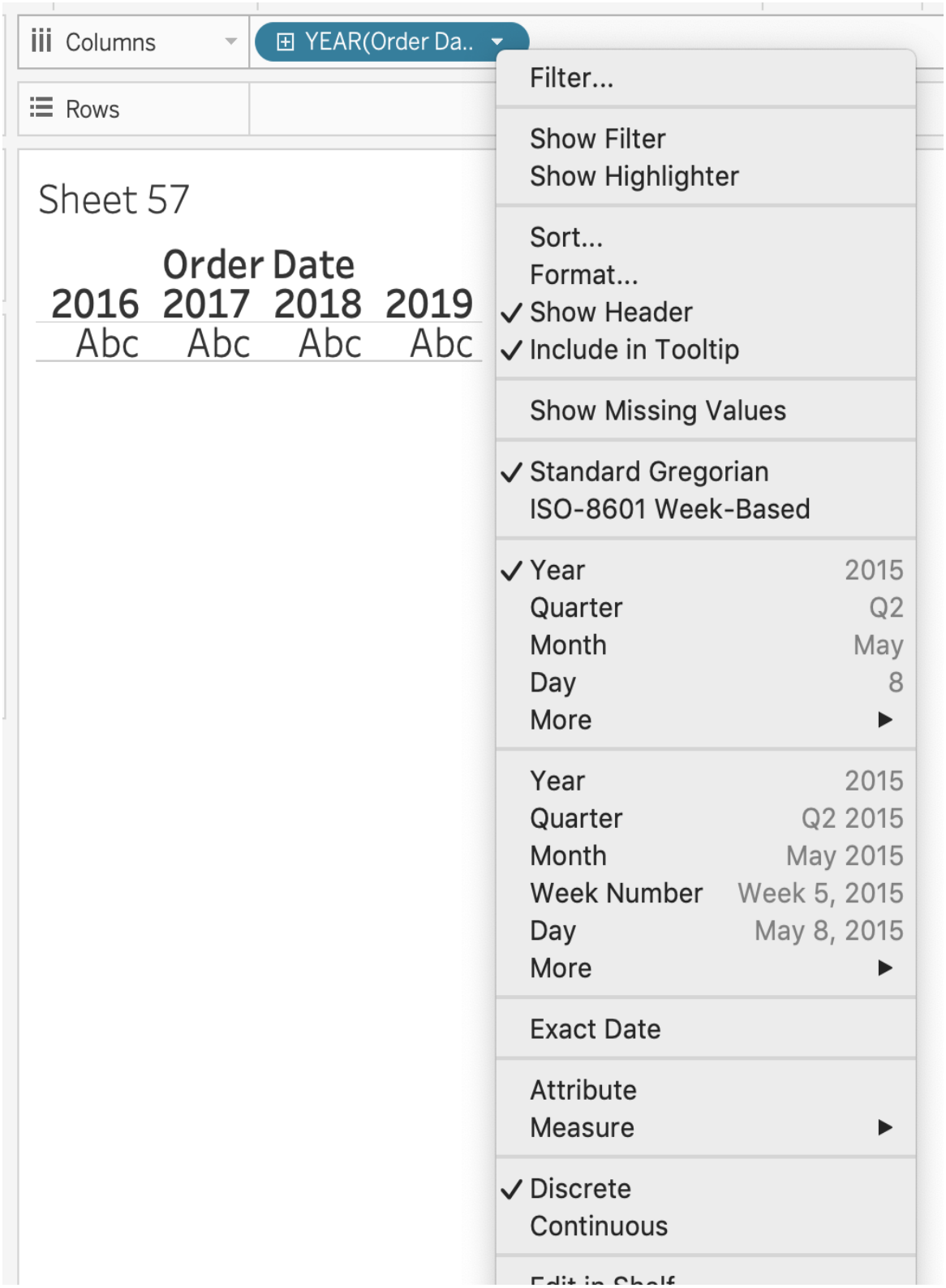
Tableau provides a very simple interface for working with time. By default, it creates a hierarchy that allows you to easily navigate any datetime possibility. If you drag any date or datetime field onto the visualization, the field will automatically appear aggregated to the nearest year. This is the default.

Tableau also places time values into a hierarchy. If you drill into that hierarchy, additional detail is added to your view by quarter, month, week, day, hour, minute, and second. This hierarchy will be available to your audience too. If you don’t want that functionality, you will need to create custom date calculations–which we highly recommend. (More on that later.)

While the default setting for a time field is a year, you can right-click and edit the year calculation and select the appropriate time.

If you do not want a year to be the default value, you can click and hold the ALT button (on a PC) or Option (on a Mac), then click and drag the time field onto the view. From there you can select the date part you want (Figure 4-1).

Figure 4-1. The view you will see after clicking-and-dragging a date onto a view and then right-clicking to change the date type.



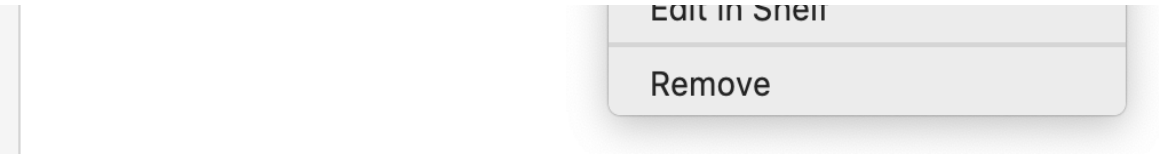


Figure 4-2. The view you will see after clicking and holding the Alt or Option button and dragging a date onto the view.

Date Parts and Date Values

Figure 4.1 shows many of the possible options of working for dates. You can choose date parts like year (2015), quarter (Q2) or month (May) and date values like quarter (Q2 2015), month (May 2015), or day (May 8, 2015). The difference between these is choosing a particular part of the date or truncating the date to the most recent date part.

Date Calculations

Tableau offers this same flexibility in calculated fields, through the DATEPART(), DATENAME(), and DATETRUNC() functions. DATEPART() returns a single numeric value for the part of the date of interest: if you specified the last digit of the year, the date May 8, 2015, would return a value of 5. DATENAME() returns a string for the part of the date of interest: May 8, 2015 would return May. It's a subtle difference, but how the information is displayed will change. Unlike DATEPART() and DATENAME(), DATETRUNC() returns an actual date or datetime value. With DATETRUNC(), values are rounded to the most recent specified date part.

Now, if the last two paragraphs sounded familiar, it's because we wanted it that way to show you this next point. While Tableau's fields on your view might say YEAR, QUARTER, or MONTH, Tableau is actually using calculations behind the scenes. Let's take a look at Figures 4-2, 4-3, and 4-4. You'll see we have the date part of month and date value of year on columns. If you double click on either of these values, Tableau will show the underlying calculations.

First, you have discrete month and continuous year on the columns shelf (Figure 4-2). These fields look nice and clean—easy to understand. But underneath each of these fields lie more complicated functions.

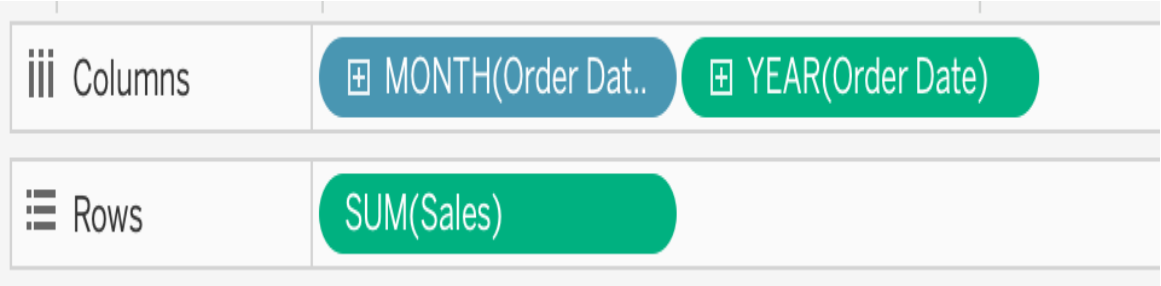


Figure 4-3. The rows and columns shelf showing discrete month and continuous year

If you double-click on the MONTH(Order Date) discrete field, this will open up the ad hoc calculation editor, which we'll use quite a bit in this book. Figure 4-3 shows the underlying calculation for MONTH(Order Date). It turns out this calculation actually uses the DATEPART() function and specifies 'month' as the first argument in the function. This helps return the month part of order date.

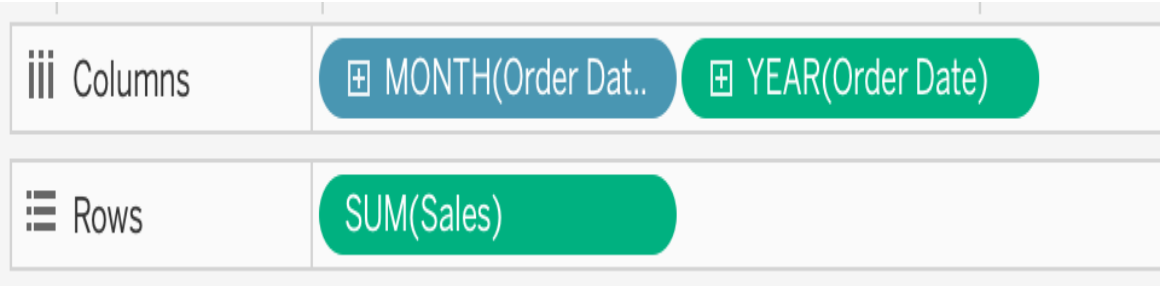


Figure 4-4. The DATEPART() function that makes up the infrastructure of the discrete MONTH(Order Date) field.

If you double-click into the continuous field of YEAR(Order Date), the underlying function is DATETRUNC(), which will, as previously mentioned, round a field down to the specified level. This is shown in Figure 4-4. In this case ‘year’ will return January 1 of the year in Order Date.

Figure 4-5. The DATETRUNC() field makes up the infrastructure of the continuous YEAR(Order Date) field.

What’s great about seeing these calculations is that it helps you learn date calculations inside Tableau while you’re using the default settings of any time field. You can change these calculations quickly, too—even in meetings, to share insights as conversation occurs.

Date Hierarchies and Custom Dates

Do you see that little “+” button to the left of the field names on your date fields on columns? That’s a hierarchy. By default, a date field automatically creates these hierarchies. When you add a date field to your view, your audience can interact with that date hierarchy. If you are looking to interact with a specific hierarchy of a date field, then working with dates can be a challenge.

Luckily, there are ways to circumvent the automated date hierarchies. You can do this by creating a custom date. You can do this by right-clicking on the original time field in the data source then choosing Create → Custom date (Figure 4-5). From there, select the date part or date value of interest. Once you’ve created this field, place it somewhere in the view—and you’ll notice the date has no hierarchy associated with it.

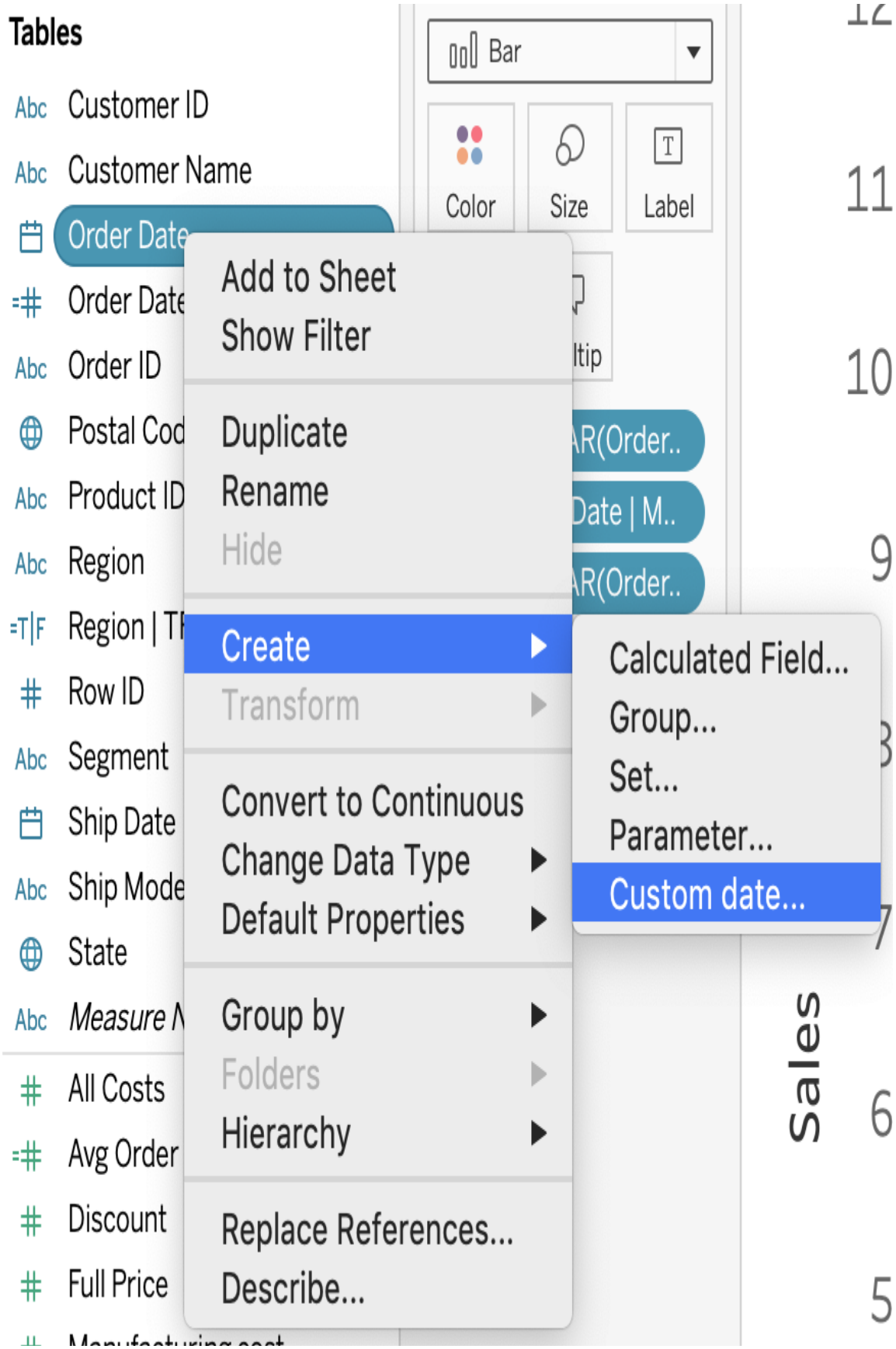

Tables
Abc Customer ID
Abc Customer Name
Order Date
Order Date
Abc Order ID
Postal Cod
Abc Product ID
Abc Region
Region | TH
Row ID
Abc Segment
Ship Date
Abc Ship Mode
State
Abc Measure N
All Costs
Avg Order
Discount
Full Price
Bar
Color
Size
Label
ltip
AR(Order..
Date | M..
AR(Order..
Add to Sheet
Show Filter
Duplicate
Rename
Hide
Create
Transform
Convert to Continuous
Change Data Type
Default Properties
Group by
Folders
Hierarchy
Replace References...
Describe...
Calculated Field...
Group...
Set...
Parameter...
Custom date...
12
11
10
9
8
7
6
5
Sales

Figure 4-6. The menu action for creating custom dates.

Discrete versus Continuous Dates

To better understand how dates are visually represented in Tableau, let’s look at four visualizations where we vary two components on the horizontal axis (Figure 4-6). We’ll start with the difference between date part and date value, then look at how discrete and continuous axes vary for the two chart types. This gives us four different options to explore.

Date Part, Discrete Axis

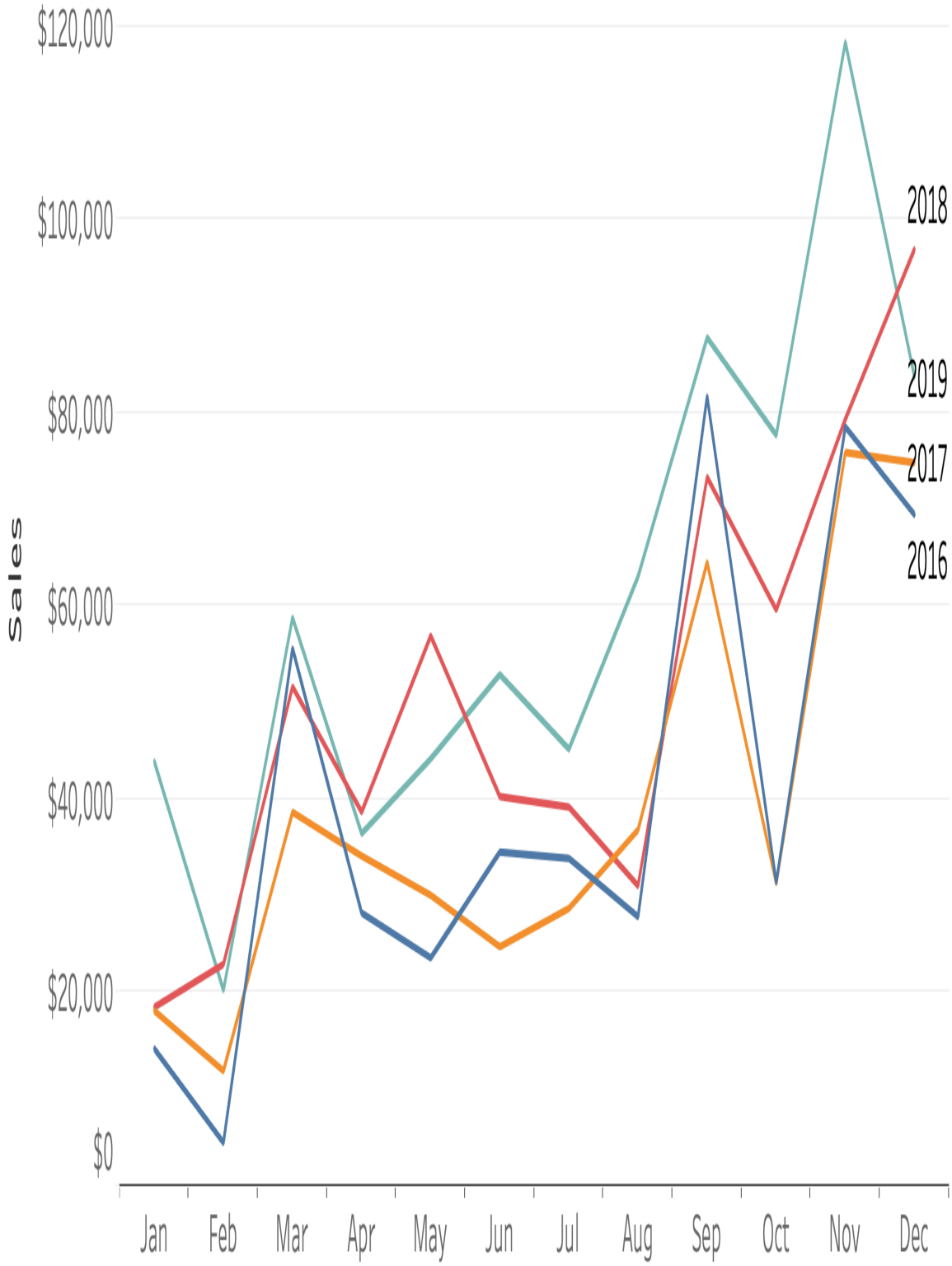


Figure 4-7. Discrete Date Part

Date Part, Continuous Axis

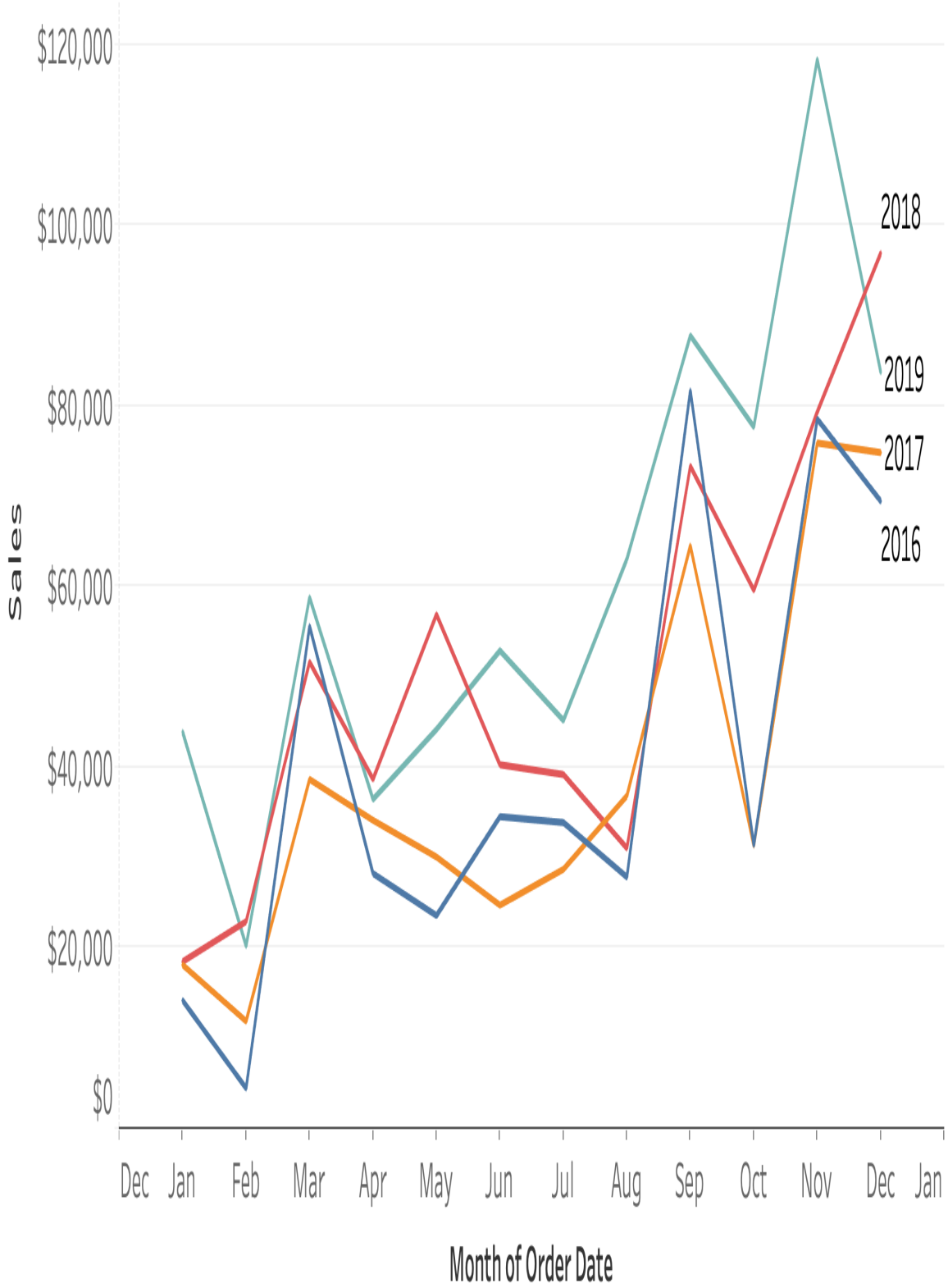


Figure 4-8. Continuous Date Part

Date Value, Discrete Axis

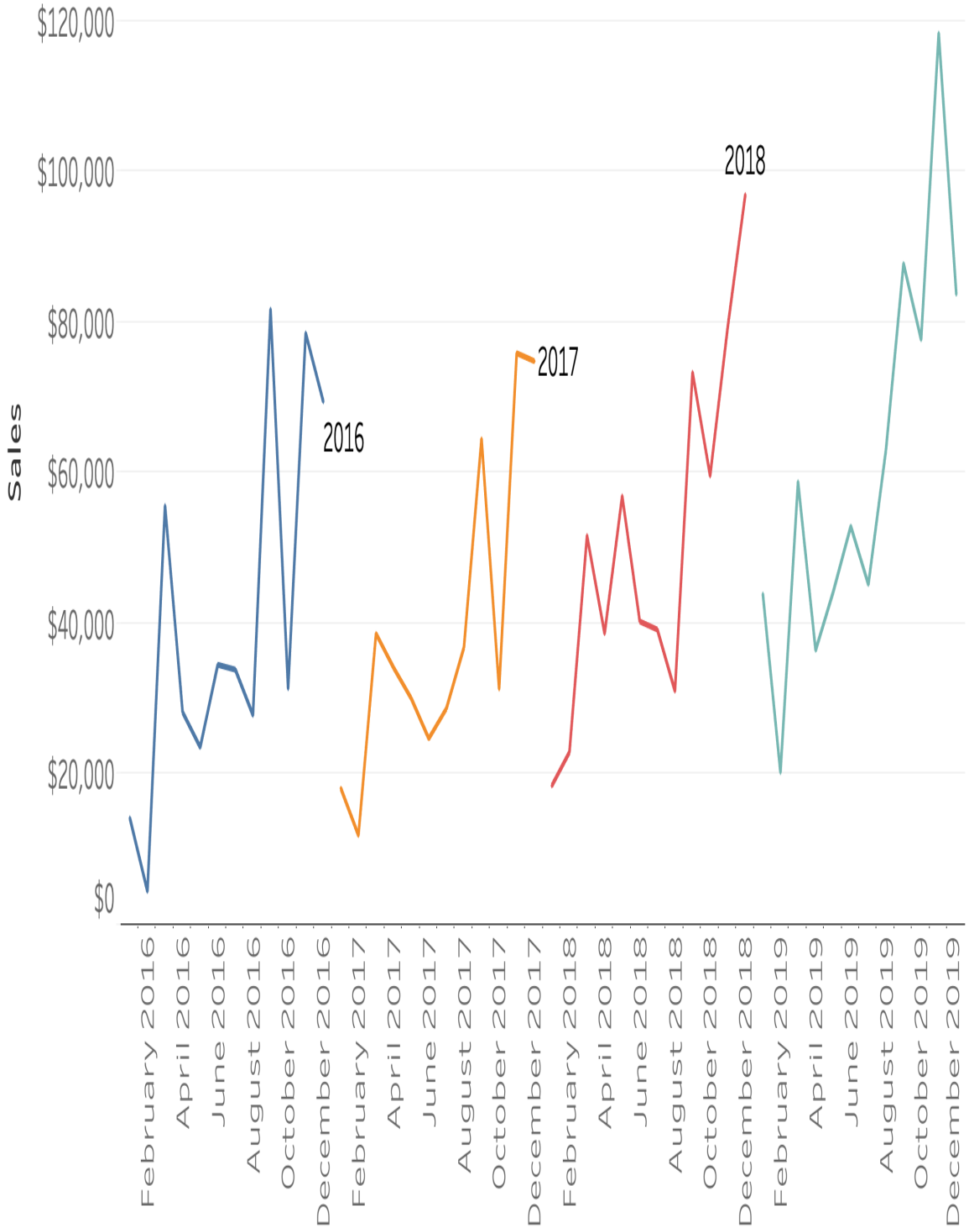


Figure 4-9. Discrete Date Value

Date Value, Continuous Axis

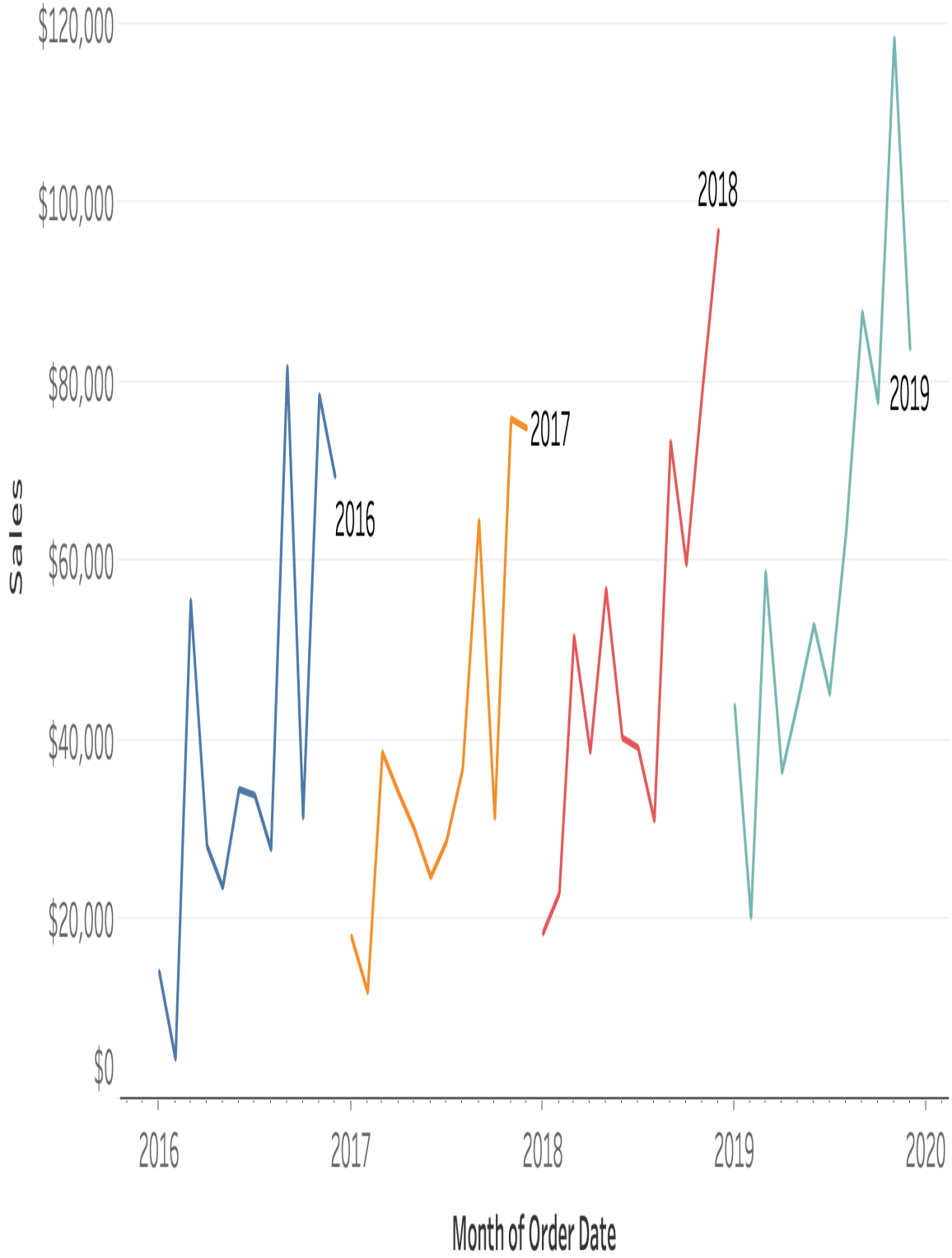


Figure 4-10. Continuous Date Value

Four charts showing the difference in the visual output depending on dates as discrete or continuous and date parts or date values.

The four charts in Figure 4-x show how changing just two options can yield four charts that operate quite differently. The two big differences that are worth reiterating: a date part selection returns only a single part of a date field. The default option for a date part is typically discrete. This creates “bins” that separate each date part. You can still convert discrete date parts to continuous, this makes the axis a single with no “bins”. The chart types will look similar but the axes function differently. If you look at the continuous date part visualization in the top-right, you will see the axis ticks for each month are centered on the month name. With discrete date parts in the top-left visualization, there are buckets for each month, and there are no ticks to align to the values for each month. These ticks make it easier for audiences to understand which months they are looking at; for this reason, we prefer working with continuous fields.

This extends to your date value options, too. If you choose a date value for month, it rounds each value down to the start of each month by default. This gives you month-by-year views of the data. By default, date values are continuous fields, but you can convert them to discrete. You’ll notice on the axis on the bottom-left of Figure 4-6 that the axes are distinct buckets of label names. The continuous date values on the bottom-right, however, show a single axis where ticks are used.

There is a place for discrete date axes. If you are planning on using a bar chart, you can use discrete date parts. If you are working with line charts, we recommend continuous axes.

Call Center Case Study: Call Frequency

Imagine you are doing call center analytics, comparing total calls. You want to know when your call volume peaks, so you can staff appropriately.

Working with time in Tableau is not without some challenges. It's easy to place hours and minutes on a chart in Tableau because if you are working with a datetime field it's automatically placed in Tableau datetime hierarchy. For instance, if you wanted to create a plot of total calls it wouldn't be that difficult.

With continuous date parts, it's fairly easy to convert a single date part into a continuous axis to plot multiple time periods. Using continuous date values truncates a date up to a certain date part. Sometimes you want to do the opposite, with a continuous axis for hour, minute, and day. For example, take a look at the plot in Figure 4-7, showing total phone inbound calls received from a call center every 15 minutes over two and a half years. For the first five blueprints in this chapter, you will convert a datetime at various levels of aggregation. To complete the final analysis, call data aggregated on a continuous axis by every 15 minutes (Figure 4-7), you'll need to get every date to be the exact same–but retain the time of each row in the data.

Total Calls (Every 15 minutes)

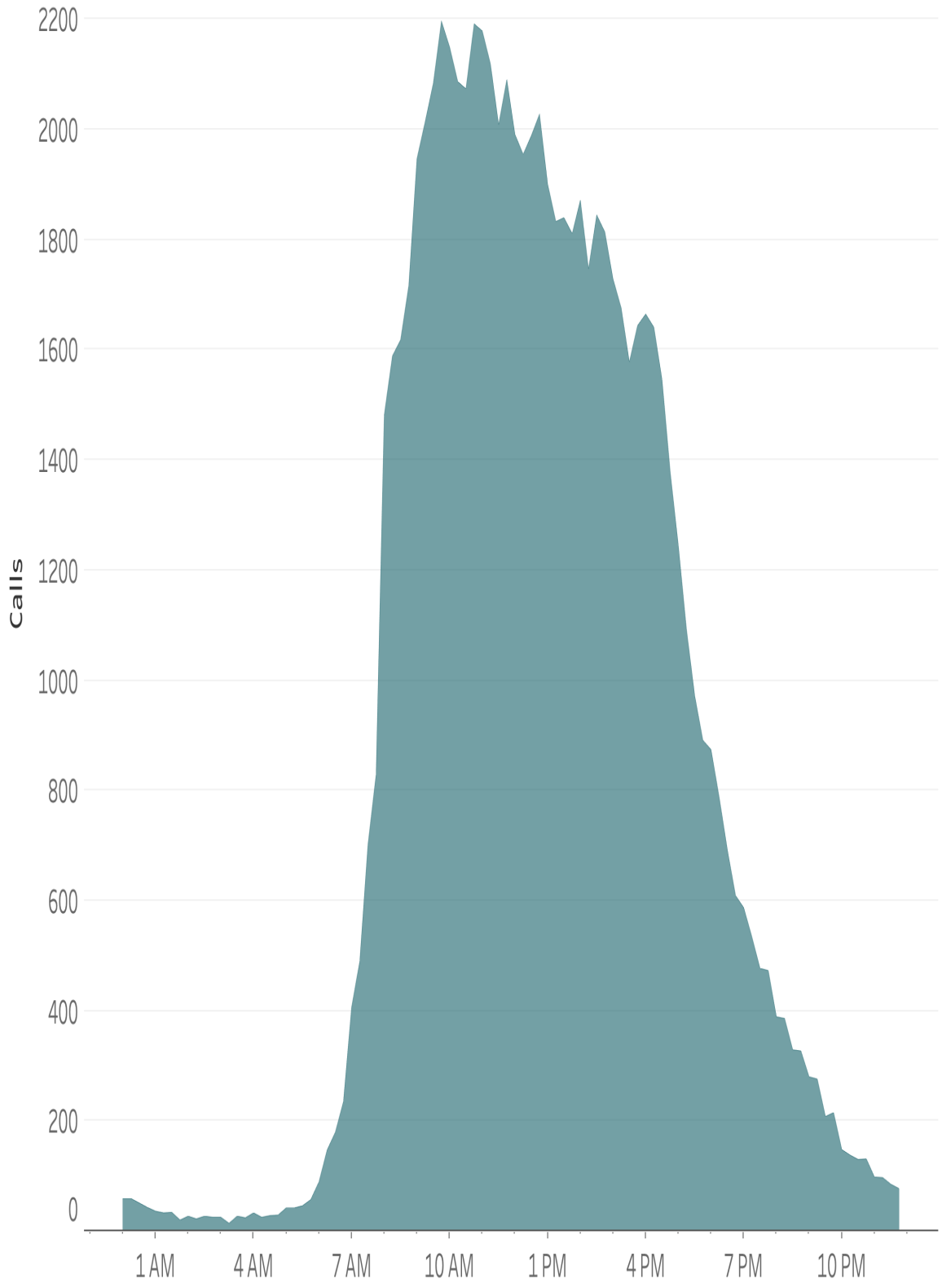
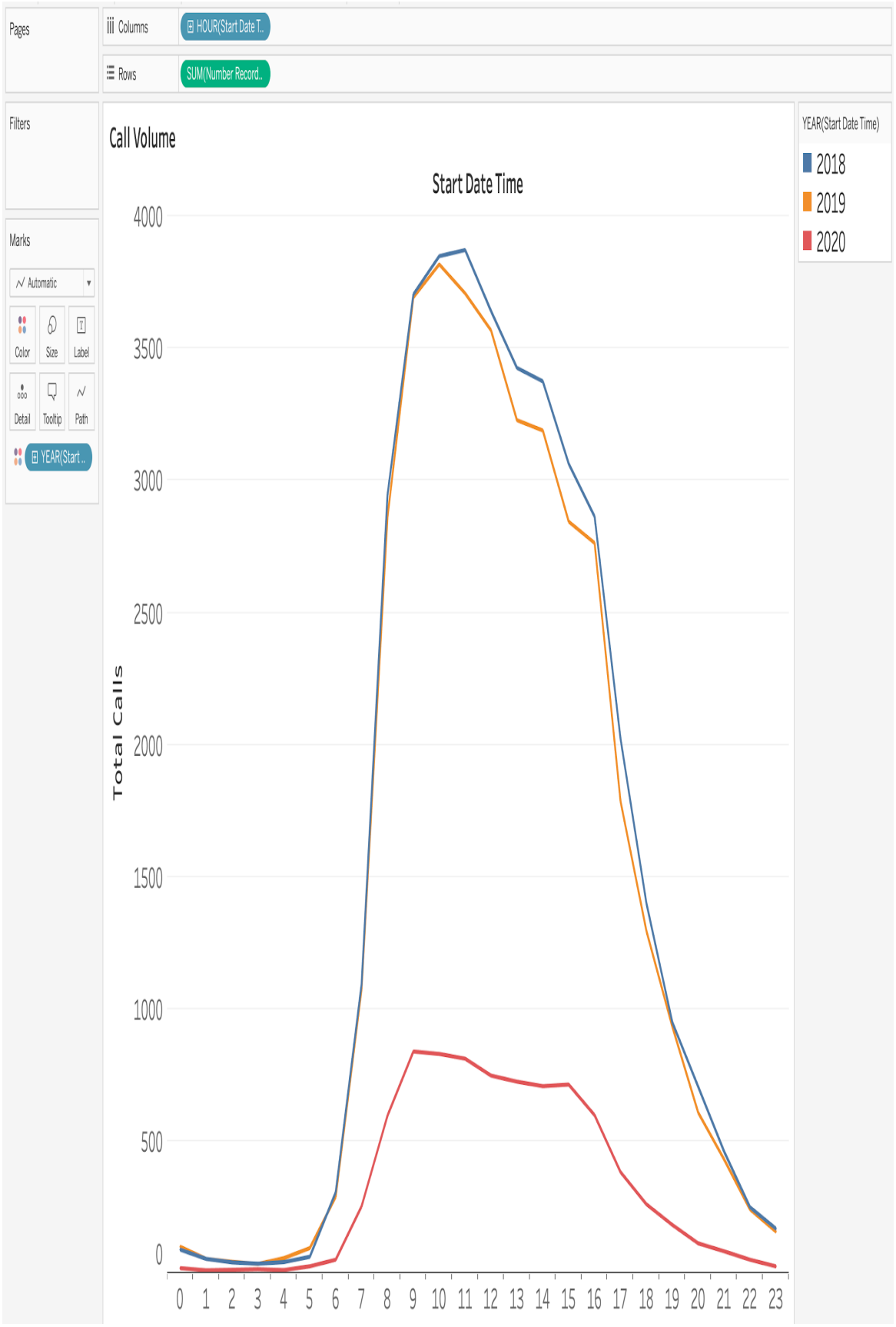


Figure 4-12. An area plot of total calls every 15 minutes.

Blueprint: Total call time by hour

In this blueprint we will tackle the challenge of working with time–particularly on a continuous axis. We'll start with a simple line plot by hour (Figure 4-8).


Pages
Columns
HOUR(Start Date T..
Rows
SUM(Number Record..
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Path
YEAR(Start ..
Call Volume
Start Date Time
Total Calls
4000
3500
3000
2500
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
YEAR(Start Date Time)
2018
2019
2020

A line plot of sales by hour for each of the three years of data.

Create a new sheet.

Place Start Date Time on columns. Then change the display to discrete hours.

Add number of records as a sum to rows.

Add the discrete year of Start Date Time to color.

In the plot in Figure 4-8, you can see that, regardless of year, the calls begin to escalate around 7 PM, but really pick up by the 8-o'clock hour. You also see that in 2020, calls are down across the board.

What's missing from this analysis is a deeper investigation. Do calls occur most at the beginning of each hour or do they peak at mid-hour? If you are creating a staffing plan, having the exact times might be more useful.

Blueprint: Creating a plot to measure total call time by minute

Step 1. Sort by minute.

Take the visualization from the last blueprint and click the "+" on the HOUR hierarchy (Figure 4-14).

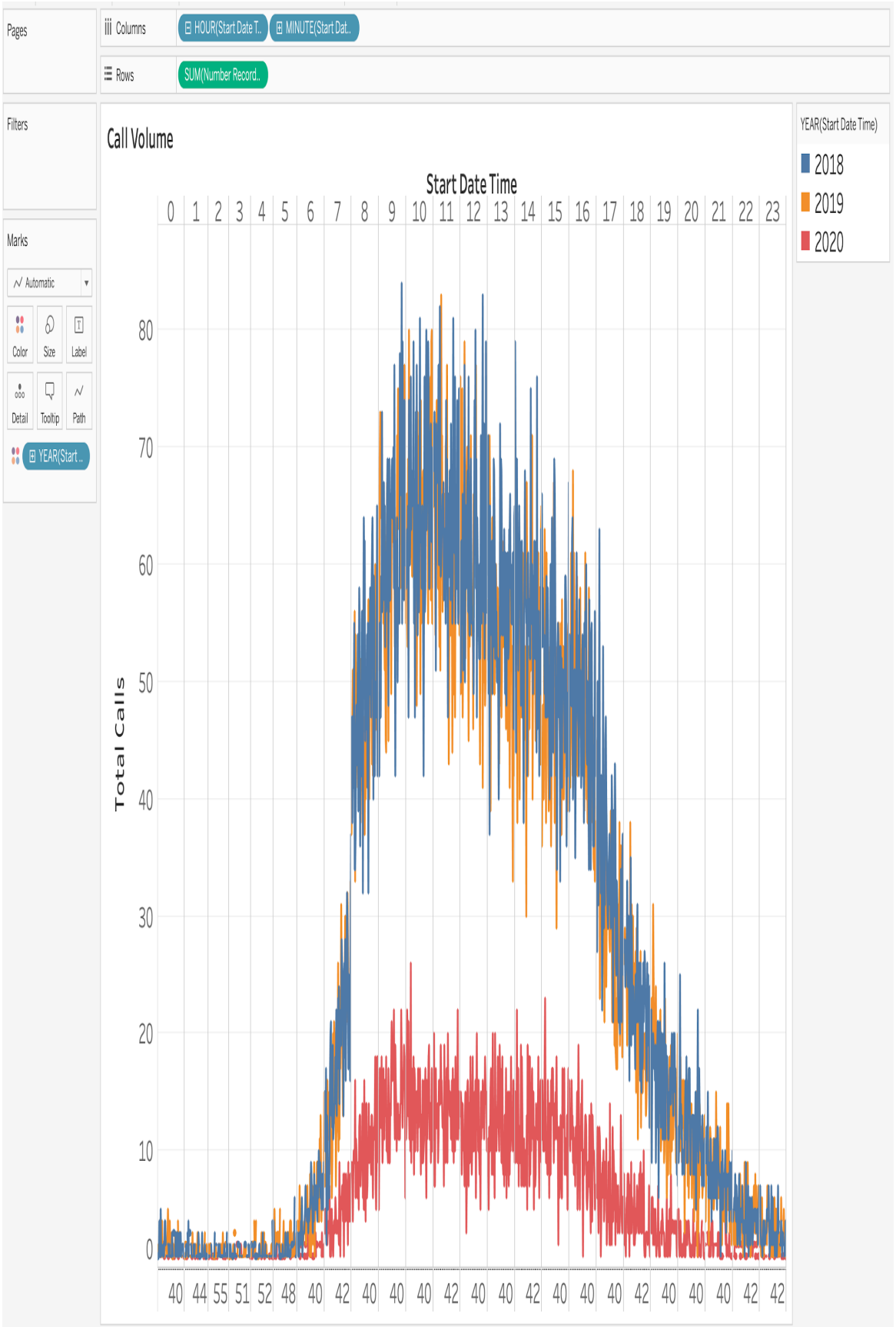

Pages
Columns
HOUR(Start Date T..
MINUTE(Start Dat..
Rows
SUM(Number Record..
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Path
YEAR(Start ..
Call Volume
Start Date Time
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Total Calls
80
70
60
50
40
30
20
10
0
40 44 55 51 52 48 40 42 40 40 40 42 40 40 42 40 40 40 42 40 40 40 42 42
YEAR(Start Date Time)
2018
2019
2020

Figure 4-14. : A line plot of calls by hour and minute for each of three years of data.

The result shows calls by minute in the day, but we've now got two different discrete axes: the top axis partitioning by hour, and a second axis for minute, creating up to 60 individual partitions within each hour. This will only create partitions where data exists: if there is no data for hour 0, minute 53 (which there are not), then the partition doesn't exist.

Step 2. Add any missing partitions.

If you want to include missing partitions of any specified date part, you need to right-click on the date part and select show missing values (Figure 4-10).

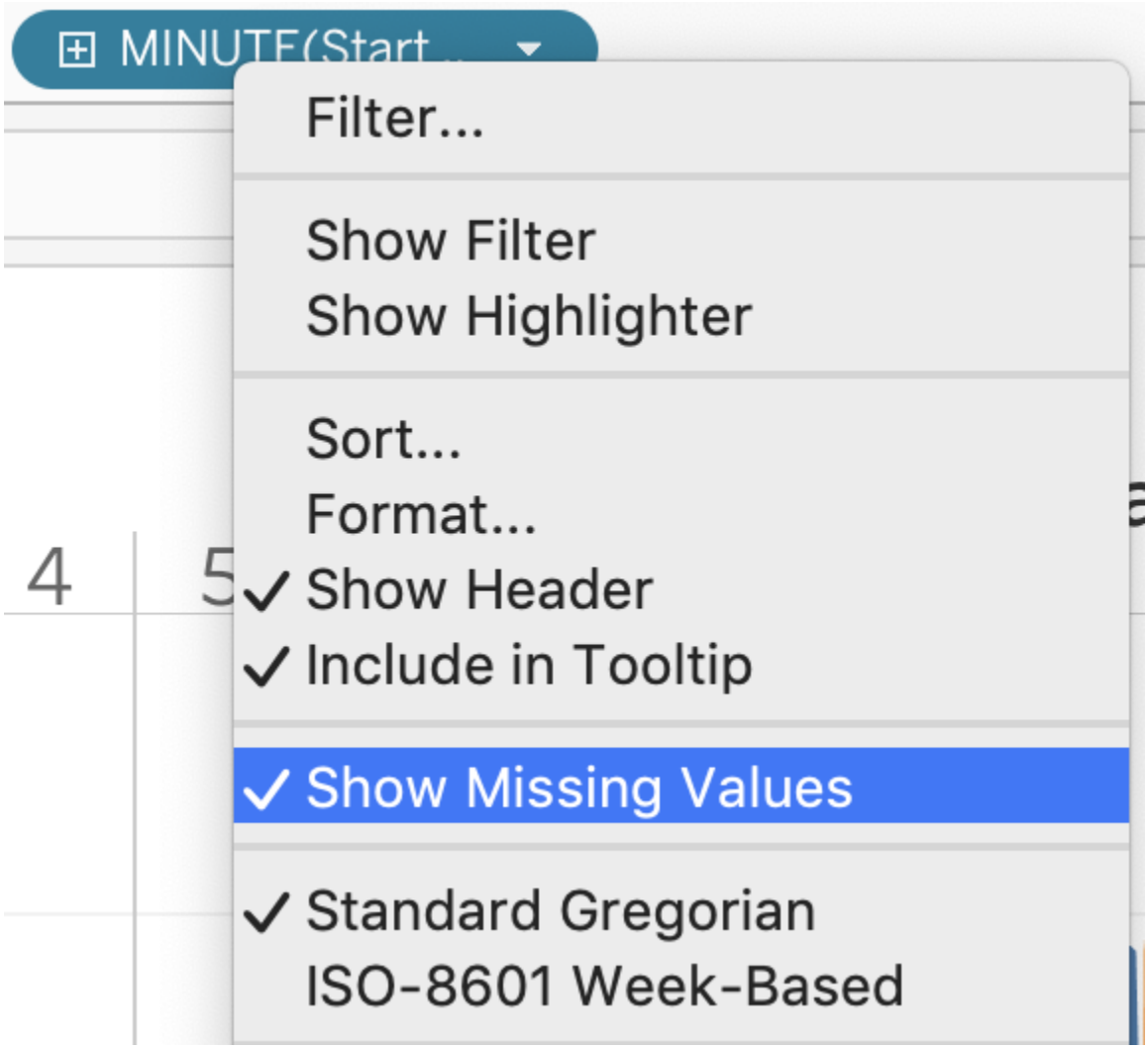


Figure 4-10: To show missing values, right-click on the date field and select Show Missing Values.

The visualization in Figure 4-14 has two sets of discrete “bins”: one for hours and one for minutes. Using discrete values for hours and minutes is difficult because there are 1,440 partitions (24 hours x 60 minutes). So how can you make them a single axis?

Continuous Date-Time Axes

For this calculation we are going to rely on two very commonly used calculations: DATEADD() and DATEDIFF().

DATEADD() adds or subtracts dates or time and requires three inputs:

- a specified date part, written inside of quotations and spelled out in lowercase. This will show the units we are adding to a date, whether it’s seconds or hours or years.
- Any integer indicating the amount of time we want to add or subtract. If a negative number is specified then time will be subtracted from the value.
- The initial datetime field.

This calculation is extremely versatile; we use it all the time.

DATEDIFF() calculates the difference between two dates based on the date part of interest and requires three inputs:

- A specified date part.
- A starting date.
- An ending date.

There are many dateparts you can specify: ‘year’, ‘quarter’, ‘month’, ‘dayofyear’, ‘day’, ‘weekday’, ‘week’, ‘hour’, ‘minute’, ‘second’, ‘iso-year’, ‘iso-quarter’, ‘iso-week’, and ‘iso-weekday’. (For more technical help, read Tableau’s documentation on Date Functions.)

You’ll now create a calculation that allows you to have a single axis for time.

Blueprint: Continuous date-time axis by the second

Step 1. Create your time field.

Create a calculated field called time and write the following calculation:

// time

DATEADD(

“day”,

DATEDIFF(“day”, [Start Date Time], {MAX(DATETRUNC(“day”, [Start Date Time]))}),

[Start Date Time]

)

This calculation will change all dates in your dataset to be equal to the maximum date in your dataset. The time–hour, minutes, and seconds–will remain the same.

Step 2. Create a continuous axis.

Remove all calculations on the columns.

Add [time] as an exact date to the columns.

This will produce the visualization in Figure 4-11:

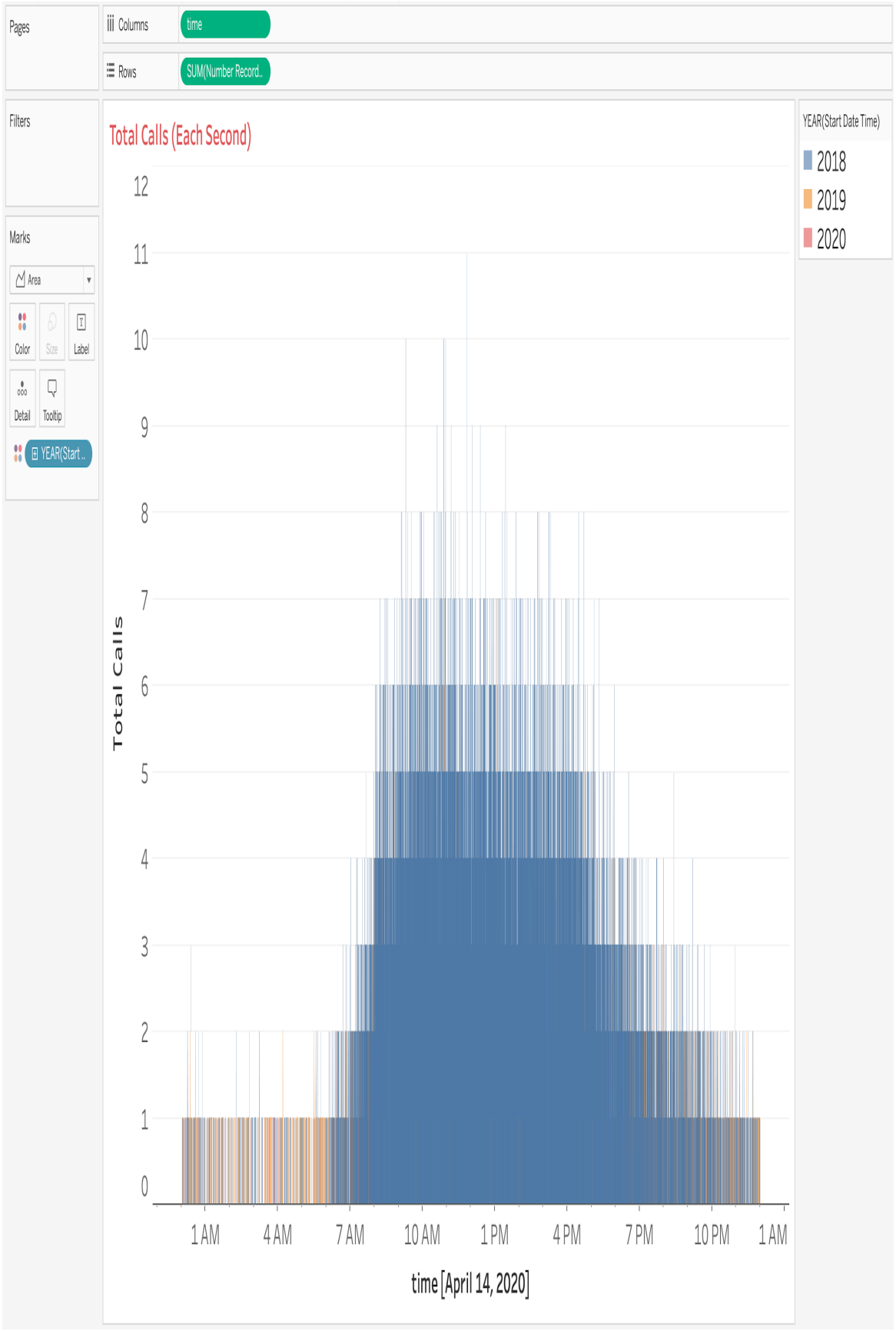

Pages
Columns
time
Rows
SUM(Number Record..
Filters
Marks
Area
Color
Size
Label
Detail
Tooltip
YEAR(Start..
Total Calls (Each Second)
YEAR(Start Date Time)
2018
2019
2020
Total Calls
0
1
2
3
4
5
6
7
8
9
10
11
12
1 AM
4 AM
7 AM
10 AM
1 PM
4 PM
7 PM
10 PM
1 AM
time [April 14, 2020]

Figure 4-15. Total calls per second of the day using a continuous axis.

You now have a continuous axis. However, the analysis here is to the second, which isn't extremely helpful or insightful. Instead of a per-second analysis, maybe you want to do every 15 seconds.

Blueprint: Continuous date-time axis for 15-second intervals

Use the visualization in Blueprint 4-3.

Create a new calculation and call it time / 15 sec.

Write the following:

// time / 15 sec

DATEADD(

"second",

-(DATEPART("second ",[time]) % 15),

[time]

)

If you click-drag-and-replace time with time / 15 sec as a continuous axis it produces the visualization in Figure 4-16.

Here you're first calculating the second for the time field. You then use the modulo symbol (%) to calculate the total seconds every 15 second. This means that, instead of counting to 60, you are instead counting to 14; instead of continuing to 15, you restart back at zero.

The result of this calculation is a datetime truncated to the most recent 15 seconds (Figure 4-16).

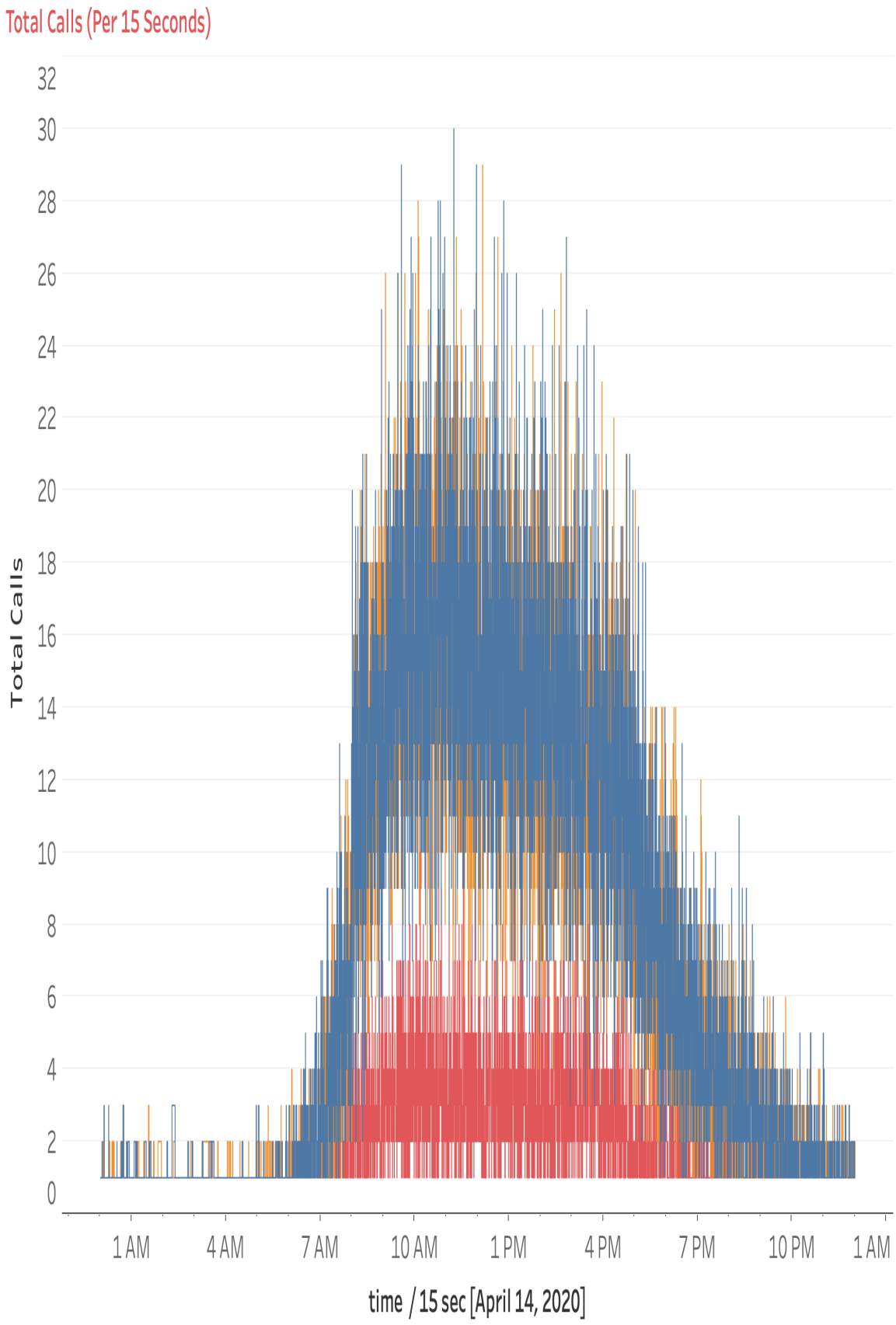

Total Calls (Per 15 Seconds)
Total Calls
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
0
1 AM
4 AM
7 AM
10 AM
1 PM
4 PM
7 PM
10 PM
1 AM
time / 15 sec [April 14, 2020]

Figure 4-16. : Total calls every 15 seconds of the day using a continuous axis.

You're starting to see patterns, like you did with the hourly plot, but this view is still too granular. Instead of every 15 seconds, what if it was every 15 minutes?

Blueprint: Continuous date-time axis for 15-minute intervals

Step 1. Create calculation.

Create a new calculation called time / 15 min.

Step 2. Write the following:

// time / 15 min

DATEADD(

"minute",

-(DATEPART("minute", [time]) % 15),

DATETRUNC("minute", [time])

)

The format for time / 15 min looks almost the same, except we've replaced 'second' with 'minute', and our third argument is now **DATETRUNC('minute', [time])** instead of time. This is because our analysis with time / 15 sec was already at the lowest level in Tableau (seconds). Since we are working at a higher level of data we need to roll-up all the values to the nearest minute.

Step 3. Roll up to the nearest minute.

Click and drag to replace time / 15 sec with time / 15 min.

On the marks card, click on path and change the line type to step. We like using a step path instead of a straight line because we know that the line represents all values across this 15-minute increment.

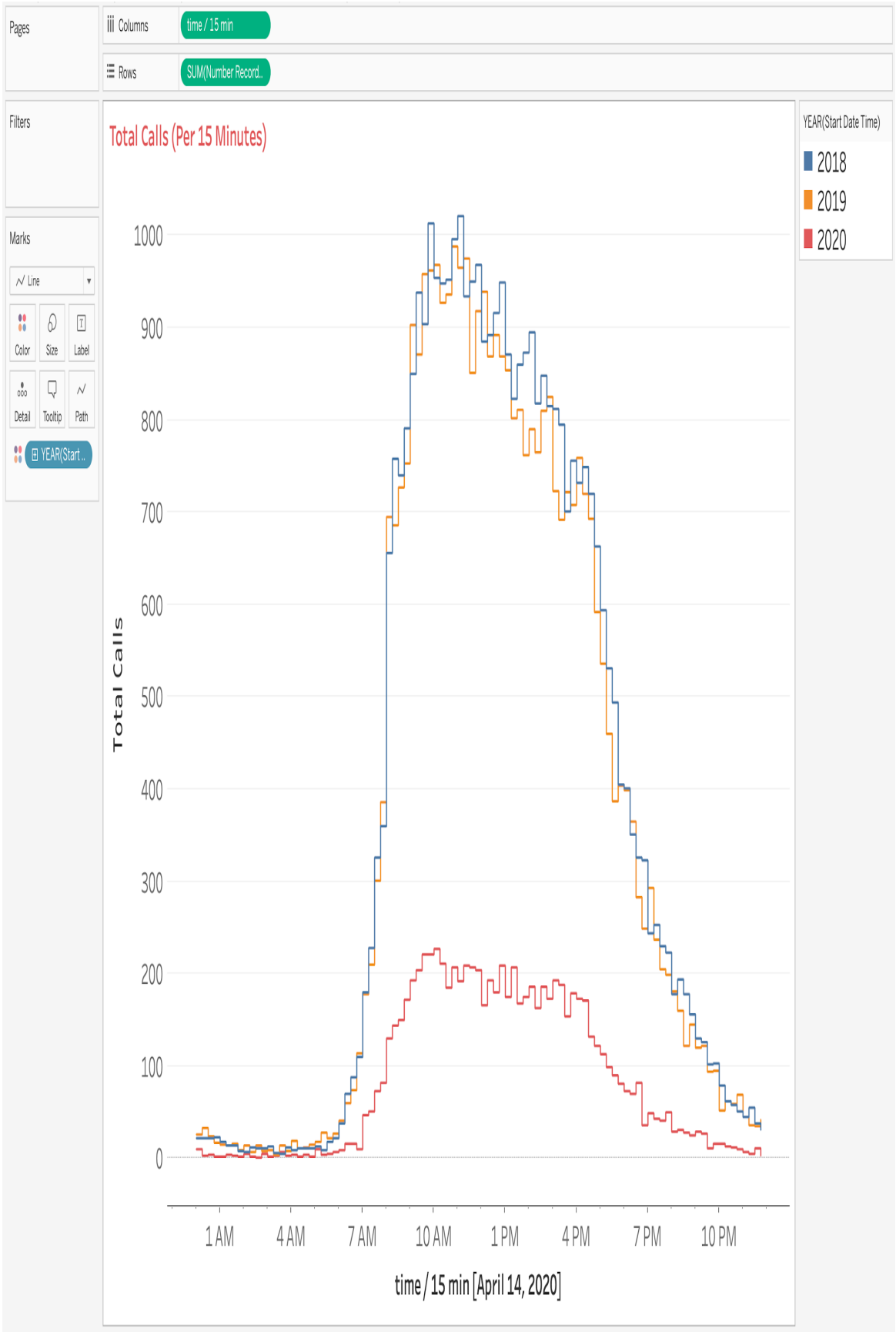

Pages
Columns
time / 15 min
Rows
SUM(Number Record..
Filters
Marks
Line
Color
Size
Label
Detail
Tooltip
Path
YEAR(Start..
Total Calls (Per 15 Minutes)
YEAR(Start Date Time)
2018
2019
2020
1000
900
800
700
600
500
400
300
200
100
0
Total Calls
1 AM
4 AM
7 AM
10 AM
1 PM
4 PM
7 PM
10 PM
time / 15 min [April 14, 2020]

Figure 4-17. Total calls every 15 minutes of the day using a continuous axis.

Finally, in Figure 4-13, we have a single axis where we can see patterns in the data at 15-minute intervals. This plot gives us lots of information about how quickly calls are scaling up each morning with much greater precision than when we sorted by hour, but it’s not too precise to be useful.

You still have the red line, representing 2020, which is much lower than all other values. This might be because we’ve only collected data up until April 14, 2020. Because business might be seasonal, it might be worth comparing across like time periods. But before we do that, we want to take a deeper look at calls per 15 minutes.

Step 4. Sort by day of the week

Specifically, let’s look at calls per 15 minutes by day of the week by adjusting the dimension on color. Right-click on YEAR(Start Date Time) and change the date type to a discrete date part of weekday. Feel free to edit the colors afterward.

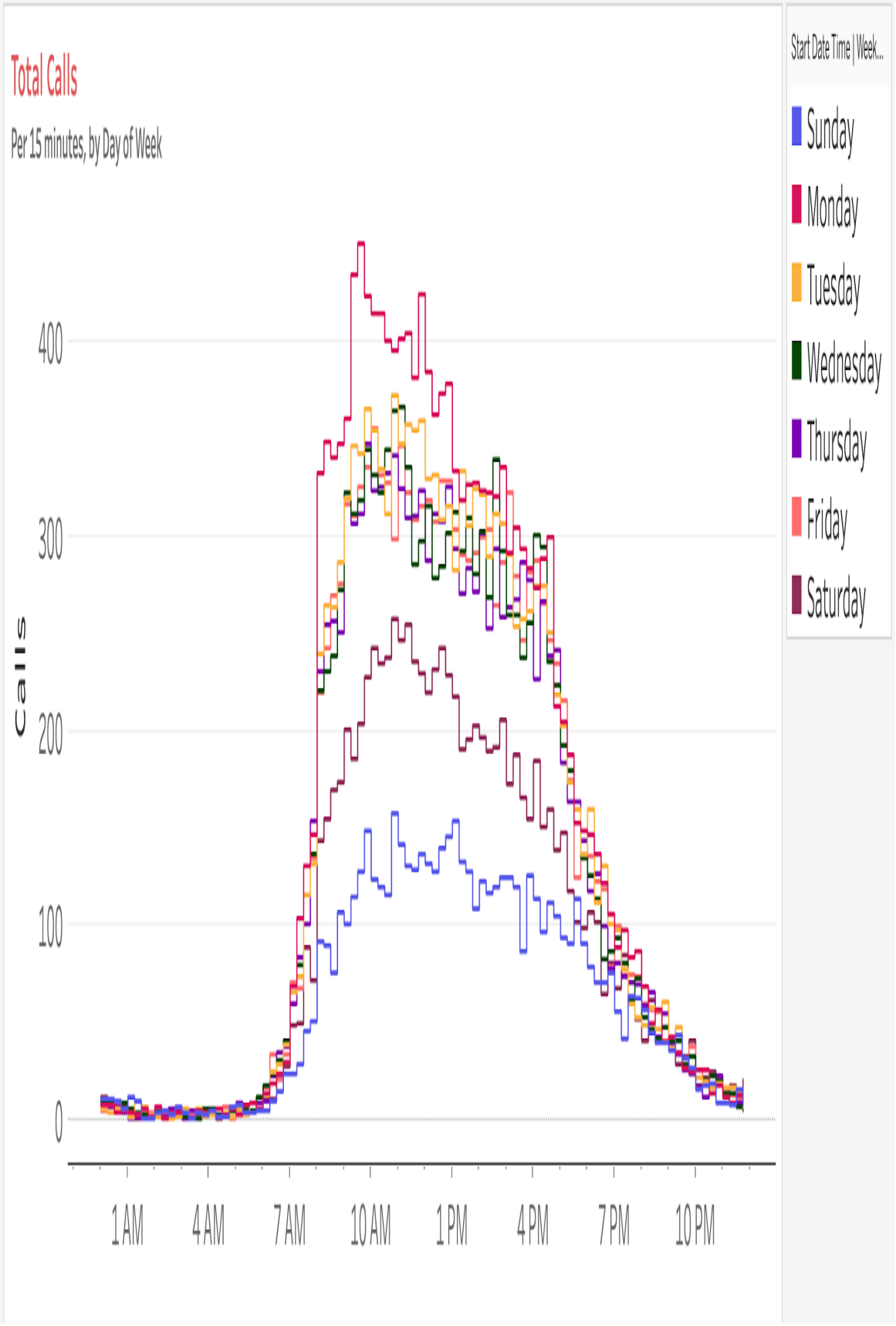

Total Calls
Per 15 minutes, by Day of Week
Calls
400
300
200
100
0
1 AM
4 AM
7 AM
10 AM
1 PM
4 PM
7 PM
10 PM
Start Date Time | Week...
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

Figure 4-18. : total calls every 15 minutes of the day, colored by day-of-the-week using a continuous axis.

Heatmaps (Highlight Tables)

The information in Figure 4-18 is extremely useful, but there are just too many lines to read through the insights. When we're working with line charts that have several lines, as you learned in chapter 3, we immediately consider other chart types. Our go-to chart type for this scenario is the heatmap (Tableau calls it a highlight table). Heatmaps allow audiences to see change via color, intensity, or hue rather than through direction. This heatmap is displayed as a matrix so that anyone can easily track change for a single member in a dimension.

Let's create a heatmap that reimagines the same analysis from the line chart in Figure 4-18.

Blueprint: Essential Heatmap

Create a new sheet

Change the mark type to square

Create a custom date for the date part of weekday using Start Date Time and place that on columns

Place time / 15 min on rows but choose the hour date part.

Place [Number of Records] on color.

Choose a color palette that works for you. I'm choosing a custom color palette.

The result is the heatmap in Figure 4-15.

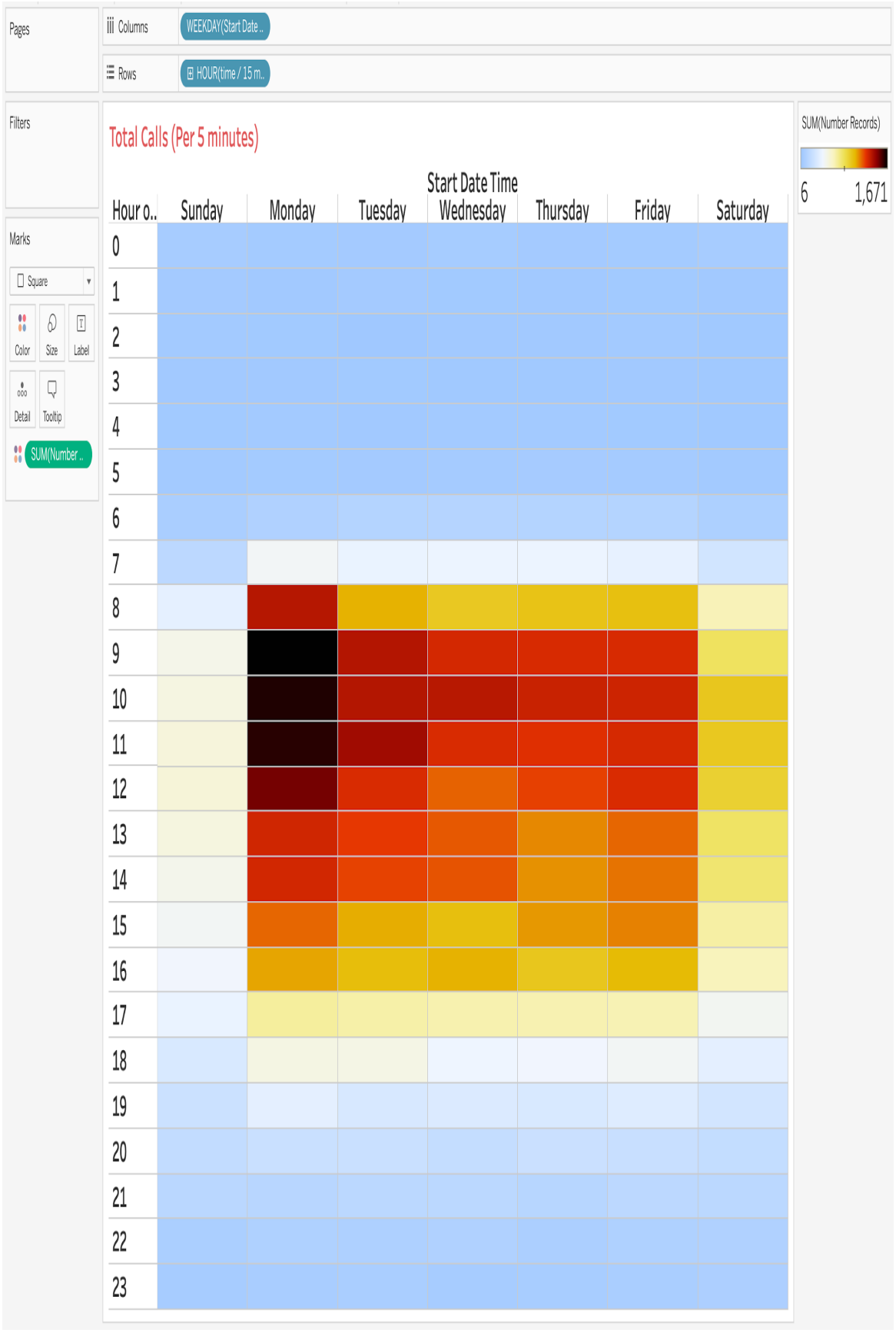

Pages
Columns
WEEKDAY(Start Date ..
Rows
HOUR(time / 15 m..
Filters
Marks
Square
Color
Size
Label
Detail
Tooltip
SUM(Number ..
Total Calls (Per 5 minutes)
Start Date Time
Hour o..
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
SUM(Number Records)
6
1,671

Figure 4-19. Total calls every hour of the day and day of the week, using a heatmap.

Once again you see, through change in color, how calls begin to pick up in the 8-o'clock hour. But you are also able to spot that Mondays, particularly in the morning, are extremely busy. Weekend are quieter, but Sunday is exceptionally slow. We also see that calls begin falling off after hour 17 (5PM).

A heatmap can be extremely helpful even when the data is fairly granular. Heatmaps' value only increases as more complexities and detail are added to a visualization.

Blueprint: Create a more detailed heatmap

Duplicate your visualization from Blueprint 4-6

Create a custom date for the date part of month using Start Date Time

Place it on columns and to the left of your weekday calculation on columns.

Click the + on HOUR(time / 15 min) on rows. This will show time rounded to the nearest 15 minutes.

You'll notice some whitespace where there are no values on your dashboard. If you want to add marks for those locations you'll need to use a lookup calculation: ZN(LOOKUP(SUM([Number of Records]),0)). (We'll talk about this in more detail in Chapter 6.) Add this calculation to color.

Format your column and row headers to be more readable. Your result should look like Figure 4-16.

Call Volume Heat Map

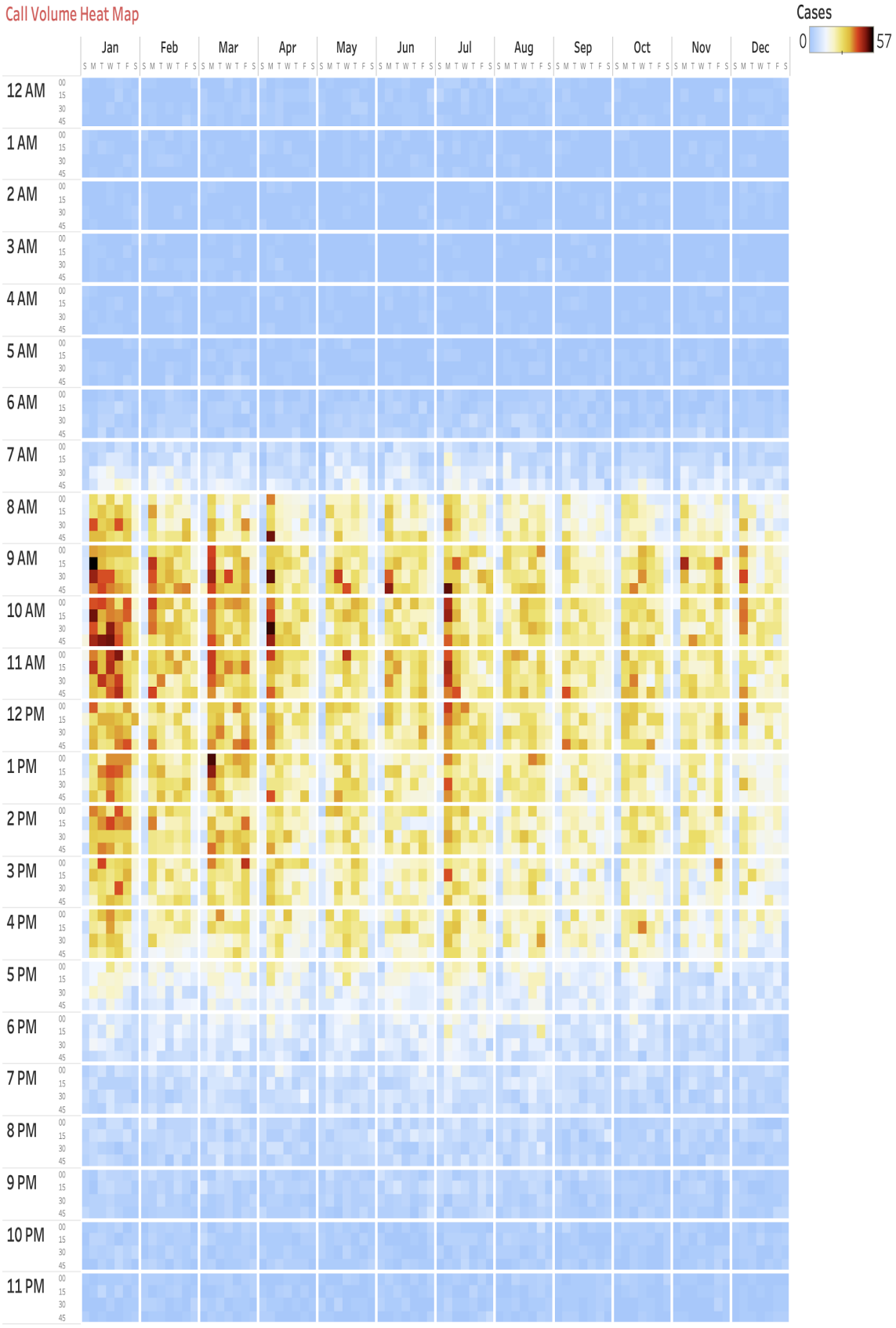


Figure 4-20. Total calls every 15 minutes of the day by month and day-of-the-week using a heatmap.

We've chosen to format dividers at the hour and Month level. This allows your audience to quickly navigate to segments of the analysis.

So what insights can you glean from this data?

- Calls pick up at 8AM for weekdays almost every month of the year.
- There tend to be more calls in the late evening during the summer months.
- Regardless of day of the week, there are often a lot of calls in January.
- In April, July, and December, there are clearly more calls on Mondays.

Identifying call patterns from this visualization could be extremely helpful in staffing. Mondays are always busy, but especially in April, July, and December: it might make sense to bulk up on those days. With call volumes higher later into the evening in the summer, it might mean staffing up until 6PM instead of 5PM. This could be offset with staff working fewer hours on non-Mondays during November and December.

We love heatmaps. They are underrated tools for representing time and are particularly useful when data are many members of a dimension.

Call Center Case Study: Comparing Values Year-to-Date

Now that they've learned about our call frequency across days and months, your call center clients are curious how things have changed from year to year. To do so, let's go back to our plot of total calls per 15 minutes by year (Figure 4-17).

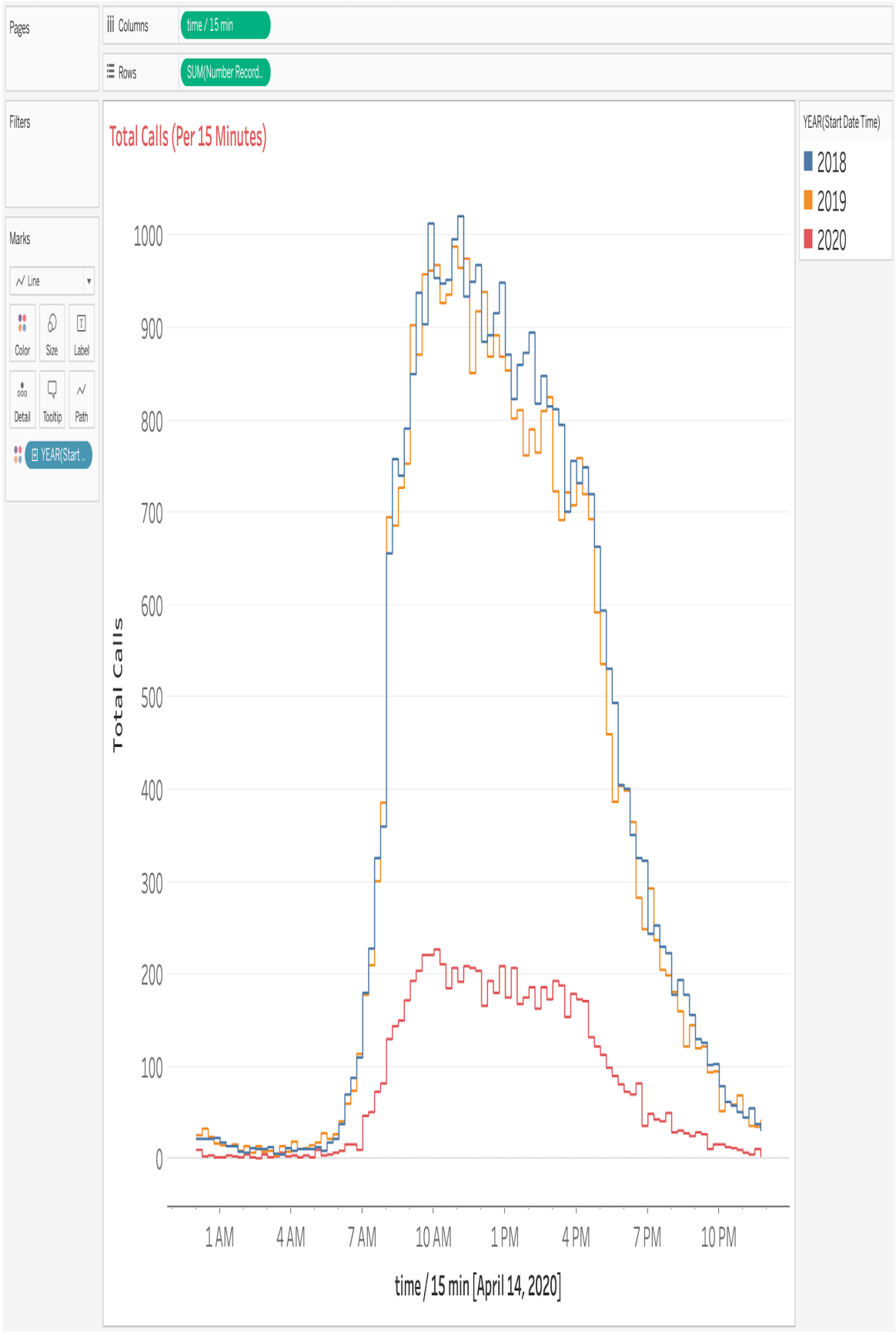

Pages
Columns
time / 15 min
Rows
SUM(Number Record..
Filters
Marks
Line
Color
Size
Label
Detail
Tooltip
Path
YEAR(Start..
Total Calls (Per 15 Minutes)
YEAR(Start Date Time)
2018
2019
2020
1000
900
800
700
600
500
400
300
200
100
0
Total Calls
1 AM
4 AM
7 AM
10 AM
1 PM
4 PM
7 PM
10 PM
time / 15 min [April 14, 2020]

Figure 4-21. Total calls every 15 minutes of the day by year.

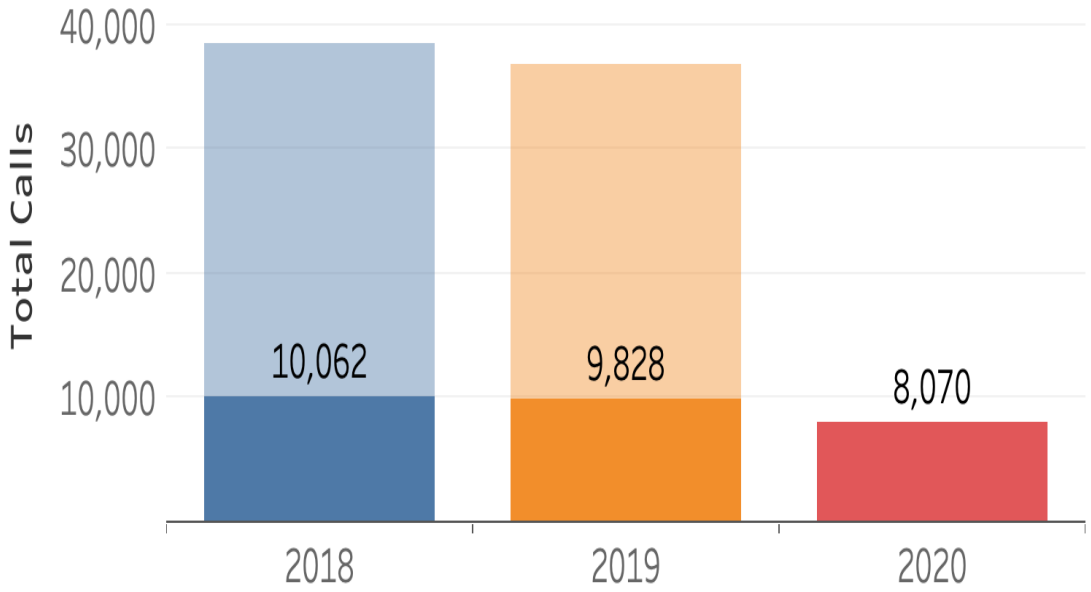
We see that calls are down, but that's because we're at a different point in 2020 than the other years: our data is only available through April 14, 2020. What might be fairer is to compare calls for 2018, 2019, and 2020 through April 14, and not by their overall totals.

This is a challenge we face quite often regardless of data type: comparing like period to like period. So how can you solve the problem? With a well-designed date calculation.

The goal of the next two blueprints is to create a calculation to compare most recent date for the most recent year to the same date for prior years. For the next blueprint, you will recreate a bar chart that shows progress to the total. This will allow your audience to keep an eye on overall values while simultaneously displaying comparable year-to-date values.

Then we'll apply a filter to our line chart, allowing for a proper-year-to-date comparison. Figure 4-18 shows the results of Blueprints 4-8 and 4-9.

Progress vs. Last Year



Total Calls (Per 15 Minutes)

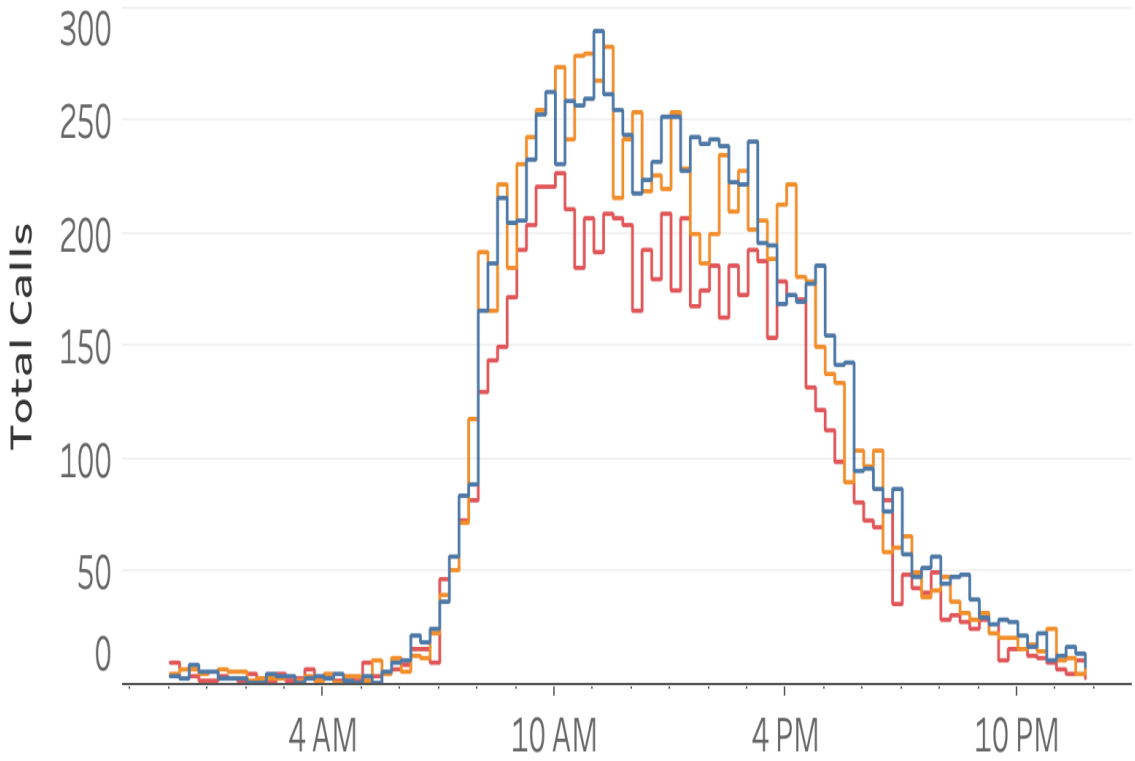


Figure 4-22. Top: total calls versus the total for the two previous years. Bottom: total calls filtered to the same day of the year for three years.

Blueprint: Progress to total, using two bar charts

Step 1: Build the calculations

Step 1a: Normalize the dates to the same year. Create a calculation called Start Date Time | Same Year. This will place all dates in the same year

// Start Date Time | Same Year

DATEADD(

“year”,

DATEDIFF(“year”, {MAX([Start Date Time])}, [Start Date Time]),

DATETRUNC(“day”, [Start Date Time])

)

Step 1b: Create a Boolean to show if a value is greater than the latest date. Create a second calculation called Start Date Time | Same Year | TF. This will be a Boolean that detects if a date is less than or equal to the day of the year for the most recent year.

// Start Date Time | Same Year | TF

DATEPART(“dayofyear”, [Start Date Time | Same Year]) <= DATEPART(“dayofyear”, {MAX([Start Date Time])})

Step 1c: Calculate the year-to-date calls. Create a third calculation called Total Calls | YTD.

```
// Total Calls | YTD
SUM(
  IF [Start Date Time| Same Year | TF]
  THEN [Number Records]
  END
)
```

This will return year-to-date values for each year.

Step 2: Build the visualization.

Add Number of Records to rows.

Add Total Calls | YTD to the right of Number of Records on rows.

Create a synchronized dual axis chart.

Add Start Date Time as a discrete year name to columns and to color.

Set the opacity on the SUM(Number of Records) marks card to 40%.

Format your visualization by removing the column and row dividers, adding a darker columns axis ruler and tick marks, styling your grid lines, showing just a left axis, and renaming the axis.

Show labels on the AGG(Total Calls | YTD) marks card.

This will result in the bar-on-bar chart shown in Figure 4-19.

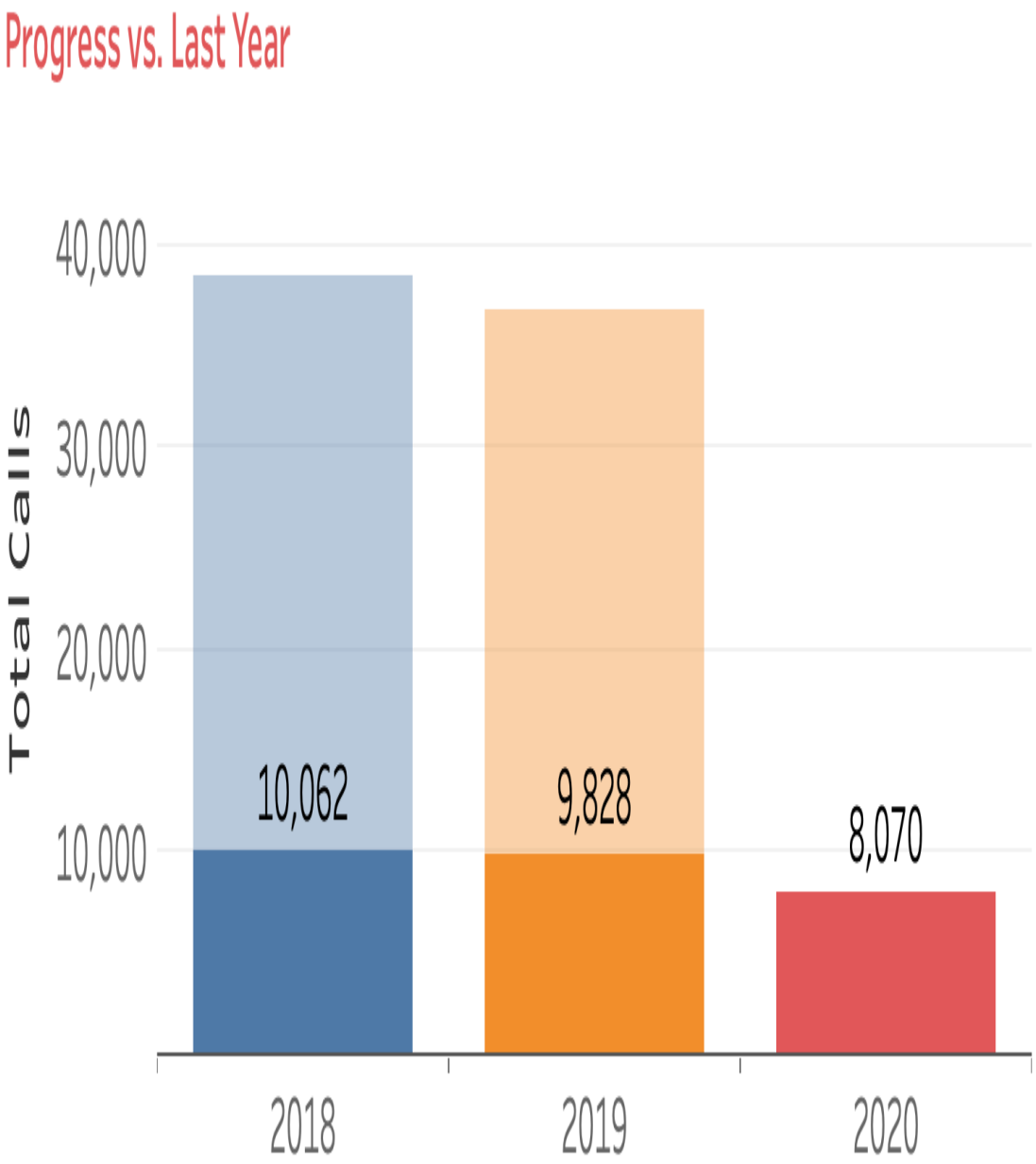


Figure 4-23. Year-to-date progress relative to the current date for the most recent year.

In this visualization, you embedded an IF statement inside the aggregation to do the filtering inside the calculation. This is something you should do regularly to allow a visualization to be dynamic. As the year continues, we'll see the 2020 bar increase. The 100% opaque bars in 2018 and 2019 will also continue to grow. These bars will eventually cap at the values shown in the 40% opaque bars shown on the same axis.

Take a look back at Figure 4-17, which shows total calls for all dates in 2018 and 2019. In 2020, just shows dates through April 14. It's not a like-for-like comparison. It would be great if we could compare these values.

Unlike the last blueprint, where we used an IF statement to filter our data, we are going to explicitly place a calculation created on the filters shelf.

Blueprint: Comparing similar periods on a line chart

Duplicate your final visualization from "Blueprint: Continuous date-time axis for 15-minute intervals."

Ensure you have completed Step 1a and Step 1b from "Blueprint: Progress to total, using two bar charts."

Edit Weekday of Start Date Time and change the date type to discrete year.

Add Start Time Date | Same Year | TF to the Filter shelf, then select TRUE and click OK.

Edit the axis and remove the axis title.

The result is shown in Figure 4-20.

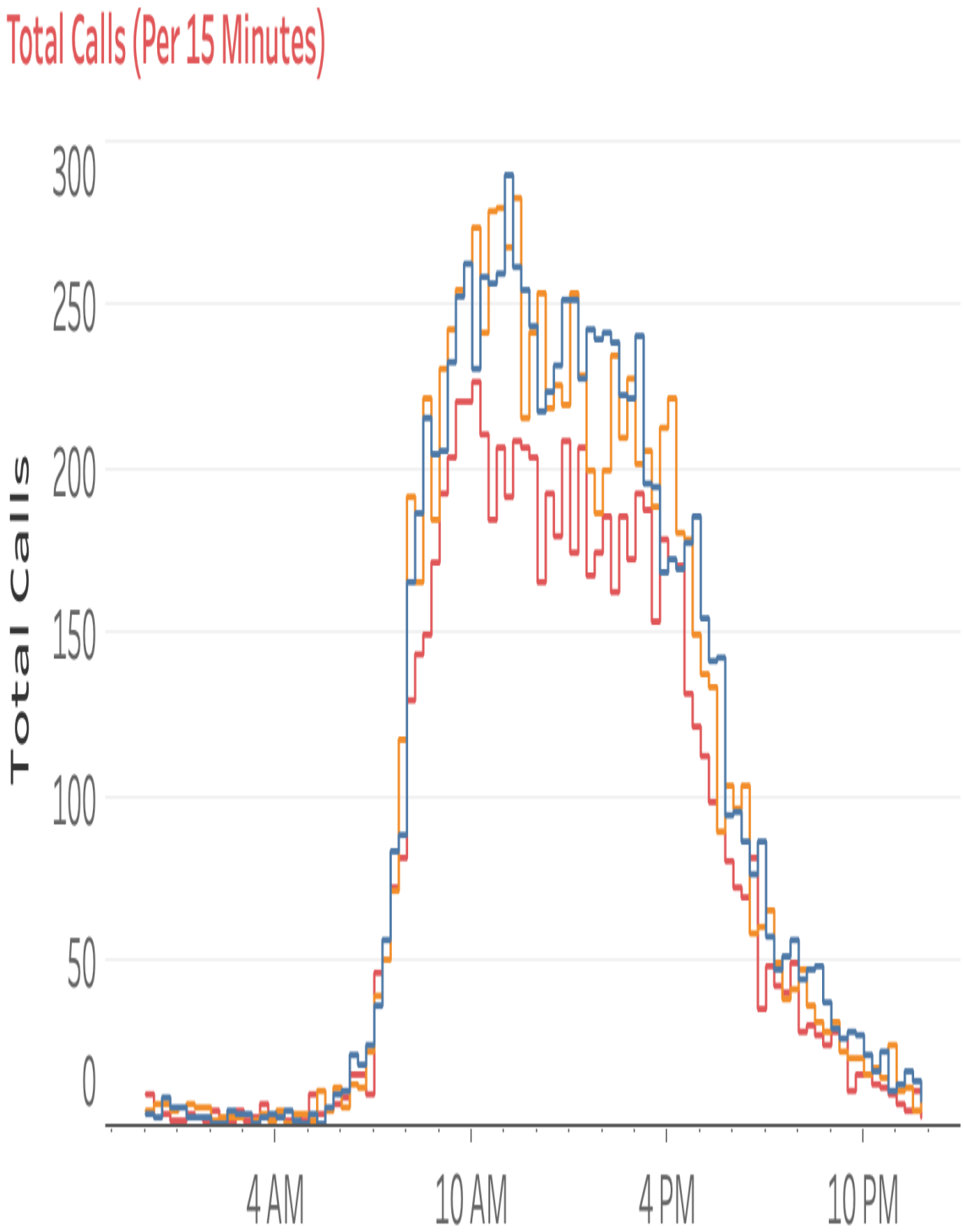
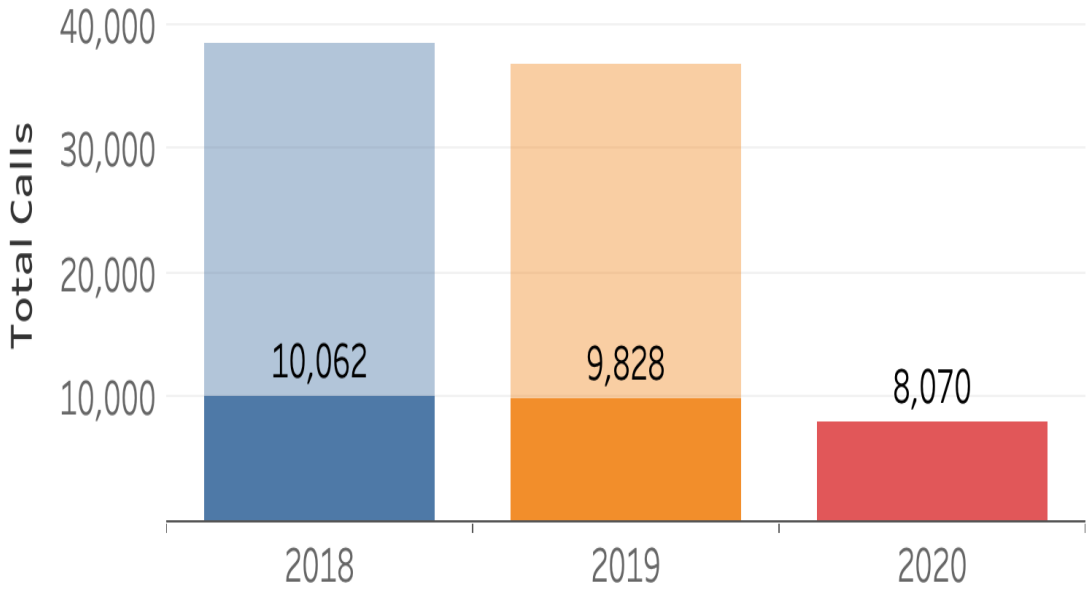


Figure 4-24. Total calls every 15 minutes of the day filtered to the same day of the year for 2018, 2019, and 2020.

When you place the results from this blueprint and the previous one side by side (Figure 4-21), you get a miniature year-to-date dashboard that allows

your audience to track total calls year to date as well as when the calls occurred.

Progress vs. Last Year



Total Calls (Per 15 Minutes)

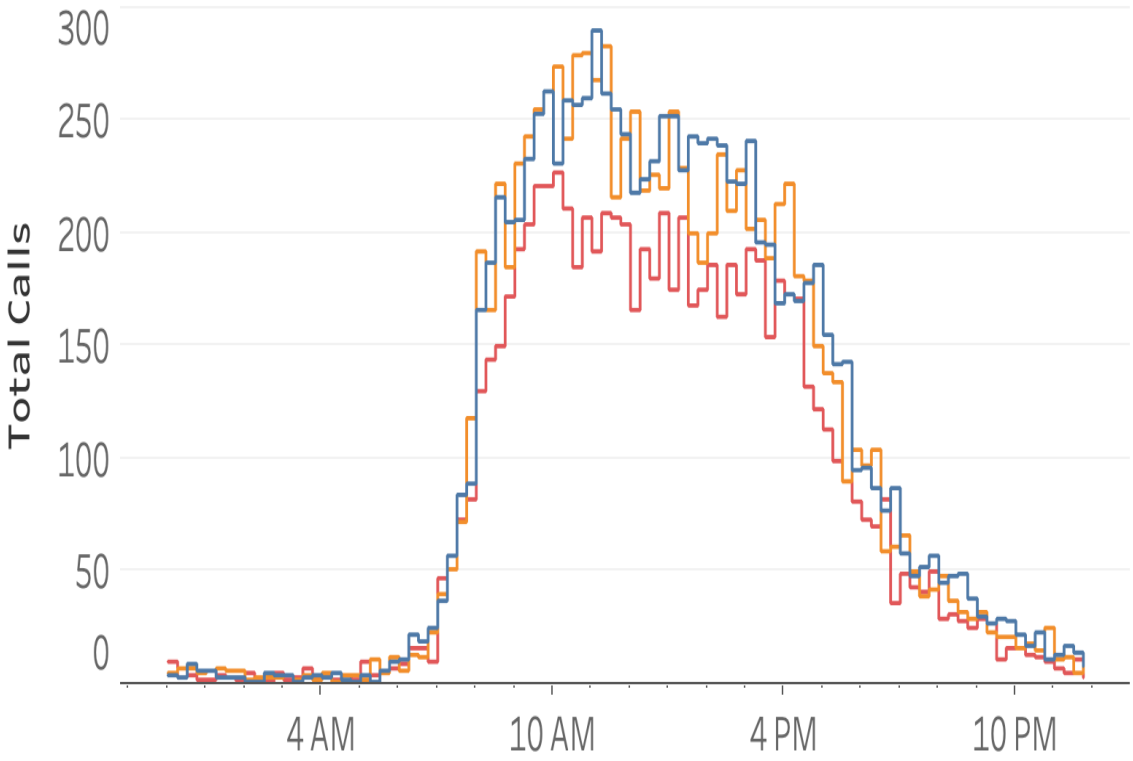


Figure 4-25. A mini dashboard showing the same day of year comparisons of total calls and every 15 minutes of the day for 2018, 2019, and 2020.

Automated Reports

Next, we are going to look at automating reports using custom calculations. Whether it's call center data, financial reports, or student enrollment numbers, many people spend a lot of time developing tables that automatically update at the end of a month.

While implementing these tables takes a little time, the effort saves a lot of time. Novice Tableau users, upon receiving new data, often manually update their data, then go to a dashboard and edit a filter to include the updated data. The goal of the next blueprint is to show how you can automatically update a dashboard based on the data that is on the dashboard.

Call Center Case Study: Automating Reports for Month-over-Month and Year-over-Year Change

Take a look at the table in Figure 4-22. It shows average calls per day as well as the percentage of change in calls month-over-month and year-over-year. The table shows data from March 2019 through March 2020. We decided to not report anything for April because our data only reports halfway through April. When we have data for the final day of April, the report will automatically update so that the table shows metrics from April 2019 through April 2020.

Additionally, this table shows a breakdown of calls by call reason and aggregates to totals.

Call Reason		Mar-19	Apr-19	May-19	Jun-19	Jul-19	Aug-19	Sep-19	Oct-19	Nov-19	Dec-19	Jan-20	Feb-20	Mar-20
Total	Calls/Day	**30.0**	**102.3**	**105.2**	**113.1**	**116.3**	**111.9**	**105.1**	**99.7**	**97.1**	**85.3**	**82.6**	**77.8**	**75.3**
	MoM	**-67%**	**241%**	**3%**	**8%**	**3%**	**-4%**	**-6%**	**-5%**	**-3%**	**-12%**	**-3%**	**-6%**	**-3%**
	YoY	**-64%**	**65%**	**2%**	**6%**	**3%**	**-4%**	**-7%**	**-5%**	**-3%**	**-12%**	**-3%**	**-5%**	**-8%**
3	Calls/Day	12.0	48.2	51.5	52.2	55.2	51.2	48.4	43.3	44.0	37.6	37.8	38.1	36.8
	MoM	-73%	301%	7%	1%	6%	-7%	-5%	-11%	2%	-14%	0%	1%	-4%
	YoY	-75%	74%	6%	1%	6%	-9%	-6%	-11%	2%	-14%	0%	1%	-11%
1	Calls/Day	13.0	36.9	38.6	40.7	41.5	39.6	36.6	37.5	34.7	30.7	29.6	27.1	27.8
	MoM	-61%	184%	5%	5%	2%	-5%	-8%	2%	-7%	-11%	-4%	-8%	3%
	YoY	-57%	58%	4%	5%	2%	-5%	-8%	2%	-9%	-12%	-3%	-7%	5%
2	Calls/Day	4.0	10.9	10.1	13.7	14.5	14.8	14.7	13.6	13.6	12.9	11.2	8.6	7.0
	MoM	-58%	173%	-7%	36%	6%	2%	0%	-8%	0%	-5%	-13%	-24%	-18%
	YoY	-55%	58%	-6%	17%	6%	2%	-1%	-9%	0%	-6%	-16%	-28%	-39%
Null	Calls/Day	1.0	6.4	4.9	6.4	5.0	6.3	5.5	5.6	4.9	4.3	4.1	4.1	4.3
	MoM	-80%	537%	-23%	31%	-22%	26%	-13%	2%	-13%	-13%	-4%	-1%	5%
	YoY	-48%	56%	-16%	15%	-17%	19%	-15%	2%	-14%	-12%	-3%	-1%	20%

Figure 4-26. Average calls per day and percentage of change in calls, month-over-month and year-over-year, March 2019 through March 2020.

Blueprint: Automated Rolling Table

Create the metric for our table, Calls/Day:

// Calls/Day

SUM([Number Records])/COUNTD(DATETRUNC(“day”, [Start Date Time]))

Step 1: Build the base table for this visualization.

Step 1b. Metric. Create the metric for our table, **Calls/Day**:

// Calls/Day

SUM([Number Records])/COUNTD(DATETRUNC(“day”, [Start Date Time]))

Start by adding Calls/Day to text. Add Call Reason to columns and sort the dimension descending by calls per day.

Step 1b. Custom date. Create a new custom date called Start Date Time | Month that returns monthly date values. Place this as a discrete value on columns.

Step 1c. Place totals at top. Add column totals and place them at the top: use the top menu, select Analysis > Totals, then select Show Column Grand Totals and Column Totals to Top (Figure 4-23).

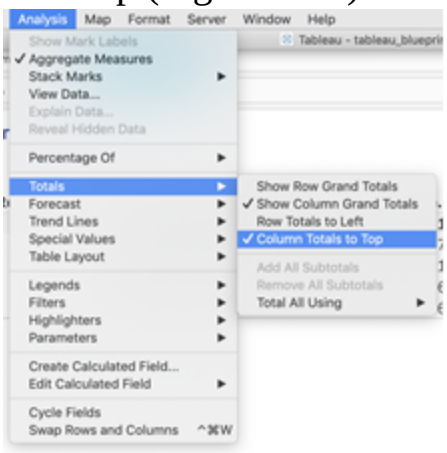
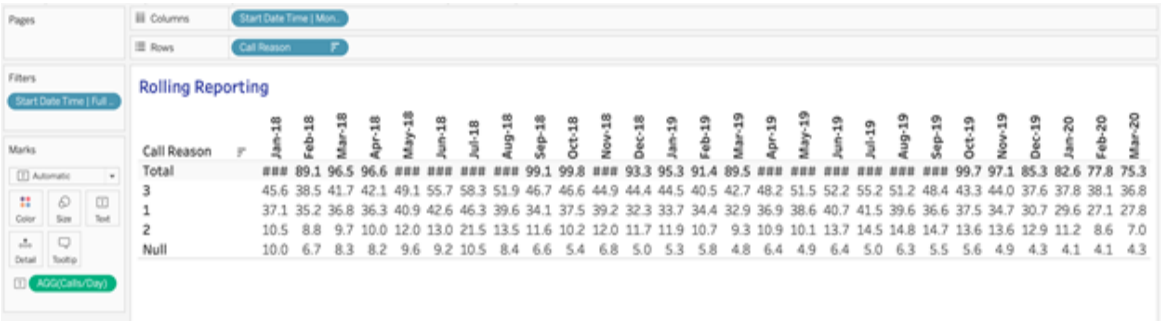


Figure 4-27. To add totals to the top, use the top menu, select Analysis > Totals, then select Show Column Grand Totals and Column Totals to Top.

Step 2: Formatting

Format your table so that only row dividers exist, as shown in Figure 4-24. Add band color to your totals only. This will serve as the base for your month-over-month calculation, your year-over-year calculation, and the automations you will create.



Call Reason	Jan-18	Feb-18	Mar-18	Apr-18	May-18	Jun-18	Jul-18	Aug-18	Sep-18	Oct-18	Nov-18	Dec-18	Jan-19	Feb-19	Mar-19	Apr-19	May-19	Jun-19	Jul-19	Aug-19	Sep-19	Oct-19	Nov-19	Dec-19	Jan-20	Feb-20	Mar-20
Total	###	89.1	96.5	96.6	###	###	###	###	99.1	99.8	###	93.3	95.3	91.4	89.5	###	###	###	###	###	###	99.7	97.1	85.3	82.6	77.8	75.3
3	45.6	38.5	41.7	42.1	49.1	55.7	58.3	51.9	46.7	46.6	44.9	44.4	44.5	40.5	42.7	48.2	51.5	52.2	55.2	51.2	48.4	43.3	44.0	37.6	37.8	38.1	36.8
1	37.1	35.2	36.8	36.3	40.9	42.6	46.3	39.6	34.1	37.5	39.2	32.3	33.7	34.4	32.9	36.9	38.6	40.7	41.5	39.6	36.6	37.5	34.7	30.7	29.6	27.1	27.8
2	10.5	8.8	9.7	10.0	12.0	13.0	21.5	13.5	11.6	10.2	12.0	11.7	11.9	10.7	9.3	10.9	10.1	13.7	14.5	14.8	14.7	13.6	13.6	12.9	11.2	8.6	7.0
Null	10.0	6.7	8.3	8.2	9.6	9.2	10.5	8.4	6.6	5.4	6.8	5.0	5.3	5.8	4.8	6.4	4.9	6.4	5.0	6.3	5.5	5.6	4.9	4.3	4.1	4.1	4.3

Figure 4-28. The table showing calls per day by call reason and month.

This will serve as the base for your month-over-month calculation, your year-over-year calculation, and the automations you will create.

Step 3: Add period-over-period calculations

Step 3a. Create the month-over-month calculation. This is done with a table calculation, which you can create using a new calculation. Create a

new calculation called Calls/Day | % Change 1:

```
// Calls/Day | % Change 1
(ZN([Calls/Day]) - LOOKUP(ZN([Calls/Day]), -1)) / ABS(LOOKUP(ZN([Calls/Day]),
-1))
```

This calculation creates a percent change based on the previous value, in this case the previous month.

Step 3b. Compare this year with last year. Because many businesses are seasonal, it's often better to compare this year's values to the previous year. To do this, create a new calculation called Calls/Day | % Change 12:

```
// Calls/Day | % Change 12
(ZN([Calls/Day]) - LOOKUP(ZN([Calls/Day]), -12)) / ABS(LOOKUP(ZN([Calls/Day]),
-12))
```

You'll notice that the calculation is just slightly different: -1 has been changed to -12.

Step 3c. Add measure names and values. Double-click on Calls/Day | % Change 1 and Calls/Day | % Change 12. This will convert the table to include Measure Names and the text is now Measure Values. The table calculations in Calls/Day | % Change 1 and Calls/Day | % Change 12 need some updating, but you can wait until we have all the components of the visualization on the view.

Step 4: Filter to complete months

Let's now move on to showing only full months.

Step 4b. Calculate the maximum date of Start Date Time. You'll do this by writing a calculation called Start Date Time | Max Date:

```
// Start Date Time | Max Date
{MAX([Start Date Time])}
```

Step 4b. Level of detail calculation. Here you'll use a level of detail calculation to calculate the maximum date in your dataset. You can monitor this date for when you need to update your table. This will allow you to calculate relevant time periods dynamically. You just need to find the start

and end points of your dynamic table. Let's first calculate the last day of the last full month and call the calculation Last Day of Last Full Month

```
// Last Day of Last Full Month
DATETRUNC("month", [Start Date Time | Max Date] + 1) - 1
```

Use this calculation to create a Boolean to filter data in the current month–which is not yet complete. Create a calculation called Start Date Time | Full Months:

```
// Start Date Time | Full Months
[Start Date Time] <= [Last Day of Last Full Month]
```

Add this calculation to filters and select TRUE.

Step 5: Filter to the last 13 complete months.

Narrow this visualization down to the most recent 13 months. Create a new calculation called Start Date Time | Last 13 Months:

```
//Start Date Time | Last 13 Months
[Start Date Time] > DATEADD("month", -12, [Last Day of Last Full Month] + 1) - 1
```

Place this calculation to the left of Start Date Time | Month. Right-click on the False header and deselect Show Header (Figure 4-29).

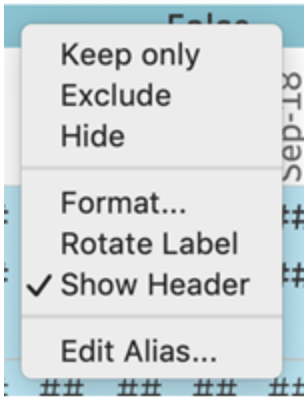


Figure 4-29. Hiding the header by deselecting Show Header.

Step 6: Finalize the table calculations

Edit the Calls/Day | % Change 1 and Calls/Day | % Change 12 table calculations in the Measure Values marks card so that only Call Reason is deselected, as shown in Figure 4-26.

Figure 4-30. The table calculation settings for Calls/Day | % Change 1 and Calls/Day | % Change 12.

Step 7: Edit Aliases

Be sure to format both month-over-month and year-over-year calculations as percentages. Finally, right-click and edit the alias (Figure 4-27). Change Calls/Day | % Change 1 to MoM (month-over-month) and Calls/Day | % Change 12 to YoY (year-over-year).

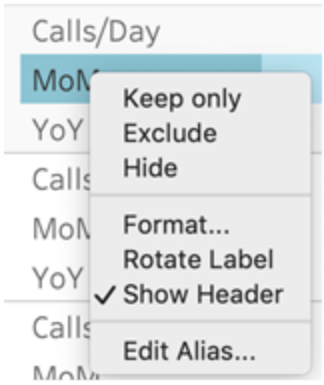


Figure 4-31. Right-click the Measure Names and select Edit Alias.

The result of all this work (Figure 4-22) is a humble table that provides a lot of great insights and is automatically updated each month.

For the most part, developers spend very little time thinking about what a year, month, or week even means. We just assume that a year goes from January 1 to December 31. But organizations define a fiscal year in many ways. This next section provides a brief overview of working with fiscal dates in Tableau.

Nonstandard calendars

Some organizations work with the standard Gregorian calendar as their fiscal year: January 1 to December 31. But a fiscal year can start at any time: January 1, June 5, even the fifth Monday of the standard calendar year. It's all relative. Tableau provides some flexibility.

If your calendar year starts at the beginning of a month, you can standardize this by right clicking, then navigating to Default Properties → Fiscal Year Start → Month of Fiscal Year (Figure 4-28). This simplifies the hierarchy associated with that particular date measure.

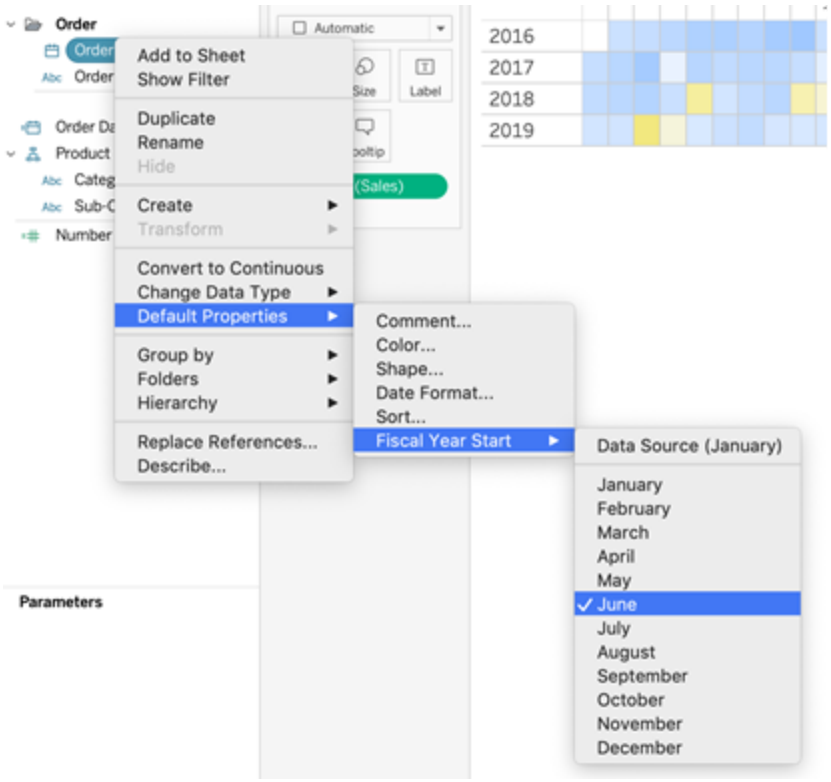


Figure 4-32. Right-click on a date to change the default properties of the fiscal year start. In this example, the fiscal year start is set to June.

Some organizations start the fiscal year or month on the first day of the week that year or month starts. For example, if January 1 falls on a Tuesday, then that fiscal year would start on December 30. This calendar type is called ISO-8601. (We'll just call it an "ISO calendar" in this section.)While the name is funky, just know that the calendar is week-based. You can specify it by right-clicking on the date value on your view and selecting ISO-8601 Week-Based (Figure 4-29).

In figures 4-29 and 4-30, you can see how data from a standard calendar can differ ever so slightly from an ISO-week calendar.

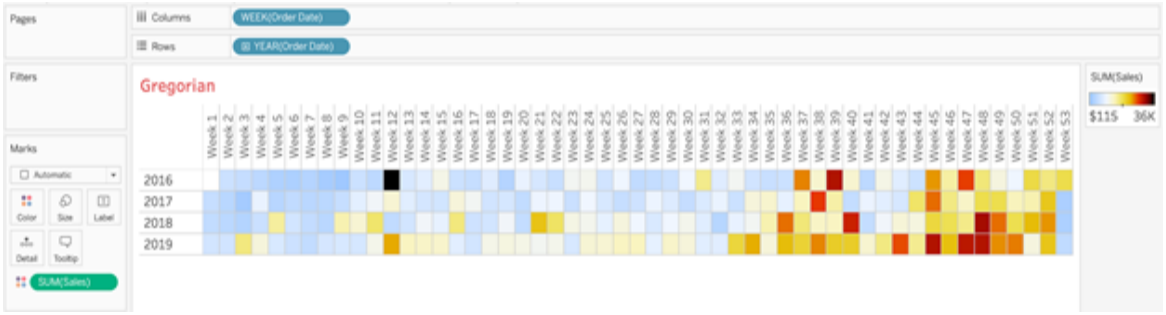


Figure 4-33. Data in a standard calendar

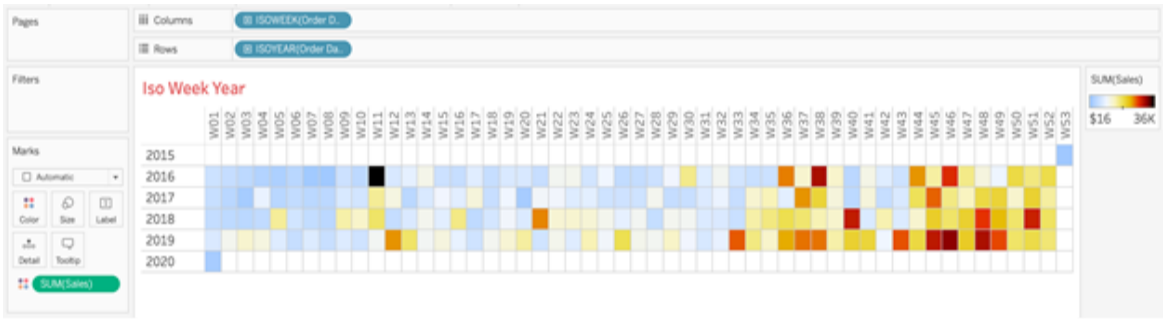


Figure 4-34. Data in an ISO calendar

Take a moment to build a visualization with a June fiscal start. To keep it simple, imagine you are building a bar chart that shows total sales by month and fiscal year.

Blueprint: Monthly Bar Chart with a June 1 Fiscal-Year Start

Duplicate the Order Date field and Call it Order Date | June

Right click on Order Date | June and change the default fiscal year start to June

Add Order Date | June as continuous data value by month to rows.

Add SUM(Sales) to columns.

Add YEAR(Order Date | June) to color.

Right-click on the September 2016 bar and add an annotation to the mark displaying the date and the total sales.

Your result should replicate Figure 4-33.

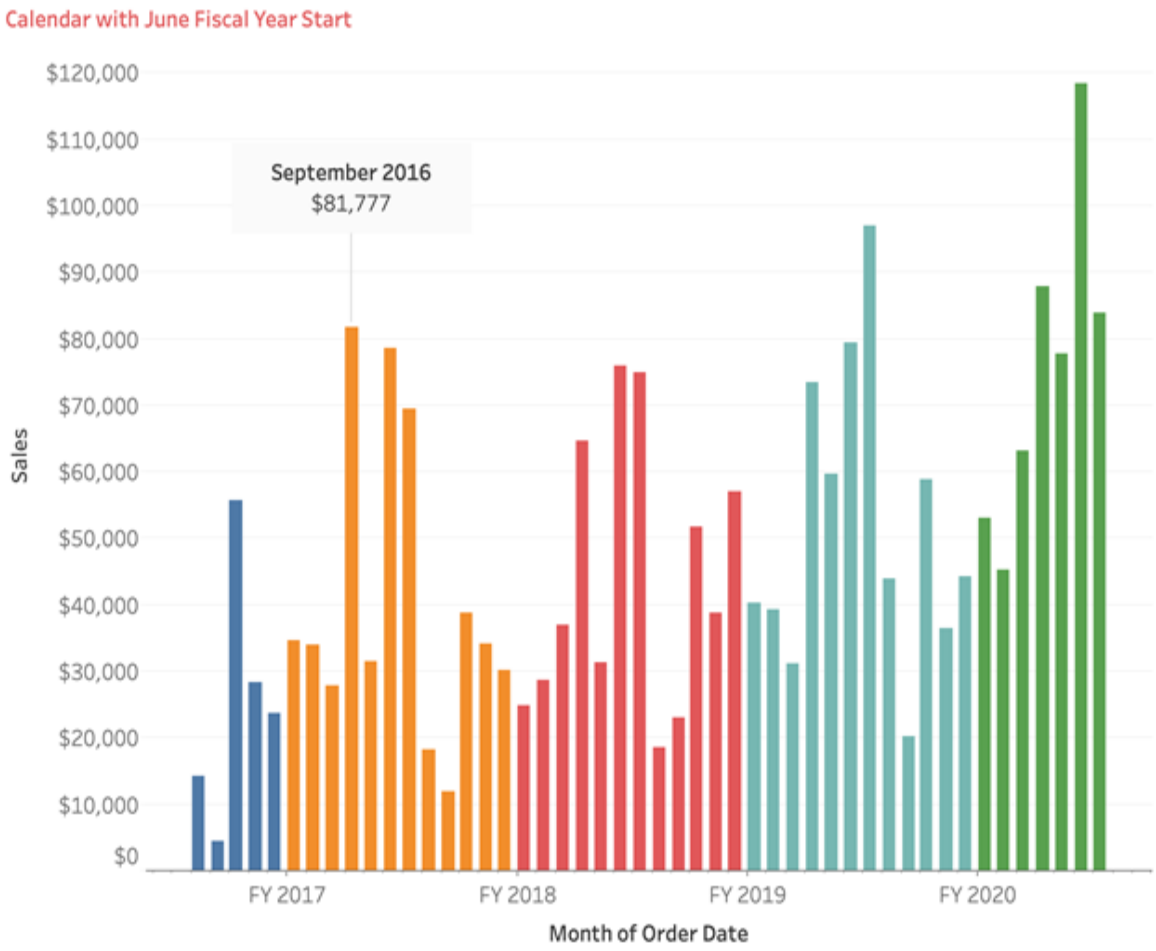


Figure 4-35. Sales by month, colored by fiscal year, using a June start to the fiscal year.

Retail Case Study: Visualizing the 4-5-4 Calendar

Retailers often use the 4-5-4 calendar, which allows them to compare sales by dividing the year into months based on a repeating pattern of 4 weeks, 5 weeks, and 4 weeks. Retailers use this calendar because it tends to make holidays line up, and because the same number of Saturdays and Sundays are displayed in comparable months. The 4-5-4 sales calendar is not perfect: because the calendar is based on 52 weeks, or 364 days, this leaves an extra day each year to be accounted for. To adjust for this, every five to six years

a week is added to the fiscal calendar. This occurred in 2012 and 2017 and will happen again in 2023.

The 4-5-4 calendar year varies from year to year. When February 1 occurs on a Thursday, Friday, or Saturday, the calendar year starts the Sunday after February 1. If February 1 occurs on Sunday, Monday, Tuesday, or Wednesday, then the calendar year starts the Sunday of the week of February 1.

The next blueprint is focused on building date components for the 4-5-4 retail calendar. These include week of the year, month of the year, quarter of the year, and week of the quarter. After you build the components, you will build a visualization highlighting some of those calculations.

Blueprint 4-12: Bar Chart using the 4-5-4 Retail Calendar

Step 1: Calculate February 1.

Create a calculation that calculates February 1. Call the calculation Feb 1.

```
// Feb 1
DATEADD(“month”, 1, DATETRUNC(“year”, [Order Date]))
```

Step 2: Determine the start of the 4-5-4 Calendar year

Calculate the start of the calendar year based on whether February 1 is after Wednesday in the week. Name the calculation 454 Year Start.

```
// 454 Year Start
IF DATEPART('weekday', [Feb 1]) > 4

THEN DATETRUNC('week', DATEADD('week', 1, [Feb 1]))

ELSE DATETRUNC('week', [Feb 1])

END
```

Step 3: Determine the start of the 4-5-4 calendar year for the prior year

Calculate February 1 for the prior year. Name the calculation Feb 1 | PY.

```
// Feb 1 | PY
```

DATEADD('year', -1, DATEADD(“month”, 1, DATETRUNC(“year”, [Order Date])))

Calculate the start of the previous calendar year. We will use this calculation with the current year values to determine the week number of the calendar year. Label the calculation as 454 Prior Year Start.

// 454 Prior Year Start

IF DATEPART('weekday', [Feb 1 | PY]) > 4

THEN DATETRUNC('week', DATEADD('week', 1, [Feb 1 | PY]))

ELSE DATETRUNC('week', [Feb 1 | PY])

END

Step 4: Parse the retail weeks of the year

Calculate the retail week.

// Retail Week

IF [454 Year Start] <= [Order Date]

THEN DATEDIFF('week', [454 Year Start], [Order Date]) + 1

ELSE ({FIXED [Feb 1] : MAX(DATEDIFF('week', [454 Prior Year Start], DATETRUNC('year',[454 Year Start])))} + DATEPART('week', [Order Date]))

END

Step 5: Build additional calendar features

Now that you have the week, you can create components like Retail Quarter, Retail Month of Quarter, Retail Week of Quarter, Retail Month, and Retail Week of Month. Build out each of these calculations.

//Retail Quarter

FLOOR(([Retail Week]-1)/13)+1

// Retail Month of Quarter

```
IF [Retail Week of Quarter] <= 4
THEN 1
ELSEIF [Retail Week of Quarter] > 4 AND [Retail Week of Quarter] <= 9
THEN 2
ELSEIF [Retail Week of Quarter] > 9 AND [Retail Week of Quarter] <= 13
THEN 3
END
//Retail Week of Quarter
(([Retail Week] - 1) % 13) + 1
//Retail Month
IF [Retail Week] <= 4
THEN “February”
ELSEIF [Retail Week] > 4 AND [Retail Week] <= 9
THEN “March”
ELSEIF [Retail Week] > 9 AND [Retail Week] <= 13
THEN “April”
ELSEIF [Retail Week] > 13 AND [Retail Week] <= 17
THEN “May”
ELSEIF [Retail Week] > 17 AND [Retail Week] <= 22
THEN “June”
ELSEIF [Retail Week] > 22 AND [Retail Week] <= 26
THEN “July”
ELSEIF [Retail Week] > 26 AND [Retail Week] <= 30
THEN “August”
```

```
ELSEIF [Retail Week] > 30 AND [Retail Week] <= 35
THEN “September”
ELSEIF [Retail Week] > 35 AND [Retail Week] <= 39
THEN “October”
ELSEIF [Retail Week] > 39 AND [Retail Week] <= 43
THEN “November”
ELSEIF [Retail Week] > 43 AND [Retail Week] <= 48
THEN “December”
ELSEIF [Retail Week] > 48 AND [Retail Week] <= 52
THEN “January”
END
// Retail Week of Month
IF [Retail Quarter Week] <= 4
THEN [Retail Quarter Week]
ELSEIF [Retail Quarter Week] > 4 AND [Retail Quarter Week] <= 9
THEN [Retail Quarter Week] - 4
ELSEIF [Retail Quarter Week] > 9 AND [Retail Quarter Week] <= 13
THEN [Retail Quarter Week] - 9
END
```

Having all of these calculations is extremely useful for any visualization that uses a retail calendar. Next you’ll create a visualization that I like to use to showcase the retail calendar.

Step 7: Build the base visualization

Add a discrete Retail Month of Quarter to columns. Add a discrete Retail Week of Month to rows and reverse the axis. Also add Retail Week of Month as a continuous dimension to rows. Add 0.0 as an ad hoc continuous

dimension to the right of Retail Week of Month. Create a synchronized dual axis.

Step 8: Add details

Step 8a: Bar Labels: Set the marks card of Retail Week of Month to a bar chart. Add Profit Ratio* (SUM(Profit)/SUM(Sales)) to color. Add SUM(Sales) and Retail Week to text. Format the text so it reads:

<SUM(sales) | Week <Retail Week>

Step 8a: 0.0 Labels: On the marks card of the 0.0 value set the mark type to Text. Add Retail Month to text.

Step 9: Dynamic Labels

You need to place a value on columns that will control both axes. Create a calculation called bar and add it to columns.

```
// bar

IF COUNTD([Retail Week]) = 1

THEN SUM([Sales])/WINDOW_MAX(SUM([Sales]))

ELSE .9

END
```

This calculation will show sales as a percentage of the maximum sales for a retail week for the bars and will place a label of each state at .9. Change the table calculation and select all values, as shown in Figure 4-32.

Figure 4-36. Table calculation settings for the 454 bar chart.

Set the axis range between 0 and 1.8, then hide the axis.

Step 10: Format

Finally, format according to your design standards.

The resulting visualization is shown in Figure 4-33.

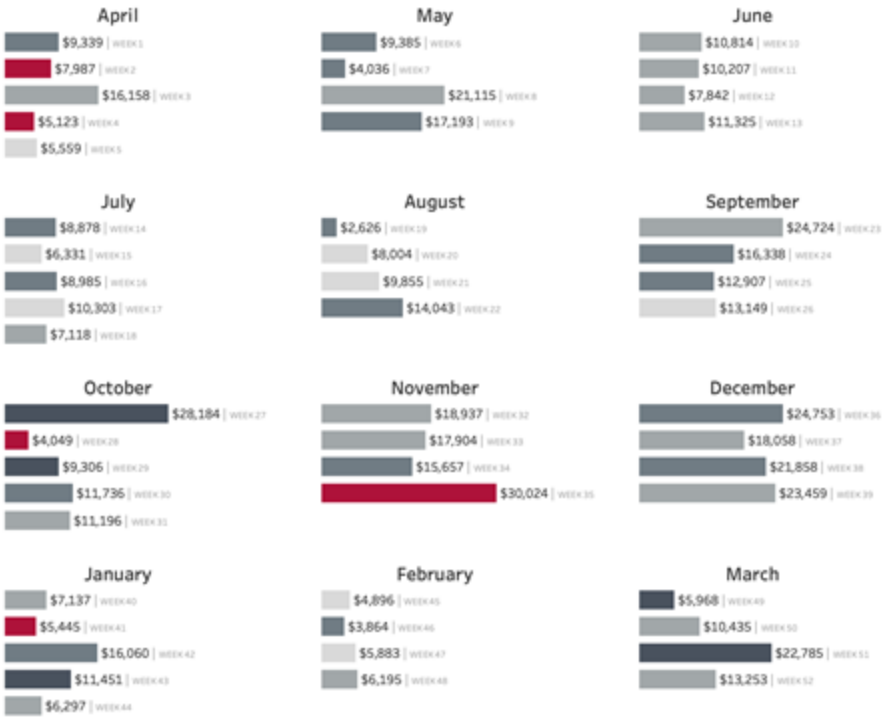


Figure 4-37. Bar chart showing sales and profit by 4-5-4 retail month and week.

Conclusion

In this chapter we've just scratched the tip of the iceberg of dates and times. Because dates and times are naturally hierarchical and can be treated both as dimensions or measures, there are nearly unlimited options for how to combine a single date field.

You learned the important difference between a date part and a date value. With date parts, Tableau returns a single component of a date or time field. The underlying calculation for date part is the DATEPART() function.

In the first two blueprints, you learned how to plot parts of a date, and how combining multiple date parts (in this case hour and minute) can lead to too many discrete values being shown. We also discussed how Tableau defaults date parts to dimensions and date values to continuous values. The final fundamental we discussed was Tableau making a date hierarchy available to your audience by default, which you can change. For most audiences, working with continuous axes is more intuitive because that's how people think about time!

You tackled the challenge of creating continuous axes from a single date part in the next three blueprints. You learned how (and why) to use line charts, bar charts, and heatmaps. The heatmap blueprints highlighted the flexibility of this chart type when working with date and time fields.

Another regular challenge with working with dates is comparing previous years to the current year to date. We showed you how to calculate the year-to-date values of any date, and how to use that inside aggregate calculations to create a proportional brush chart. We extended this same year-to-date calculation to adjust line charts to make fair comparisons.

We also looked at how to create automated reports by using both level of detail and table calculations, and how to use a table calculation to complete month-over-month and year-over-year calculations.

We wrapped up the chapter by talking about non-Gregorian calendars, such as fiscal years, and the retail 4-5-4 calendar.

In the next chapter, you'll learn about key performance indicators (KPIs): standalone metrics that businesses use to drive day-to-day operations or guide strategic decisions. If you become fluent with date fields in Tableau, creating dynamic, automated KPIs will come naturally.