



Don't Just Use Your Data... Exploit It

Technomics, Inc.

Adam H. James, Jeff Cherwonik, Brandon Bryant

2/25/2019

ICEAA 2019 Professional Development & Training Workshop

Revision 1 (3/31/2019)

Don't Just Use Your Data... Exploit It

Using data science techniques to better normalize, categorize, store, and analyze larger volume cost data

The Data Age is here. Data are being collected at an exponential rate and cost analysts are struggling to exploit it. Limited structured contextual information and clunky formatting are significant barriers to efficient use. This paper demonstrates the importance of establishing a modern data strategy that considers how analysts identify, access and use data in the context of an example exploitation of a large CCDR data set for the Army's Stryker Family of Vehicles.

ADAM H. JAMES, JEFF CHERWONIK, BRANDON BRYANT

TECHNOMICS, INC.

| | | |
|----------|----------------------------------------------|-----------|
| 1 | INTRODUCTION | 1 |
| 2 | WORKING WITH DATA | 3 |
| 2.1 | Storing and Structuring Data | 3 |
| 2.2 | Understanding a Grammar | 9 |
| 2.3 | Translating the Grammar into Excel | 15 |
| 3 | CCDR DATA | 19 |
| 3.1 | Working with the Data | 22 |
| 3.2 | Relationship to the Plan | 24 |
| 4 | STRYKER EXAMPLE | 26 |
| 4.1 | Research Questions | 26 |
| 4.2 | Project Workflow | 27 |
| 5 | CLOSING THOUGHTS | 34 |
| | APPENDIX A CELL FORMULATIONS IN EXCEL | 35 |
| | APPENDIX B SAMPLE TOOL OUTPUT | 37 |

1 Introduction

Cost analysts have historically operated on small data sets. Although identifying, collecting, and preparing the 'best' data for analysis can prove tedious and time consuming, the small volume of information made for a manageable effort to acquire, ready, and analyze the data. The combination of modern data management tools *and* an ever-increasing volume of centrally collected/stored cost (and to a lesser extent, technical, and programmatic) data represents a tremendous opportunity for the DoD cost analysis community and the various consumers of its analytical products.

This paper discusses an approach to normalize, categorize, and analyze data as applied to a large subset (1,000,000+ records) of the OSD Cost Assessment and Program Evaluation (CAPE) managed Contractor Cost Data Report (CCDR) dataset. The resulting product is a compact, usable database that can answer any question the data itself is capable of answering, without additional prep work. Example problems quickly answered (by virtue of a dynamic dashboard) include investigating labor rates over time, cost trends by WBS, and cost growth.

The methodologies presented are consistent with data science best practices – formalized by the popular “grammar of data manipulation” implemented in R through the collection of `tidyverse` packages.^{1,2} The goal is to communicate a flexible, scalable thought process which can be applied to the remainder of the CCDR library, as well as any other dataset.

Why are we doing this?

The motivation for this paper is a project initiated in fall 2017 by the Project Manager Stryker Brigade Combat Team (PM-SBCT). The cost team had amassed a large collection of CCDRs and was looking for new ways to exploit the data. More specifically, the goal of this project was to develop a Stryker-specific CCDR database and analysis tool to assist in the advanced validation and benchmarking of future data submissions.

The CCDR dataset employed for this effort is voluminous and inconsistent. Past CCDR-oriented research typically involved a limited data subset to target a specific, isolated question. This approach resulted in a never ending cycle of data cleansing and rework each and every time.

In the interest of future efficiencies, this project operated on the entire Stryker dataset for the purpose of normalizing the data for any/all future analytical purposes. As part of this forward-looking non-recurring effort, the work also established a consistent framework and structure to streamline data maintenance and analysis. While the primary goal of the work described in this paper was analysis of Stryker data, the workflow and data strategies were developed for use with any subset

¹ R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. <https://www.R-project.org/>

² <https://www.tidyverse.org/>

of the CCDR data available to cost analysts today. As such, virtually all the work discussed herein can be easily adapted to another program (or programs).

What's wrong with Excel?

Excel is often our tool of choice. Nearly all analysts are trained in its use and it is available on virtually all government workstations. While convenient, Excel has severe limitations. Working with data in it is actually quite difficult. It is very hard to produce repeatable workflows, and it struggles to keep up with even a moderately sized dataset. The software itself enforces no sense of consistency. As a result, Excel analyses tend to be “throw away” – built for a specific task but far too difficult to maintain and expand as a living solution.

What are the other options?

There are a variety of software tools on the market to work with data. An option growing rapidly in popularity is R, an open source project focused on statistical computing. For decades it lived as a niche product, but has exploded in popularity during the “data science” boom in the mid-2000s. R is able to handle larger sets of data much more easily than Excel. As a scripting language, it is easy to create repeatable, reproducible work.

This paper discusses best practices for working with data through the lens of R. However, the concepts are not specific to the software package itself. The topics discussed in this paper can be applied to any other software choice, including Excel.

2 Working with Data

All data analysis projects follow a series of steps. Practitioners across a wide variety of disciplines may argue the specifics, but a general workflow involves:

- (1) Research question formulation
- (2) Data collection
- (3) Data manipulation and cleaning (wrangling)
- (4) Exploratory data analysis
- (5) Formal analysis

This workflow can become very cyclical in nature. A specific research question may guide data collection, cleaning, and analysis. If the analysis yields unexpected results, then the process is restarted – hopefully reusing some of the original data.

It is very difficult to perfectly execute each step. However, a well-executed data collection followed by proper manipulation and cleaning creates a strong foundation for both exploratory and formal analyses. This foundational dataset should be able to support multiple research questions, including ones of the future.

The remainder of Section 2 discusses best practices for storing, structuring, and manipulating data. This is not an exhaustive list; it takes experience to fully grasp data manipulation best practices.

2.1 Storing and Structuring Data

It all starts with the data structure. More often than not, if you find yourself spending a lot of time reorganizing data before performing a task (e.g., summarizing data, creating a graphic) it is because your data are inefficiently structured. While some post processing will always be required, simple techniques exist to greatly ease data manipulation down the road.

Data should be convenient for the computer to work with. This is not necessarily natural and easy for a human to read! This is a critical distinction. Data are **stored** to be easy to use and manipulate. Data are **reported** to be easy to consume by a human – be it graphical, tabular, or another form.

Rows and Columns

It sounds obvious at first, but structured data are best organized in a tabular format. Think of the table as a collection of rows and columns.

Rows represent unique data observations. Each data row must be consistently defined. Deviation from this definition across rows must be avoided.

Columns represent variables. Variables are either hard coded data or calculations based on other variables. The consistent definition across all rows within a column allows for the use of vector operations. Vector operations require some practice at first, but are simpler (and more computationally efficient) in the long run.

Consider the following data structure in Table 1. This represents a reporting format, not a storage structure.

- (1) There is white space within the table in the form of blank cells. While easy to read, this creates problems recognizing data entries from a computer's perspective. This includes both blank rows and spacing in the System column.
- (2) Column definitions are inconsistent. Sometimes this is unavoidable, but often there are better ways to handle it. In this example we are mixing three ideas. There is cost data (Engine, Remaining, PM), quantity data (Number of Units), and then a calculation ($Cost \times Units$) for Total Cost.

Table 1: Example poor data structure

| System | WBS Element | Cost (\$) |
|---------|-----------------|-----------|
| Truck A | | |
| | Engine Cost | 50,000 |
| | Remaining Cost | 150,000 |
| | PM Cost | 1,500 |
| | Number of Units | 10 |
| | Total Cost | 2,015,000 |
| Truck B | | |
| | Engine Cost | 40,000 |
| | Remaining Cost | 120,000 |
| | PM Cost | 2,500 |
| | Number of Units | 5 |
| | Total Cost | 812,500 |

Table 2: Improved data structure

| System | Metric | Element | Value |
|---------|-----------|-----------------|--------|
| Truck A | Unit Cost | Engine | 50000 |
| Truck A | Unit Cost | Remaining | 150000 |
| Truck A | Unit Cost | PM | 1500 |
| Truck A | Unit Cost | Surface Vehicle | 201500 |
| Truck A | Quantity | Surface Vehicle | 10 |
| Truck B | Unit Cost | Engine | 40000 |
| Truck B | Unit Cost | Remaining | 120000 |
| Truck B | Unit Cost | PM | 2500 |
| Truck B | Unit Cost | Surface Vehicle | 162500 |
| Truck B | Quantity | Surface Vehicle | 5 |

Table 2 shows a much improved structure. At a quick glance this actually appears more difficult to read. The summary Total Cost is gone, which some may find to be valuable information. The repetition in System adds more text which can be distracting. However, this structure is much more convenient to work with computationally.

For larger applications, the structure shown across Table 3 and Table 4 may be even better. Duplication has now been removed. Row and column definitions are clear and consistent. Adding (appending) additional data is easy and the data tables are easily to manipulate to create a more desirable reporting table or graphic.

Table 3: Cost data table

| System | Element | Cost |
|---------|-----------|--------|
| Truck A | Engine | 50000 |
| Truck A | Remaining | 150000 |
| Truck A | PM | 1500 |
| Truck B | Engine | 40000 |
| Truck B | Remaining | 120000 |
| Truck B | PM | 2500 |

Table 4: Quantity data table

| System | Quantity |
|---------|----------|
| Truck A | 10 |
| Truck B | 5 |

Single Purpose Variables

Always create single purpose variables. A prime example of this is within a Work Breakdown Structure (WBS). It is very common to see a WBS with Model incorporated into the structure. In this case, the WBS now represents both a Model and a Cost Element. For example:

- 1 Surface Vehicle System
 - 1.1 Variant A
 - 1.1.1 Surface Vehicle
 - 1.1.1.1 Engine
 - 1.1.1.2 Remaining Vehicle
 - 1.2 Variant B
 - 1.2.1 Surface Vehicle
 - 1.2.1.1 Engine
 - 1.2.1.2 Remaining Vehicle

This structure is useful for reporting, but makes analysis painful. For 1.2.1.1 Engine, we must trace up to its grandparent element, 1.2, to know that it belongs to Variant B. This is not an easy task to do repeatedly within an analysis. A much better structure would be to move the Model out of the WBS into a column tag, as shown in Table 5.

Table 5: Single purpose WBS tagging

| Original WBS | Modified WBS | Element | Model |
|--------------|--------------|------------------------|-----------|
| 1 | 1 | Surface Vehicle System | |
| 1.1.1 | 1.1 | Surface Vehicle | Variant A |
| 1.1.1.1 | 1.1.1 | Engine | Variant A |
| 1.1.1.2 | 1.1.2 | Remaining Vehicle | Variant A |
| 1.2.1 | 1.1 | Surface Vehicle | Variant B |
| 1.2.1.1 | 1.1.1 | Engine | Variant B |
| 1.2.1.2 | 1.1.2 | Remaining Vehicle | Variant B |

This is not intended as an instruction on how to develop a WBS, but instead a demonstration of a preprocessing step to structure the data more effectively for analysis.

Other Data Tips

The following sections briefly overview some other topics to consider when structuring data.

Variable Names

Naming variables can be hard. Names should be descriptive, yet concise. Create variable names that:

- are unique (i.e., do not name two different columns the same thing)
- do not have special characters (except the underscore)
- do not have spaces
- start with a letter

Use underscores and/or CamelCase to string together multiple words. In general, use common sense and include a data dictionary or definition so that there are no questions as to what a specific variable represents.

Table 6: Example variable names

| Bad Name | Better Name |
|--------------------------|---------------|
| Work Breakdown Structure | WBS |
| % Complete | PercComp |
| Cost (TY \$K) | CostTY_K |
| Unit Cost (FY18) | UnitCost_FY18 |
| 1970 | Cost_1970 |

Ordered Records

Do not ascribe meaning to the physical sequencing of data. Data in a table are unordered. Data rows should be able to be shuffled randomly without any loss of information.

Define a variable to store any information that may be required to preserve the ordering of a dataset. Simply sort on that variable should something happen to the data and you need to restore or utilize the ordering.

Formatting

There is an important distinction between a data **value** and a data **format**. The value is the true representation of the data. The formatting is simply what is being displayed. Table 7 shows a few common examples.

Table 7: Values and formatting

| Value | Formatted |
|-------------|-----------------|
| 100000 | 100,000 |
| 2018-10-05 | October 5, 2018 |
| 2695.255648 | \$2,695.26 |

Best practice is to store unformatted data. Formatting is later applied during reporting. Excel's usefulness with formatting varies from convenient to highly problematic. Conveniently, Excel allows a cell to store a value, but display formatting. For example, a cell value may be 37284.345, but Excel will display 37,284.3 on the screen. No information is lost, and Excel retains the true cell value for calculations. Problematically, Excel sometimes "guesses" wrong (such as formatting WBS element

1.10 to 1.1) and exports formatted values that are not easily recognized by other software. This is discussed in more detail in the following section.

Data Types

Be sure to understand your variable data types. There are three generic data types:

- (1) numeric
- (2) date
- (3) text

Software developers break these down even further, but less formal applications are fine considering just these three. The following are brief notes on each data type.

- (1) **Numeric** data are any type of number; be it a {0, 1} field to represent true and false, an integer, or a number with decimals. A formatted number such as 1,000 is not a numeric field. It looks like a number, but the comma character is text. This is an issue particularly when exporting data as CSV from Excel.
- (2) **Date** data are actually a special case of numeric data. Computers store dates as the number of days from some start date (1/1/1900 for Excel). The start date is generally different for each software package. Therefore, the safest way to store a date is as a character string in the ISO date format: YYYY-MM-DD.³ This is an internationally recognized standard that any software will be able to interpret with certainty.
- (3) **Text** data are basically anything that is not a number. Any formatting of a numeric variable causes the data to be converted to text. Some software packages will automatically convert back to numeric for you, but most will not.

When in doubt, it is best to control data types on your own. In our cost analysis world, WBSs can be highly problematic. A WBS “number” is actually a text field. Most data analysis packages will handle this naturally by defining the entire variable as text. However, we must be careful in Excel.

| | A | B |
|---|-------------|----------------|
| 1 | Text Format | General Format |
| 2 | 1 | 1 |
| 3 | 1.1 | 1.1 |
| 4 | 1.1.1 | 1.1.1 |
| 5 | 1.1.2 | 1.1.2 |
| 6 | 1.2 | 1.2 |
| 7 | 1.9 | 1.9 |
| 8 | 1.10 | 1.1 |
| 9 | 1.11 | 1.11 |

Figure 1: WBS issues in Excel

Figure 1 exemplifies what occurs in Excel when you allow it to handle formatting. Column A shows a WBS number with the Number Format set to Text. Column B contains the same values, but with

³ <https://www.iso.org/iso-8601-date-and-time-format.html>

the General format selected. Excel adheres to standard convention of left justifying text and right justifying numbers, by default.

Clearly, this automatic formatting is a problem. Excel recognizes the WBS to be numeric, with the exception of 1.1.1 and 1.1.2, which are treated as text. In particular, 1.10 is now 1.1, as it dropped the trailing zero.

This is actually more problematic than it appears. The naive solution is simply to format any WBS column as text. Unfortunately, this fails. A simple `SUMIF()`, for example, will match 1.1 to both 1.1 and 1.10, summing values incorrectly. The following are some ways to mitigate this issue:

- Prefix the WBS with a letter (e.g., X1.1, X1.1.1, etc.) to force the structure into text
- Fill in a complete WBS to the lowest level with zeros (e.g., 1.0.0, 1.1.0, 1.1.1)
- Parse the WBS into multiple numeric fields (e.g., 1.2.3 becomes three fields: {1, 2, 3})
- Do not use Excel

Redundant Information

Avoid storing redundant data. Redundant data can be calculated from other data fields. Below are a few examples:

- (1) Only store the child element values of a WBS. The values for the parent tasks can easily be created by summing their children.
- (2) Do not store subtotals and totals. These can also be calculated very easily. A common occurrence is non-recurring and recurring costs. There is no need to store total cost in this case.
- (3) Do not store values that can be derived from a formula. There is no need to store density in a dataset that contains mass and volume, assuming the simple equation holds.

$$density = \frac{mass}{volume}$$

Intermediary Tables

Redundant information may actually be important and it is tedious and inefficient to constantly recreate it. Using example (3) from the prior section, your main variable of interest may be density. In this case, create intermediary tables as part of your workflow.

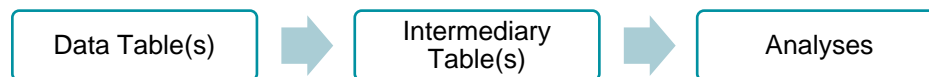


Figure 2: Intermediary table workflow

Assuming properly structured data tables, simple transformations (such as those described in the next section) easily and automatically create the intermediary tables. Base your analysis on these representations rather than the raw data for a clean, logical, and maintainable workflow. This strategy naturally segregates data from analysis, making the data more useful for alternative use cases.

Sparse Data

Data are considered sparse if many of the records have values equal to zero. The representation in Table 8 is considered the dense representation. We can reduce the amount of information stored by assuming that any value not referenced is in fact a zero.

Table 8: Dense table example

| wbs | recurring | non_recurring |
|-----|-----------|---------------|
| 1.1 | 0 | 0 |
| 1.2 | 0 | 0 |
| 1.3 | 1000 | 0 |
| 1.4 | 2000 | 0 |
| 1.5 | 0 | 0 |
| 1.6 | 0 | 0 |

The data in Table 9, Table 10, and Table 11 can be used to rebuild that in Table 8. While this does involve an extra step, creating these tables constitutes best practice in a database environment.

Under this representation, we only need to store two versus 12 data records.

Table 9: Sparse WBS

| id_wbs | wbs |
|--------|-----|
| 1 | 1.1 |
| 2 | 1.2 |
| 3 | 1.3 |
| 4 | 1.4 |
| 5 | 1.5 |
| 6 | 1.6 |

Table 10: Sparse categories

| id_rec | r_nr |
|--------|---------------|
| 1 | recurring |
| 2 | non_recurring |

Table 11: Sparse values

| id | id_wbs | id_rec | value |
|----|--------|--------|-------|
| 1 | 3 | 1 | 1000 |
| 2 | 4 | 1 | 2000 |

Storing data under a sparse model is not always appropriate. But in certain cases, it yields powerful results by greatly reducing the total volume of data.

2.2 Understanding a Grammar

Grammar, the rules to which we abide within a language, enables us to communicate in a structured, repeatable way – even if the meaning itself is nonsensical. While most known in the context of written and spoken language, grammars have become popular elsewhere. Two of the most well-known grammars in data science are the:

- (1) Grammar of Graphics
- (2) Grammar of Data Manipulation

gram·mar (noun)

The whole system and structure of a language or of languages in general, usually taken as consisting of syntax and morphology (including inflections) and sometimes also phonology and semantics.

- OED Online, Oxford University Press

These two grammars have been implemented with great success by Hadley Wickham and RStudio in the extremely popular R packages `ggplot2` and `dplyr`.⁴ The grammars provide a set of instructions, guided by a **noun** (i.e., a dataset) and a set of **verbs**. These concepts translate well to other languages and modeling tools, including Excel and Access. Section 4 demonstrates how we leveraged the Grammar of Data Manipulation to create a flexible, scalable framework to analyze Stryker cost data.

Grammar of Data Manipulation

Data manipulation is an essential step in any analytical environment. In R, “dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges”. The authors maintain the following primary verbs are sufficient for the vast majority of common data manipulation challenges:⁵

- `mutate()` – adds new variables that are functions of existing variables
- `select()` – picks variables based on their name
- `filter()` – picks cases based on their values
- `summarize()` – reduces multiple values down to a single summary
- `arrange()` – changes the ordering of rows

An additional verb that serves as a natural helper is:

- `group_by()` – perform any operation “by group”

Other secondary verbs (non-exhaustive) exist which are useful in other situations:

- `distinct()` – filter down to unique observations
- `spread()` – pivot data from a column into multiple columns
- `gather()` – consolidate data from multiple columns into a single column

Each operation is simple in itself, although it requires practice to frame more complex problems in this way. Those well versed in SQL will instantly notice the similarities. The following sections cover each primary verb in more detail with supporting examples.

Mutate

Mutate creates a new variable as a function of other variables. This can be a simple mathematical operation, such as multiplication, or a complex function with case logic (e.g., if-then criteria).

⁴ Hadley Wickham (<https://github.com/hadley>) is a Chief Scientist at RStudio, focusing on tool development to “make data science easier, faster, and more fun”. He is viewed as an expert and thought leader in the development of the open source software R, specifically in the areas of data manipulation and visualization.

⁵ <https://dplyr.tidyverse.org/>

$$\text{Example 1} \quad \text{cost_BY2020} = \frac{\text{cost_BY2010} \times \text{escalation}}{1000}$$

$$\text{Example 2} \quad \text{density} = \frac{\text{mass}}{\text{volume}}$$

$$\text{Example 3} \quad \text{size} = \begin{cases} \text{small,} & \text{weight} < 10000 \\ \text{large,} & \text{weight} \geq 10000 \end{cases}$$

Select

Select chooses which variables to include in an analysis. If cleverly implemented, you can derive a lot more power than simply grabbing a few columns of data. The obvious inverse of including a variable is excluding a variable. A more powerful extension is including/excluding variables based on a pattern.

Example 1 include variables: *fiscalyear, cost, hours, volume, mass*

Example 2 exclude variables: *volume, mass*

Example 3 include variables: starting with *FY*

The shading in Table 12 demonstrates the variables returned from a notional dataset based on the three examples above.

Table 12: Data columns returned by 'select'

| Example | fiscalyear | cost | hours | volume | mass | escalation | speed | FY2011 | FY2012 | FY2013 |
|---------|------------|------|-------|--------|------|------------|-------|--------|--------|--------|
| 1 | x | x | x | x | x | | | | | |
| 2 | x | x | x | | | x | x | x | x | x |
| 3 | | | | | | | | x | x | x |

Filter

Filter is a very simple operation as well. It subsets a dataset based on the values of one or more variables. This may be a specific level of a categorical variable, or a numerical value (or range) from a continuous variable.

Example 1 include observations: *engine = diesel*

Example 2 include observations: *fiscalyear ≥ 2000*

Example 3 include observations: *metric = hours*
include observations: *fiscalyear ≥ 2000*

The shading in the Ex (Example) 1-3 columns in Table 13, below, demonstrate the observations returned from a notional dataset based on the three examples above.

This dataset structures the metric of cost or hours into one column instead of two. This method allows us to filter the dataset based on either metric, and then any analysis follows easily. For

example, you may have a common visualization (such as a bar chart) which displays the total by year. In this case, it may be easier to set up one graphic and `filter` the rows to either cost or hours than to `select` a cost or hours columns.⁶

Table 13: Data rows returned by 'filter'

| Obs | Ex 1 | Ex 2 | Ex 3 | fiscalyear | metric | engine | value |
|-----|------|------|------|------------|--------|--------|-------|
| 1 | x | x | | 2010 | cost | diesel | 10 |
| 2 | | | | 1992 | cost | gas | 8 |
| 3 | x | x | | 2011 | cost | diesel | 2 |
| 4 | x | x | | 2006 | cost | diesel | 19 |
| 5 | x | x | x | 2008 | hours | diesel | 22 |
| 6 | x | | | 1989 | hours | diesel | 14 |
| 7 | x | x | x | 2016 | hours | diesel | 16 |
| 8 | | x | x | 2014 | hours | gas | 9 |

Distinctions such as this take practice and experience. However, the right structure can be the difference between a complex, slow, repetitive workflow and one that is streamlined.

Summarize

Summarize allows you to summarize data into an overall aggregation. This, in essence, turns multiple rows of data into a single consolidated row. Summarize will typically drop all variables from the output, except those which are relevant to the aggregation. For example, if you summarize by summing over an entire table, your result may have only one row named Total, and only the columns which you elected to sum over.

| | |
|-----------|------------------------------------------------------|
| Example 1 | count number of rows |
| Example 2 | sum over the rows |
| Example 3 | calculate a summary statistic (e.g., mean, variance) |

This may sound very elementary and uninteresting. However, it is a basic building block of any data manipulation and, when utilized with `group_by`, becomes much more powerful.

Arrange

Arrange sorts or otherwise logically orders the data. Most commonly this is a sort, but you can also abstract the concept into a more robust type of order. This would order the entire dataset based on some criteria set by one or more variables.

⁶ This is especially true when working in Excel. Filtering rows is much more natural in Excel than selecting columns in most cases.

| | |
|-----------|-------------------------------------------|
| Example 1 | sort ascending on a variable or variables |
|-----------|-------------------------------------------|

| | |
|-----------|--------------------------------------------|
| Example 2 | sort descending on a variable or variables |
|-----------|--------------------------------------------|

| | |
|-----------|------------------------------------------|
| Example 3 | order the data by the observation number |
|-----------|------------------------------------------|

Group_by

Grouping allows you to treat the dataset in blocks. This is very helpful when summarizing data by a categorical variable or when iterating over some criteria. Grouping may be done on a single variable, multiple variables, or some combination or calculation (best derived beforehand by `mutate`).

| | |
|-----------|------------------------------------------|
| Example 1 | group data by the metric (cost or hours) |
|-----------|------------------------------------------|

| | |
|-----------|-------------------------------|
| Example 2 | group data on the fiscal year |
|-----------|-------------------------------|

| | |
|-----------|------------------------------------------|
| Example 3 | group data on the engine type and metric |
|-----------|------------------------------------------|

The following example in Table 14 shows how groups can be divided based on Example 1. This has the effect of breaking the dataset into two blocks: cost data and hour data. Grouping, paired with the primary verbs described earlier, creates a very powerful way to work with data.

Table 14: Grouping in Example 1 based on metric

| Obs | Group | fiscalyear | metric | engine | value |
|-----|-------|------------|--------|--------|-------|
| 1 | 1 | 2010 | cost | diesel | 10 |
| 2 | 1 | 1992 | cost | gas | 8 |
| 3 | 1 | 2011 | cost | diesel | 2 |
| 4 | 1 | 2006 | cost | diesel | 19 |
| 5 | 2 | 2008 | hours | diesel | 22 |
| 6 | 2 | 1989 | hours | diesel | 14 |
| 7 | 2 | 2016 | hours | diesel | 16 |
| 8 | 2 | 2014 | hours | gas | 9 |

Putting it all Together

To this point, we have described the basic components of the grammar of data manipulation. Back to the English language, we can now build basic sentences:

- The boy ran.
- The cost analyst jumped.
- The cow mooed.

This is a rather simplistic way to communicate, but we can add complexity by combining statements:

- The boy ran and jumped.
- The cost analyst mooed and then swam to the moon.

While a bit silly and nonsensical, the second statement is grammatically correct. Keep this point in mind with data manipulations. A grammar will allow you to write and execute statements that simply

make no sense. Much like it is up to the author to write thoughtful, meaningful words, it is up to the analyst to construct useful and meaningful data manipulations.

A Simple Example

Consider the following vehicle dataset in Table 15.

Table 15: Vehicle data

| Obs | fiscalyear | metric | engine | value | escalation |
|-----|------------|--------|--------|-------|------------|
| 1 | 2010 | cost | diesel | 10 | 1 |
| 2 | 1992 | cost | gas | 8 | 0.8 |
| 3 | 2011 | cost | diesel | 2 | 1.02 |
| 4 | 2006 | cost | diesel | 19 | 0.96 |
| 5 | 2008 | hours | diesel | 22 | n/a |
| 6 | 1989 | hours | diesel | 14 | n/a |
| 7 | 2016 | hours | diesel | 16 | n/a |
| 8 | 2014 | hours | gas | 9 | n/a |

A common task would be to view average cost and hours, accounting for escalation. This breaks down into the following steps:

- (1) Divide the dataset into cost and hours
- (2) For each of the cost values, adjust for escalation
- (3) Calculate the average for the escalation adjusted costs
- (4) Calculate the average for the hours

In the grammar of data manipulation, this translates into the following pseudo code representation:

```
1: Vehicle data
2:   mutate value_adj =
3:     if (metric == cost) then (value / escalation)
4:     else value
5:   group_by metric
6:   summarize avg =
7:     mean of value_adj
```

This may look a bit odd and complex, but it really is a simple representation. Here is an explanation line-by-line.

```
1: Operate on the Vehicle dataset
2: Utilize a mutate to calculate a new variable: value_adj
3: If the metric is cost, divide value by escalation
4: If the metric is anything else, simply take the value
5: Group by the metric variable (cost and hours)
6: Utilize a summarize to output to variable named average
7: In this new variable, calculate the mean
```

The results from this example would look something like Table 16.

Table 16: Putting it all together results

| metric | avg |
|--------|-------|
| cost | 10.44 |
| hours | 15.25 |

Here, `summarize()` dropped everything except the results relevant to the aggregation. In this case, we requested in line 5 to `group_by()` the metric. So, we returned a row for each of the two groups. We then stated in line 7 that we wish to create a new column `avg` based on the `value_adj` column. Therefore, this is the only other column returned.

This pseudo code is very close to what actual code would look like in the `dplyr` package in R. Given a data frame (R's internal representation of a data table) named `vehicledata`, the following would return the result in Table 16 based on the data in Table 15:

```
vehicledata %>%  
  mutate(value_adj = if_else(metric == "cost", value / escalation, value)) %>%  
  group_by(metric) %>%  
  summarize(avg = mean(value_adj))
```

The point here is not to teach R, but to demonstrate how simple data manipulation problems can be with the right tool.

2.3 Translating the Grammar into Excel

This way of thinking works very well in a scripting language, such as R. But sometimes it is more practical (or the only choice) to use Excel. Fortunately, the grammar works quite well in this setting. The PivotTable is a natural extension to the grammar. It takes a structured, tabular input and outputs some summarization based on user input. Custom Excel sheets can be created using data structures and formulas which adhere to the grammar.⁷

Excel is very flexible in that it allows you to do almost anything you want with your data. However, this flexibility can also be the downfall of a large, complex data project. Some of the many drawbacks include:

- (1) Ability to mix data types within a column
- (2) Ability to name multiple columns the same thing
- (3) Ability to mix definitions (e.g., formulas) within a column
- (4) Slow calculation speeds
- (5) Inability to abstract repeated tasks into functions
- (6) Lack of ability to perform iteration

In essence, the main problem is **lack of consistency**. A practitioner can do anything they want, so Excel sheets tend to be unstructured and freeform – the opposite of what a grammar should represent.

⁷ This is not intended to serve as a training guide for advanced Excel modeling. Many concepts discussed here are elaborated on in extreme detail by a simple Google search. This paper assumes a strong familiarity with Excel and its common functions.

The Table Object

The word 'table' can be a bit ambiguous in the Excel world. Most users employ the term simplistically to describe a grouping of cells in which data are stored. While not wrong, a 'table' is also a distinct and powerful object within the Excel universe.

The Table object enforces several of our best practices for structuring data and has many benefits.⁸ Specifically, the Table object:

- can be named as an object, providing easy use and access
- requires each column to be uniquely named
- processes column operations more efficiently (and therefore faster)
- allows for reference to a column of data by name rather than row numbers
- suggests (but does not require) consistency within a column

The final bullet is very important for proper data management. A Table wants to treat a column with absolute consistency, but does allow for deviation if a user (perhaps foolishly) chooses to do so. For example, when a formula is entered into a cell in the table, Excel automatically applies it to the entire column (i.e., a consistent column definition). However, the user can then go back in and manually overwrite specific cells with custom values and/or different formula specifications.

The Table can be very helpful when dealing with data. The fact that you can refer to data by a vector (column) name instead of row numbers means formulas are able to scale more easily when data are added or subtracted. Further, it helps encourage good practice for data storage.

PivotTables

The following components make up a PivotTable, as seen in the Excel sidebar when you have it "Show Field List":

- PivotTable Fields – analogous to the `select` verb
- Filters – analogous to the `filter` verb
- Columns – analogous to the `group_by` helper verb
- Rows – analogous to the `group_by` helper verb
- Values – analogous to the `summarize` verb

The `arrange` verb is considered within the PivotTable itself, where you have the option to sort on columns, rows, and values. It is possible to `mutate`, or calculate values within the PivotTable, but it is often easier to add a new column to your input dataset.

Consider the same simple example, using the data in Table 15. Our first task is to `mutate` to adjust the value for escalation. In Excel, this is easy. We add a new variable in column **G** and name it

⁸ The Table object can be found under the INSERT menu in the Excel Ribbon.

'value_adj'. Starting in cell **G2**, we drag down the formula: `=IF(C2 = "cost", E2 / F2, E2)`. The result is shown in Figure 3.

| | A | B | C | D | E | F | G |
|---|-----|------------|--------|--------|-------|------------|-----------|
| 1 | Obs | fiscalyear | metric | engine | value | escalation | value_adj |
| 2 | 1 | 2010 | cost | diesel | 10 | 1 | 10 |
| 3 | 2 | 1992 | cost | gas | 8 | 0.8 | 10 |
| 4 | 3 | 2011 | cost | diesel | 2 | 1.02 | 1.960784 |
| 5 | 4 | 2006 | cost | diesel | 19 | 0.96 | 19.79167 |
| 6 | 5 | 2008 | hours | diesel | 22 | n/a | 22 |
| 7 | 6 | 1989 | hours | diesel | 14 | n/a | 14 |
| 8 | 7 | 2016 | hours | diesel | 16 | n/a | 16 |
| 9 | 8 | 2014 | hours | gas | 9 | n/a | 9 |

Figure 3: Excel example of 'mutate'

Now, we can add in a Pivot Table. Under the ROWS criteria, we add in 'metric'. This sets our `group_by` criteria. For VALUES, we add in 'value', and change the calculation from Sum to Average. This is equivalent to `summarize`. The resulting pivot table is shown in Table 17.

Table 17: Excel pivot table results for Vehicle data example

| Row Labels | Average of value_adj |
|--------------------|----------------------|
| cost | 10.43811275 |
| hours | 15.25 |
| Grand Total | 12.84405637 |

Here, Excel also provided a Grand Total line by default. Excel Pivot Tables are highly customizable and can be tuned to user liking.

Cell Formulas

While possible, Excel makes it very difficult to implement a flexible, traceable, repeatable data management solution. A simple set of instructions applied to a dataset (i.e., what the grammar constitutes) often requires a series of manual steps to implement within Excel.

Today

Effectively using cell formulas according to the grammar means the datasheet must be designed using sound foundations (see Section 2.1 Storing and Structuring Data). Appendix A contains a short discussion on each of the verbs. These methodologies work best when using with data stored in a Table.

Future

Excel continues to be an actively developed and rapidly advancing product. At the writing of this paper, several new functions are being tested as beta features.⁹ Following are a few highlights of the new functions as they pertain directly to the grammar.

- (1) `FILTER()` is an implementation of the *filter* primary verb.
- (2) `SORT()` (and `SORTBY()`) are implementations of the *arrange* primary verb.
- (3) `UNIQUE()` is an implementation of the *distinct* secondary verb.

While still not perfect, these pending additions demonstrate the recognition for an improvement in data management best practices within Excel.

⁹ <https://support.office.com/en-us/article/filter-function-f4f7cb66-82eb-4767-8f7c-4877ad80c759>

3 CCDR Data

The Contractor Cost Data Report (CCDR) is a data Contract Data Requirements List (CDRL) item for a population of DoD contracts related to acquiring and sustaining weapons and automated information systems. The report is largely unchanged since the 1960s and collects contractor cost and hours by:

- work breakdown structure
- functional category
- recurring/non-recurring breakout

OSD CAPE manages the collection and hosts the data in the Cost Assessment Data Enterprise (CADE) system.¹⁰

The CCDR itself is comprised of multiple data formats. The two primary formats are the Cost Data Summary Report (CDSR) and the Functional Cost-Hour Report (FCHR), known as the DD Forms 1921 and 1921-1, respectively. Table 18 highlights their content.

Table 18: DD Form 1921 and 1921-1 comparison

| Feature | 1921 | 1921-1 |
|-----------------------------------------------------|------|--------|
| report metadata (e.g., contractor, contract, dates) | ✓ | ✓ |
| cost (dollars) data | ✓ | ✓ |
| hours data | | ✓ |
| work breakdown structure breakout | ✓ | ✓ |
| recurring/nonrecurring breakout | ✓ | ✓ |
| contract summary (i.e., G&A, UB, MR, FCCM, fee) | ✓ | |
| functional category breakout | | ✓ |

This information is extremely useful, but the reports show their age. The forms are exactly that, forms. This medium is useful to view on the screen in Excel or in hard copy, but not much else. Both forms represent perfect examples of reporting formats, but they are unfortunately not storage formats. Analyzing CCDRs in their native medium is difficult, frustrating, and time consuming. Example 1921 and 1921-1 forms are shown in Figure 4 and Figure 5 on the following pages, respectively.

¹⁰ <https://cade.osd.mil/>

| COST DATA SUMMARY REPORT | | | | | | | | | | Form Approved OMB No. 0704-0188 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----------------------------------------------------------------------------------------------|------------------------------|-----------------------------------------------------------------------------------------------------------------|-------------|-----------------------------------------|
| The public reporting burden for this collection of information is estimated to average 8 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Executive Services Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR COMPLETED FORM TO THE ABOVE ORGANIZATION. | | | | | | | | | | |
| 1. PROGRAM a. MDAP: H-99 Powhatan b. PHASE: Production | | 2. PRIME MISSION PRODUCT H-99 Powhatan | | 3. CONTRACTOR TYPE (X one) <input checked="" type="checkbox"/> PRIME / ASSOCIATE <input type="checkbox"/> DIRECT-REPORTING SUBCONTRACTOR | | 4. NAME/ADDRESS (Include ZIP Code) Mystery, Inc. 5493 Jinkies Road, St. Louis, MO 61183 | | 5. APPROVED PLAN NUMBER A-09-Q-C5 | | |
| 6. CUSTOMER (DIRECT-REPORTING SUBCONTRACTOR USE ONLY) | | 7. CONTRACT TYPE FFP | | 8. CONTRACT PRICE \$458,680.4 | | 9. CONTRACT CEILING N/A | | 10. TYPE ACTION a. CONTRACT NO.: W84HSK-11-C-43' c. SOLICITATION NO.: b. LATEST MODIFICATION: d. NAME: | | |
| 11. PERIOD OF PERFORMANCE a. START DATE (YYYYMMDD): 20101210 b. END DATE (YYYYMMDD): 20150627 | | 12. APPROPRIATION <input type="checkbox"/> RDT&E <input checked="" type="checkbox"/> PROCUREMENT <input type="checkbox"/> O&M | | 13. REPORT CYCLE <input type="checkbox"/> INITIAL <input type="checkbox"/> INTERIM <input checked="" type="checkbox"/> FINAL | | 14. SUBMISSION NUMBER 2 | | 15. RESUBMISSION NUMBER 0 | | 16. REPORT AS OF (YYYYMMDD) 20130429 |
| 17. NAME (Last, First, Middle Initial) Scooby Doo | | 18. DEPARTMENT Business Operations | | 19. TELEPHONE NUMBER (Include Area Code) 555-314-2275 | | 20. EMAIL ADDRESS scooby.doo@mystery.inc.com | | 21. DATE PREPARED (YYYYMMDD) 20130628 | | |
| WBS ELEMENT CODE A | WBS REPORTING ELEMENTS B | NUMBER OF UNITS TO DATE C | COSTS INCURRED TO DATE | | | NUMBER OF UNITS AT COMPLETION G | COSTS INCURRED AT COMPLETION | | | |
| | | | NONRECURRING D | RECURRING E | TOTAL F | | NONRECURRING H | RECURRING I | TOTAL J | |
| 1.0 | H-99 Powhatan | 49.7 | \$0.0 | \$353,260.5 | \$353,260.5 | 50.0 | \$0.0 | \$365,202.3 | \$365,202.3 | |
| 1.1 | Air Vehicle (AV) | 49.7 | \$0.0 | \$352,152.8 | \$352,152.8 | 50.0 | \$0.0 | \$364,126.6 | \$364,126.6 | |
| 1.1.1 | Airframe | 49.7 | \$0.0 | \$308,794.5 | \$308,794.5 | 50.0 | \$0.0 | \$323,654.7 | \$323,654.7 | |
| 1.1.2 | Propulsion | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.1.3 | AV Applications Software | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.1.4 | AV System Software | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.1.8 | Fire Control | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.1.9 | Data Display and Controls | 49.7 | \$0.0 | \$39,108.6 | \$39,108.6 | 50.0 | \$0.0 | \$35,903.2 | \$35,903.2 | |
| 1.1.15 | Armament | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.1.16 | Weapons Delivery | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.1.17 | Auxiliary Equipment | 49.7 | \$0.0 | \$0.0 | \$0.0 | 50.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.1.18 | Crew Station | 49.7 | \$0.0 | \$4,249.7 | \$4,249.7 | 50.0 | \$0.0 | \$4,568.7 | \$4,568.7 | |
| 1.2 | System Engineering/Program Management | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.3 | System Test and Evaluation | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.4 | Training | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.5 | Data | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.6 | Peculiar Support Equipment | 0.0 | \$0.0 | \$1,107.7 | \$1,107.7 | 0.0 | \$0.0 | \$1,075.7 | \$1,075.7 | |
| 1.7 | Common Support Equipment | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.8 | Operational/Site Activation | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.9 | Industrial Facilities | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| 1.10 | Initial Spares and Repair Parts | 0.0 | \$0.0 | \$0.0 | \$0.0 | 0.0 | \$0.0 | \$0.0 | \$0.0 | |
| | Subtotal Cost | | | | \$353,260.5 | | | | \$365,202.3 | |
| | Reporting Contractor G&A | | | | \$42,391.3 | | | | \$43,824.3 | |
| | Reporting Contractor Undistributed Budget | | | | | | | | \$0.0 | |
| | Reporting Contractor Management Reserve | | | | | | | | \$0.0 | |
| | Reporting Contractor FCCM | | | | \$2,022.7 | | | | \$2,022.7 | |
| | Total Cost | | | | \$397,674.5 | | | | \$411,049.3 | |
| | Reporting Contractor Profit/Loss or Fee | | | | \$33,197.4 | | | | \$47,631.1 | |
| | Total Price | | | | \$430,871.9 | | | | \$458,680.4 | |
| 22. REMARKS | | | | | | | | | | |

Figure 4: CDSR Report (DD Form 1921)

| FUNCTIONAL COST-HOUR REPORT | | | | | | Form Approved OMB No. 0704-0188 | |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|-------------------------------------------------------|--|------------------------------------------------------------------------------------------------------------------------------------------------|--------------|--------------------------------------------------------------------------------------------------------------|-------|
| The public reporting burden for this collection of information is estimated to average 16 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Executive Services Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. | | | | | | | |
| 1. PROGRAM a. MDAP: b. PHASE: Production | | 2. PRIME MISSION PRODUCT H-99 Powhatan | | 3. CONTRACTOR TYPE (X One) <input checked="" type="checkbox"/> PRIME / ASSOCIATE <input type="checkbox"/> DIRECT-REPORTING SUBCONTRACTOR | | 4. NAME/ADDRESS (Include Zip Code) Mystery, Inc., 5493 Jinkies Road, St. Louis, MO 61183 | |
| 5. APPROVED PLAN NUMBER A-09-Q-C5 | | 6. CUSTOMER (Direct-Reporting Subcontractor Use Only) | | | | 7. TYPE ACTION a. CONTRACT NO.: W84HSK-11-C-4311 c. SOLICITATION NO.: b. LATEST MODIFICATION: d. NAME: | |
| 8. PERIOD OF PERFORMANCE a. START DATE (YYYYMMDD): 20101210 b. END DATE (YYYYMMDD): 20150627 | | | | 9. REPORT CYCLE <input type="checkbox"/> INITIAL <input type="checkbox"/> INTERIM <input checked="" type="checkbox"/> FINAL | | 10. SUBMISSION NUMBER 2 | |
| | | | | 11. RESUBMISSION NUMBER 0 | | 12. REPORT AS OF (YYYYMMDD) 20130429 | |
| 13. NAME (Last, First, Middle Initial) Scooby Doo | | 14. DEPARTMENT Business Operations | | 15. TELEPHONE NO. (Include Area Code) 555-314-2275 | | 16. EMAIL ADDRESS scooby.doo@mystery.inc.com | |
| 17. DATE PREPARED (YYYYMMDD) 20130628 | | 18. WBS ELEMENT CODE 1.0 | | 19. WBS REPORTING ELEMENT H-99 Powhatan | | 20. NUMBER OF UNITS a. TO DATE: 49.7 b. AT COMPLETION: 50.0 | |
| | | | | 21. APPROPRIATION <input type="checkbox"/> RDT&E <input checked="" type="checkbox"/> PROCUREMENT <input type="checkbox"/> O&M | | | |
| FUNCTIONAL DATA ELEMENTS | | | | COSTS AND HOURS INCURRED TO DATE | | | |
| | | | | A. NONRECURRING | B. RECURRING | C. TOTAL | |
| | | | | COSTS AND HOURS INCURRED AT COMPLETION | | | |
| | | | | D. NONRECURRING | E. RECURRING | F. TOTAL | |
| ENGINEERING | | | | | | | |
| (1) DIRECT ENGINEERING LABOR HOURS | | | | 0.0 | 26.5 | 26.5 | 0.0 |
| (2) DIRECT ENGINEERING LABOR DOLLARS | | | | \$0.0 | \$3,066.5 | \$3,066.5 | \$0.0 |
| (3) ENGINEERING OVERHEAD DOLLARS | | | | \$0.0 | \$898.5 | \$898.5 | \$0.0 |
| (4) TOTAL ENGINEERING DOLLARS | | | | \$0.0 | \$3,965.1 | \$3,965.1 | \$0.0 |
| MANUFACTURING OPERATIONS | | | | | | | |
| (5) DIRECT TOOLING LABOR HOURS | | | | 0.0 | 0.0 | 0.0 | 0.0 |
| (6) DIRECT TOOLING LABOR DOLLARS | | | | \$0.0 | \$0.0 | \$0.0 | \$0.0 |
| (7) DIRECT TOOLING & EQUIPMENT DOLLARS | | | | \$0.0 | \$0.0 | \$0.0 | \$0.0 |
| (8) DIRECT QUALITY CONTROL LABOR HOURS | | | | 0.0 | 0.0 | 0.0 | 0.0 |
| (9) DIRECT QUALITY CONTROL LABOR DOLLARS | | | | \$0.0 | \$0.0 | \$0.0 | \$0.0 |
| (10) DIRECT MANUFACTURING LABOR HOURS | | | | 0.0 | 1,271.3 | 1,271.3 | 0.0 |
| (11) DIRECT MANUFACTURING LABOR DOLLARS | | | | \$0.0 | \$46,626.0 | \$46,626.0 | \$0.0 |
| (12) MANUFACTURING OPERATIONS OVERHEAD DOLLARS (Including Tooling and Quality Control) | | | | \$0.0 | \$71,494.4 | \$71,494.4 | \$0.0 |
| (13) TOTAL MANUFACTURING OPERATIONS DOLLARS (Sum of rows 6, 7, 9, 11, and 12) | | | | \$0.0 | \$118,120.4 | \$118,120.4 | \$0.0 |
| MATERIALS | | | | | | | |
| (14) RAW MATERIAL DOLLARS | | | | \$0.0 | \$0.0 | \$0.0 | \$0.0 |
| (15) PURCHASED PARTS DOLLARS | | | | \$0.0 | \$0.0 | \$0.0 | \$0.0 |
| (16) PURCHASED EQUIPMENT DOLLARS | | | | \$0.0 | \$222,655.4 | \$222,655.4 | \$0.0 |
| (17) MATERIAL HANDLING/OVERHEAD DOLLARS | | | | \$0.0 | \$6,853.9 | \$6,853.9 | \$0.0 |
| (18) TOTAL DIRECT-REPORTING SUBCONTRACTOR DOLLARS | | | | \$0.0 | \$0.0 | \$0.0 | \$0.0 |
| (19) TOTAL MATERIAL DOLLARS | | | | \$0.0 | \$229,509.3 | \$229,509.3 | \$0.0 |
| OTHER COSTS | | | | | | | |
| (20) OTHER COSTS NOT SHOWN ELSEWHERE (Specify in Remarks) | | | | \$0.0 | \$1,665.8 | \$1,665.8 | \$0.0 |
| SUMMARY | | | | | | | |
| (21) TOTAL COST (Direct and Overhead) | | | | \$0.0 | \$353,260.5 | \$353,260.5 | \$0.0 |
| 22. REMARKS | | | | | | | |

Figure 5: FCHR Report (DD Form 1921-1)

3.1 Working with the Data

The forms are useful as a reporting format, but inadequate for storage and analysis. Moreover, the problem is exacerbated when attempting to work with multiple reports concurrently. Section 2.1, Storing and Structuring Data, addressed best practices for storing data. The following points highlight deficiencies with the 1921 and 1921-1 with these best practices in mind.

Rows and Columns

Metadata is not reported in a tabular structure. Aggregating metadata from multiple reports requires tediously extracting each field one at a time. This process can be automated through code, but is error prone for older reports due to format inconsistencies in the Excel files.

Data are displayed in a pivoted view. Each row contains multiple records instead of one. Consider the example in Table 19. A single row in the 1921 represents eight values, i.e., the quantity ('to date' and 'at completion') and cost ('to date' and 'at completion' by non-recurring, recurring, and total).

Table 19: Native 1921 data record format

| WBS Code | WBS Elements | To Date Values | | | | At Completion Values | | | |
|----------|--------------|----------------|-------|-------------|-------------|----------------------|-------|-------------|-------------|
| | | Units | NR | R | Total | Units | NR | R | Total |
| 1.1 | Airframe | 49.7 | \$0.0 | \$352,152.8 | \$352,152.8 | 50.0 | \$0.0 | \$364,126.6 | \$364,126.6 |

A more effective way to store the data is shown in Table 20. This tabular structure is much more convenient to work with under the grammar of data manipulation. The verbs *filter* and *summarize* with a careful use of *group_by* will answer most questions. Further, the tabular representation can be easily manipulated to mimic the original format.

Table 20: Improved 1921 data record format

| wbs_code | wbs_element | td_ac | r_nr | unit | value |
|----------|-------------|---------------|--------------|----------|----------|
| 1.1 | Airframe | To Date | Nonrecurring | Cost | 0.0 |
| 1.1 | Airframe | To Date | Recurring | Cost | 352152.8 |
| 1.1 | Airframe | To Date | Total | Cost | 352152.8 |
| 1.1 | Airframe | To Date | Total | Quantity | 49.7 |
| 1.1 | Airframe | At Completion | Nonrecurring | Cost | 0.0 |
| 1.1 | Airframe | At Completion | Recurring | Cost | 364126.6 |
| 1.1 | Airframe | At Completion | Total | Cost | 364126.6 |
| 1.1 | Airframe | At Completion | Total | Quantity | 50.0 |

Formatting is stripped from the values. Variable names are short, concise, and “software friendly” in that they do not violate the rules for variable names outside of Excel. This table can be further reduced by creating a separate table for quantities, as demonstrated in Table 3 and Table 4.

The CADE system will return data similar to this format for some but not all CCDR reports.¹¹ Minor cleanup is required, but the cleansed version resembles best practice much more closely than the Excel forms.

Redundant Information

The 1921 and 1921-1 contain duplicative information, both within a single form and between the two forms. Consider the following remedies:

- (1) The 1921-1 data are simply a more granular representation of the 1921 data. Store the 1921-1 data when available. When unavailable, store the total line from the 1921. Do this within the context of the 1921-1 structure so that everything is consistent.
- (2) Remove redundant records. Several columns and rows exist in the 1921-1 that are totals of other columns and rows (e.g., 'C. Total', '(4) Total Engineering Dollars'). This would include rows 3 and 7 in Table 20.

Sparse Data

The CCDR data are sparse by nature. The 1921-1 form is structured as follows:

- 21 functional elements
- 2 time periods (to date and at completion)
- 3 metric "type" fields (nonrecurring, recurring, and total)

This yields $21 \times 2 \times 3 = 126$ total data records per form. If we eliminate the redundant items (four rows and 2 columns), we still have 68 total data records per 1921-1.

The sample 1921-1 in Figure 5 has non-zero data in only 52 of the 126 possible records. The means that 59% of the records are zero. Counting again with redundant items removed, only 18 of the 68 possible records have data. Now we have a whopping 74% of the data records being zero. That is a lot of information to unnecessarily store. Suppose the following is true for an example CCDR:

- 50 WBS elements
 - 40 of which are at the lowest level
 - 20 at the lowest level which have no cost
- Each WBS has the same level of sparsity as in Figure 5

The data are to follow a similar structure as demonstrated in Table 20, but ignoring quantity and adding in a column for the functional element. This makes for quick math to determine the total number of data rows.

$$\begin{aligned} rows_{all} &= 50 \times 21 \times 2 \times 3 \\ &= 50 \times 126 \end{aligned}$$

¹¹ All data are obtainable in the "legacy" report format – the Excel form view. Recent, current, and future data are reported compliant to an XML schema which facilitates ingestion into the system's relational database. In these cases, use of the "cross program export" will return the tabular structure. Reconciling the two sources is not trivial.

$$= 6300$$

Now follow best practice to eliminate duplication.

$$\begin{aligned} rows_{dense} &= 20 \times 17 \times 2 \times 2 \\ &= 50 \times 68 \\ &= 1360 \end{aligned}$$

Finally, utilize the sparse storage model.

$$\begin{aligned} rows_{sparse} &= 20 \times 18 \\ &= 360 \end{aligned}$$

The results are clear. Storing the entirety of the cost report requires 6,300 data records. Adhering to best practices and utilizing a sparse storage model reduces this by 94% to a mere 360 records, with **no loss of information**. Applying this across a collection of CCDRs yields significant benefits in storage requirements and computational efficiency.

3.2 Relationship to the Plan

The content and periodicity of a CCDR for a particular contract for a particular program is specified in a well-defined document, the CSDR Plan.¹² In short, the Plan is the mechanism to communicate the government reporting requirements for CCDRs in an RFP. Two primary functions of the Plan are to:

- (1) Set the timeline of CCDR submission to the government
- (2) Define the WBS for use in each CCDR submission

Figure 6 conceptualizes these relationships. A Plan has one or more “Submission Events”, or points at which industry is required to deliver a CCDR. Each CCDR identifies cost according to the WBS specified in the Plan.

Various factors such as changes to policy (MIL-STD updates), data requirement needs, and data reporting capability result in an ever-evolving WBS. This requires additional effort to normalize between reports for comparison and model building.

¹² Together, CCDRs and another significant data product, Software Resource Data Reports (SRDRs), constitute what's called the Cost and Software Data Reporting (CSDR) process. CSDR Plans created by the Cost Working Integrated Product Team (CWIPT) for a given defense program specify the breadth and depth of data industry is required to deliver for a particular contract.

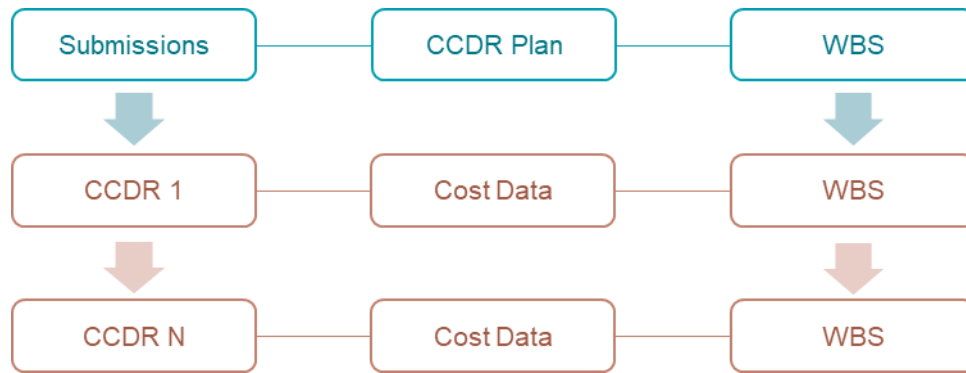


Figure 6: Plan and CCDR relationship

The WBS may vary as the plan is revised, but should ultimately result in a high degree of commonality. This fact provides a convenient avenue for dimension reduction in a normalization step. Since a Plan has multiple reports, applying WBS normalizations at the Plan level has a cascading effect on each report. Thus, fewer WBSs may be operated on to achieve the same results.

4 Stryker Example

The prior sections of this paper discussed data storage and structuring techniques, the grammar of data manipulations, and the CCDR data source. We build upon this foundational knowledge and skillset to demonstrate a powerful example on the Stryker Family of Vehicles (FoV), or simply “Stryker”.

The Stryker program is managed by Project Manager Stryker Brigade Combat Team (PM-SBCT) under PEO GCS out of TACOM in Warren, Michigan.¹³ Stryker is a large, 8-wheeled combat vehicle. There are ten distinct variants (M1126 – M1135), which as a group have evolved over four platform evolutions. Figure 7 depicts an early generation Infantry Fighting Vehicle (IFV).¹⁴ The Army has purchased and maintained thousands of vehicles from the early 2000s to present and continues to procure vehicles today.



Figure 7: Stryker ICV (M1126)

During this time, the Army has amassed one the largest collection of CCDRs for a single program in DoD. However, this data has not been utilized to its full potential because it is:

- Too voluminous to handle within spreadsheets
- Too diverse to quickly understand
- Ever growing (further amplifying the problem)

A project was initiated in fall 2017 with the goal to **develop a Stryker-specific CCDR database and analysis tool to assist in the advanced validation and benchmarking of future data submissions.**

4.1 Research Questions

The overarching research question revolved around the ability to develop trends to validate and benchmark future data submission. A more specific set of initial questions were:

- (1) How do profit rates vary across delivery orders?
- (2) What are the overhead rates and how have they trended over time?
- (3) How does actual cost/hours grow across the WBS from the initial submission to the final submission?
- (4) How does per unit cost/hours change over time across major WBS product areas?
- (5) How does the level of effort change over time for select contractor support areas?

These specific questions guided what data to collect and how to normalize it.

¹³ <https://peogcs.army.mil/sbct.html>

¹⁴ https://commons.wikimedia.org/wiki/File:Stryker_ICV_front_q.jpg

4.2 Project Workflow

The project was built upon four distinct enabling items, or “task building blocks” (Figure 8). The following sections discuss each, highlighting how we applied the best practices introduced in Section 2.

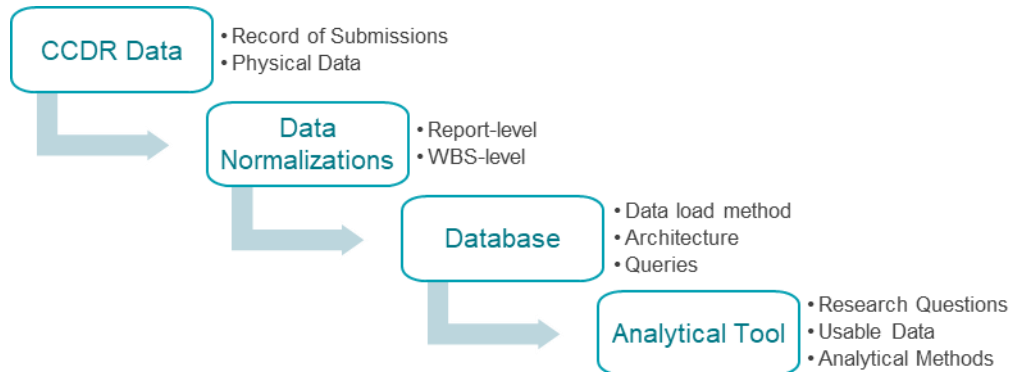


Figure 8: Task building blocks

CCDDR Data

The data source for this project was CCDDR data. At the time, Stryker had 207 reports – well over half of the population of CCDDRs for the entire surface vehicle commodity class. Figure 9 shows the number of reports submitted over time, by data format. The CCDDRs were cross-referenced and mapped to their respective CCDDR Plans. This linkage provided two functions:

- (1) Served as a “checklist” to ensure all records were accounted for
- (2) Reduced size from 207 reports to only 40 plans

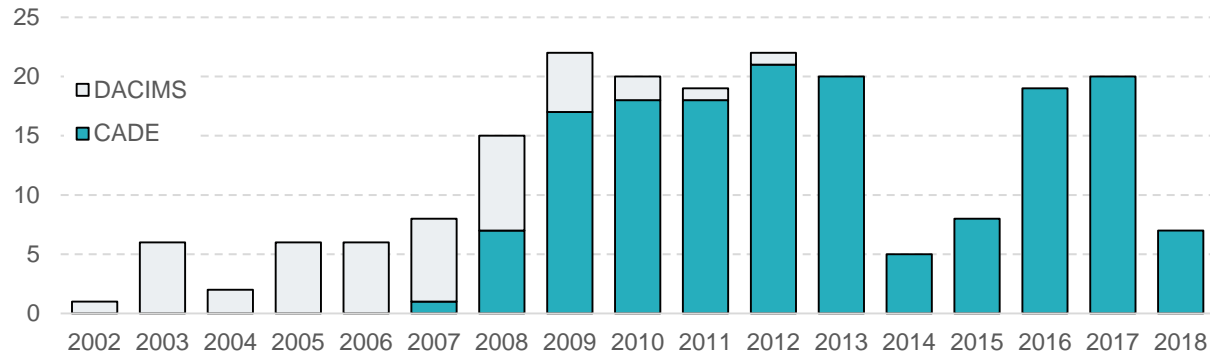
Item (2) is directly relevant to the data normalization step. Specifically, this allowed us to map only 40 WBSs to a common structure rather than 207.

For consistency, the legacy Excel forms were transformed into the CADE tabular format, similar to Table 20. This adheres to our **rows and columns** best practice and allows for all data to be treated identically throughout the remainder of the workflow.

We next applied repeatable, automated scripts to the data to remove **redundant information**. As discussed in Section 3.1, we removed the unnecessary 1921/1921-1 fields and implemented a sparse storage model. For the first data subset obtained from CADE, this reduced the data from 1,294,154 rows of data down to only 94,921 (7.3% of the original size).¹⁵ While Excel is certainly not our tool of choice, the reduced 94,921 data rows is certainly manageable within a single workbook.

Finally, we stripped all **formatting** and transformed dates to the standard ISO representation.

¹⁵ The 32-bit version of Excel (which nearly all users are running) is limited to 1,048,576 rows within a single sheet.

Figure 9: Stryker CCDR reports by format over time¹⁶

Data Normalizations

Data normalizations are required for the data to be useful. The data normalization strategy for this project operated at two levels: Report-level and WBS-level normalizations.

Report-level Normalizations

The report-level normalizations allow for both a human and a computer to effectively locate a relevant report. The following list contains summarizes the implemented strategy:

- (1) Metadata tagging
 - a. Phase (Development, Production, Sustainment)
 - b. SubPhase (Name, Start, End, and Description)
 - c. EndItem (Ground Vehicle, Vehicle Kits)
 - d. Process (Manufacturing, Maintenance, Systems Engineering, etc.)
 - e. ReportSequence (ID, Description, Type)
 - f. DateEvent
- (2) Additional True/False codes are added to assist in “selecting the right report”
 - a. Redundant
 - b. Prior
 - c. Mapped
- (3) Escalation according to OSD published indices
- (4) Additional fields are available natively through the metadata that are useful
 - a. Contract Number
 - b. Delivery Order
 - c. Plan Number

All items on this list are important. However, the metadata tagging structure presented in item (1) is what truly enables the grammar to work. These consistent and carefully structured fields allow for filtering down to the “right” data and proper sequencing of the data (e.g., initial and final reporting,

¹⁶ The DACIMS label represents reports stored as Excel file in the legacy data repository, Defense Automated Cost Information System. The CADE label represents reports available as a more modern flat file out of the CADE system.

lot sequencing). Many of the tags are intuitive from the name, with the following warranting a bit more explanation.

SubPhase defines a production lot sequence. Table 21 shows an example population of the four SubPhase fields, which relate to either Low Rate Initial Production (LRIP) or Full Rate Production (FRP) lot data.

Table 21: SubPhase examples

| SubPhase_Name | SubPhase_Start | SubPhase_End | SubPhase_Description |
|---------------|----------------|--------------|----------------------|
| LRIP | 1 | 1 | LRIP Lot 1 |
| LRIP | 2 | 2 | LRIP Lot 2 |
| FRP | 1 | 1 | FRP Lot 1 |
| FRP | 2 | 2 | FRP Lot 2 |
| FRP | 3 | 5 | FRP Lots 3-5 |

ReportSequence defines sequencing of reports within a singular scope. More simply put, ReportSequence uniquely groups a time series of report submissions (e.g., initial, interim, and final). Table 22 shows an example population of the three ReportSequence fields. The first three rows are an initial, interim, and final report for Lot 2 Production. The last three rows are all final reports for annual Program Management.

Table 22: ReportSequence examples

| ReportSeq_ID | ReportSeq_Description | ReportSeq_Type |
|--------------|---------------------------|----------------|
| 1 | FRP Lot 2 Production | Cumulative |
| 1 | FRP Lot 2 Production | Cumulative |
| 1 | FRP Lot 2 Production | Cumulative |
| 2 | Project Management FY2010 | Discrete |
| 2 | Project Management FY2011 | Discrete |
| 2 | Project Management FY2012 | Discrete |

A 'Cumulative' sequence is one where the total costs are found by taking only the most recent report. A 'Discrete' sequence is one where the total costs are found by summing each report within the sequence.

DateEvent is simply a tag to describe the nature of a given report. In the Table 22 example, for the first three rows the DateEvent tags may be 'initial', 'interim', and 'final', respectively. In some cases information such as 'Contract Award' or 'Purchase Year 1' may be appropriate to store.

WBS-level Normalizations

The WBS-level normalizations allow for both a human and a computer to effectively locate a relevant data record. This step is critical ensures that the WBS is a **single purpose variable**. The native representation ascribes multiple purposes to the WBS field. The following list contains the normalizations applied to each WBS element.

- (1) Map to a common WBS
 - a. Surface Vehicle (product based)
 - b. Vehicle Kits (product based)

- c. Support Services (process based)
- d. Sustainment (process based)
- (2) Variant/Model
- (3) Platform Generation
- (4) Kit Name
- (5) Funding Source

The two primary WBS-level normalizations are items (1) and (2). These allow for the by WBS and by Model analyses.

Database

Data are stored in a relational data model. For this project, Microsoft Access was selected for its ease of use on the desktop. For the sake of repeatability and maintainability, scripts were developed to load the consistent, normalized data into the database. Figure 10 demonstrates the conceptual database relationships. The main takeaways are:

- Normalizations, plans data, and CCDR data have natural boundaries for database maintainability and to enable reuse for other purposes
- Report-level normalizations map to submissions which map to reports
- WBS-level normalizations map to plan WBSs which map report WBSs

The database was created using best practices for **variable names** and **data types**. Further, no meaning is ascribed to the **ordering** of the records. Any desired sorting can be performed utilizing the normalization structure.

All data is extracted from the database through three primary queries. These queries are flat representations of the Cost Data, tagged and normalized report metadata (Submission Events), and the tagged and normalized WBS structure (Plan WBS). These queries serve as the input for any analysis and/or tool development. By keeping things simple, updates to the data (and subsequently the database) flow seamlessly into the downstream analytical products.

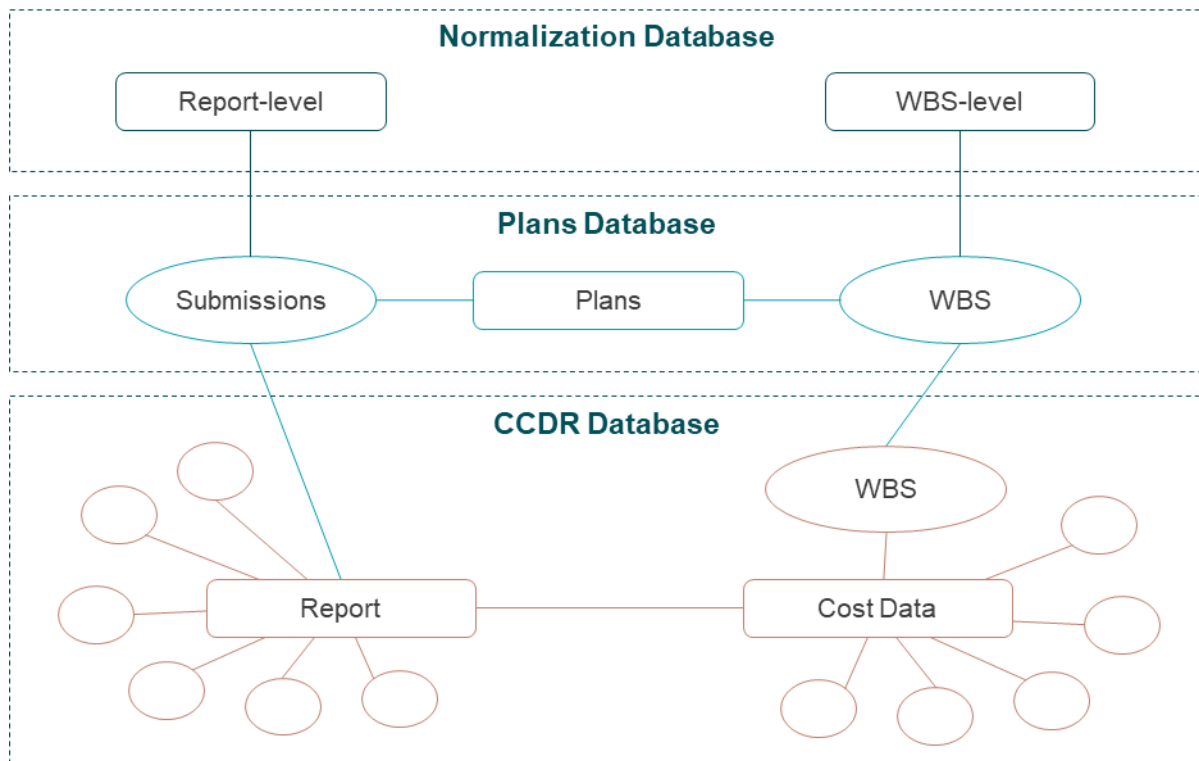


Figure 10: Conceptual database relationships

Analytical Tool

The analytical tool is where all of the data preparation work finally pays off. The analytical tool is an Excel workbook with dynamic dashboards intended to answer the five research questions in Section 4.1. However, we did not want to be limited to these questions. Instead, the goal was to create an Excel tool framework capable of answering a variety of research questions and problems. In this framework we would answer the original questions.

While this may sound complicated, the upfront work and concept of the grammar makes working in this way fairly simple. As a whole, the tool must be maintainable. It must be robust enough such that new data can be added to it seamlessly. Further, it must be expandable to address new questions. Additionally, each research question requires a few considerations:

- (1) What data do I need to answer the question?
- (2) What variations of the question exist?
- (3) How do I best present the data and results?

With this foresight, the tool development is broken out into a few steps. This process flow is shown in Figure 11.

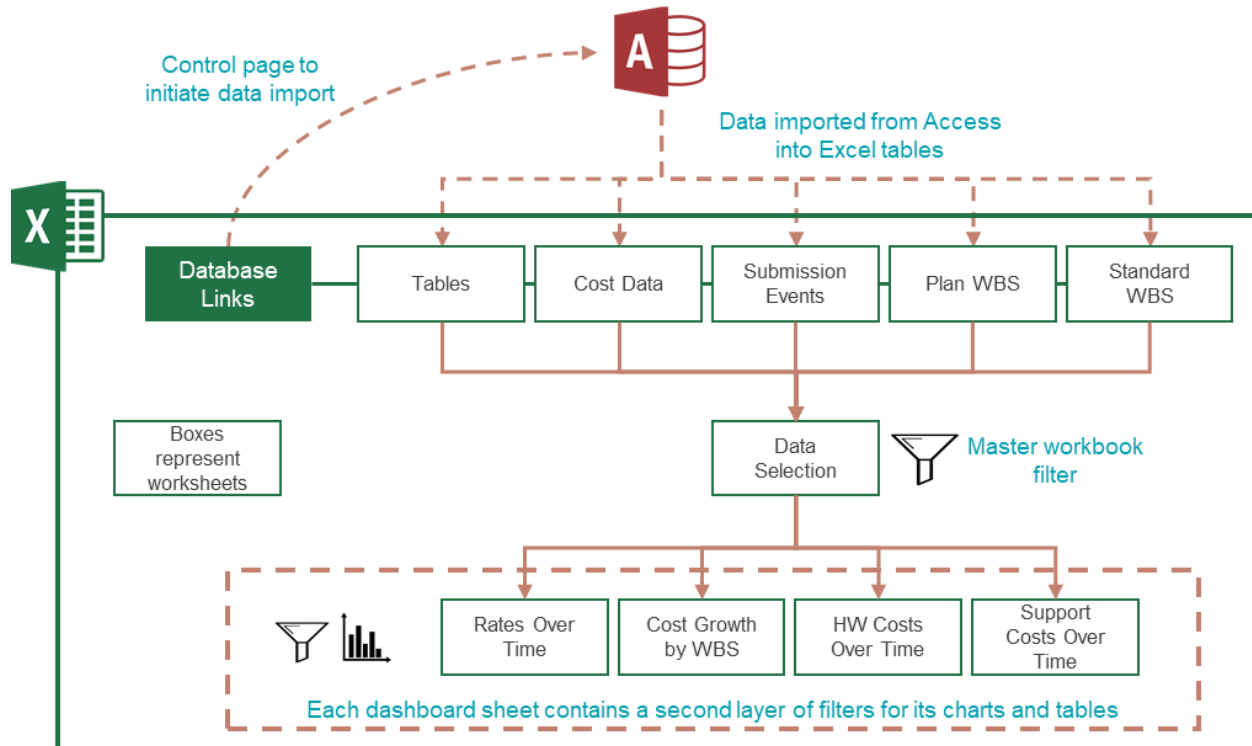


Figure 11: Analytical tool process flow

The Excel tool separates the data from the analysis. A control sheet, 'Database Links', facilitates the download of data directly from Access into five distinct Tables. These include the three primary queries from the database (**Cost Data**, **Submission Events**, and **Plan WBS**), some helper **Tables** (e.g., Functional Categories, Unit Types), and the **Standard WBS**. These can be found in the top row of Figure 11. This data all flow through a master workbook filter by utilizing a series of Excel implemented `filter()` actions.

To support benchmarking, the tool facilitates the loading of a new CCDR not yet in the database. This report is displayed within each dashboard so you can determine how it does (or does not) fit into the historical trends.

From here, research questions receive their own dashboards, each of which comprises a set of filters (e.g., Contract Number, EndItem, Process, WBS element) and data displays (i.e., graphs and tables). Additional questions are either added into one of the existing dashboards or built into a new dashboard, depending on how common the data manipulation steps are. For example, research questions (1) and (2) are virtually identical. One considers profit rates while the other considers overhead rates. These research questions naturally display within the same types of charts and tables. The only difference is the `filter()` criteria to return the relevant data rows.

The following is a pseudo code representation of how each research question can be answered by the data. Each follows a systematic approach within Excel to create a secondary, dashboard specific dataset for use with various data displays.

- (1) How do profit rates vary across delivery orders?

```
filter(desired categories (e.g., process))
filter(relevant 1921-1 data rows)
spread(functional categories into columns)
mutate(rates = cost / hours)
arrange(by date)
```

- (2) What are the overhead rates and how have they trended over time?

same as (1) but with different filter criteria

- (3) How do actual costs and hours grow across the WBS from the initial submission to the final submission?

```
filter(desired categories (e.g., process))
filter(total dollars at completion)
group_by(report sequence and wbs element)
mutate(cost growth = current cost - prior cost)
arrange(by date)
```

- (4) How do per unit costs and hours change over time across major WBS product areas?

```
filter(desired categories (e.g., process))
filter(total dollars and units at completion)
mutate(unit cost = cost / units at completion)
group_by(wbs element and variant)
arrange(by date)
```

- (5) How does the level of effort change over time for select contractor support areas?

```
filter(desired categories (e.g., process))
filter(total dollars at completion)
group_by(wbs element and process)
arrange(by date)
```

| Master Chart Filters | |
|-----------------------------------------------------|--------------------------------------|
| <i>Inherits Initial filters from Data Selection</i> | |
| Costs Normalization | |
| 2 | Base Year 2018 |
| WBS Element | |
| 1 | 0 Report Level Total (Subtotal Cost) |
| Other Options | |
| <input checked="" type="checkbox"/> | Ignore Prior Reports |
| EndItem | |
| <input checked="" type="checkbox"/> | Ground Vehicle System |
| <input checked="" type="checkbox"/> | Vehicle Kits |
| Process | |
| <input type="checkbox"/> | Data |
| <input type="checkbox"/> | Development |
| <input checked="" type="checkbox"/> | Maintenance |
| <input type="checkbox"/> | Manufacturing |
| <input type="checkbox"/> | Operational/Site Activation |
| <input type="checkbox"/> | Program Management |
| <input type="checkbox"/> | Support Services |
| <input type="checkbox"/> | Sustainment |
| <input type="checkbox"/> | System Test & Evaluation |
| <input type="checkbox"/> | Systems Engineering |
| <input type="checkbox"/> | Training |
| <input checked="" type="checkbox"/> | Unit Level Sustainment |
| <input type="checkbox"/> | |
| <input type="checkbox"/> | |
| <input type="checkbox"/> | |
| <input type="checkbox"/> | |
| <input type="checkbox"/> | |
| <input type="checkbox"/> | |
| <input type="checkbox"/> | |
| Show Benchmark | |
| <input checked="" type="checkbox"/> | Include Benchmark Point |
| No benchmark CSDR currently loaded | |

Figure 12: Example filters

Appendix B contains several visual representations of the output. Filters such as EndItem, WBS Element, Process, and other categorical inputs are all controlled through control components such as the checkboxes and dropdown selectors shown in Figure 12.

5 Closing Thoughts

The Stryker example demonstrates how systematically adhering to a few basic, **practical rules** results in elegantly complex solutions. As future data are added and new questions are theorized, nothing about the tool and database fundamentally changes. A well-crafted **data structure** enables the **grammar of data manipulations** to process data sets of any size to efficiently answer a variety of research questions.

This approach intersects the old problem of data management within DoD cost analysis with the current trend of data science. Buzzwords can drive an industry. ICEAA itself has recently experienced an influx of data science presentations at the annual conference. Many focus conceptually on “what it is”, or strive to “wow” with flashy graphs, algorithms, or software. But few address the data itself. Data science is about the application of rigor and scientific methods. The same degree of care and attention that is being spent on the trendy software and analyses must be invested up front in identifying, collecting, and preparing the data.

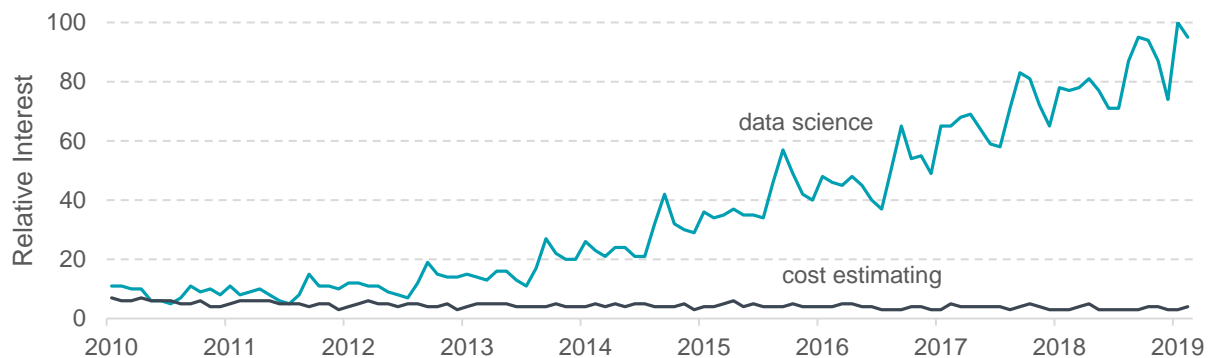


Figure 13: Google search term relative interest over time¹⁷

Steps cannot be skipped in the process. Enthusiastic new analysts are quick to point to modern tools (e.g., R, Python, Tableau) and methods (e.g., machine learning, Bayesian statistics), but find themselves with a **hammer looking for a nail**. The data simply is not there in a usable format. All cost analysts of any experience level can practice better data hygiene. Until the community as a whole commits to better data management, the true power and value of data science will never be realized.

Figure 13 shows that in 2010 the terms “cost estimating” and “data science” were each searched for with the same relative frequency. By 2019, it is no longer close. The future of data analytics is here. It is time for the cost analysis community to join the future.

¹⁷ Data source: Google Trends (<https://www.google.com/trends>)

Appendix A Cell Formulations in Excel

Row indexing is a concept which makes working with data in Excel much easier. Set your first column (usually called 'N', 'Obs', or 'Index') to a simple increasing counter (similar to a primary key in SQL) and operate on this column. Looking up the primary key for use in the `INDEX()` function can provide significant efficiencies down the line.

A.1 Mutate

Mutate is perhaps the easiest verb to use in Excel. Simply add a new column of data and enter in any formula you want. The table will automatically apply the formula to all rows.

A.2 Select

Select becomes unnatural in the Excel world. Usually when working with Excel, you simply input only the variables you need. Therefore, any given formula is selecting variables simply by their use within the equation. Excel does not operate off a distinct data object, therefore this verb is a bit awkward.

The `INDEX()` function is the most direct way to utilize the select concept. `INDEX` can be used not only to return a value, but also an entire range or vector. Consider the table object (or range) `DATA`. The formula `INDEX(DATA, , 2)` will return the entire second column. The `MATCH()` function allows you to determine the location of text within a vector. So the classic `INDEX-MATCH` pair can essentially be used to select an individual variable. This can be nested inside a summarization function (e.g., `SUMIFS()`, `COUNTIFS()`) to swap a different variables in and out of an analysis.

A.3 Filter

Filtering has a couple different purposes. If you simply wish to subset and work with the data, Tables can easily do this through a convenient drop down box in the header.

Often though we are filtering within a data analysis. If data are prepared correctly, this is usually easy to do. Filters amount to criteria within a `SUMIFS()`, `COUNTIFS()`, or `AVERAGEIFS()` formula. For more advanced uses, the `IF()` function can be used in an array form to return a set of rows. Here is where the 'Index' column can be very beneficial. Simply return the row number, and then use `INDEX()` to look up the remaining columns from the table. This technique provides a powerful way to filter based on row references.

A.4 Summarize

Summarize is also a fairly natural extension within Excel. An inconvenient fact of life with Excel is that you have to usually manually prepare how you would like to summarize. Since Excel output is limited to a single cell or a predefined array, we cannot simply let the software figure out what to return on its own. Summarizing usually involves defining row and/or column headers, and then using a formula such as `SUMIFS()` to aggregate on that criteria.

A.5 Arrange

Tables are easy to sort by built in Excel sorting features. The more advanced user may choose to utilize `RANK ()` or a clever `COUNTIFS ()` to develop an ordering to call from `INDEX ()`.

A.6 Group_by

Here is where Excel really struggles. Excel cannot naturally group or iterate, and therefore the user must become creative. Implementing a grouping criteria in Excel involves manually setting up row and/or column definitions (similar to summarize). From here, the aggregation functions can be used to fill in the values. Grouping is not very flexible in Excel. The advanced user can create these definitions dynamically, but even then an area must be prepped to consider the extreme cases.

Appendix B Sample Tool Output

Table Options

Lowest WBS

2

WBS 0 Report Level Total (Subtotal Cost)

Using Data from Rates Over Time (Chart 1)

Metric

Engineering

Mfg Ops

Materials

Blended

Mean

Median

Std Dev

#####

#####

#####

#####

Rates Benchmarking Results

1

2

7

11

Overhead Rates

Engineering

Mfg Ops

Materials

Blended

Row

Level

WBS

Element

Total To Date

Total At Completion

1

1

1.0

Stryker Family of Vehicles System

#####

#####

2

2

1.1

ICV

#####

#####

3

2

1.2

RV

#####

#####

4

2

1.3

MC

#####

#####

5

2

1.4

CV

#####

#####

6

2

1.5

FSV

#####

#####

7

2

1.6

ESV

#####

#####

8

2

1.7

MEV

#####

#####

9

2

1.8

ATGM

#####

#####

10

2

1.9

NBCRV

#####

#####

11

2

1.10

MGS

#####

#####

14

2

1.13

Family of Vehicles System-Systems Engineering/Program Management

#####

#####

15

2

1.14

Family of Vehicles System-ST&E

0.0

0.0

16

2

1.15

Family of Vehicles System - Training

0.0

0.0

17

2

1.16

Family of Vehicles System - Data

0.0

0.0

18

2

1.17

Family of Vehicles System - Peculiar Support Equipment

0.0

0.0

19

2

1.18

Family of Vehicles System-Common Support Equipment

0.0

0.0

20

2

1.19

Family of Vehicles System-Operational/Site Activation

0.0

0.0

21

2

1.20

Family of Vehicles System-Industrial Facilities

0.0

0.0

22

2

1.21

Family of Vehicles System-Initial Spares and Repair Parts

0.0

0.0

23

2

1.22

Family of Vehicles Operations and Support (O&S)

0.0

0.0

Figure 14: Sample benchmark rates comparison output (research question 2)

Figure 14 shows the WBS of a newly loaded report. This report is being benchmarked over a collection of historical data, as specified by the user. This table will provide rate calculations and data against a statistical sample. This is just one of a few views to assess this information. Figure 15, below, provides another view of how various rates will change over time. Note that the new report is “in-sample” and therefore shows within the data range.

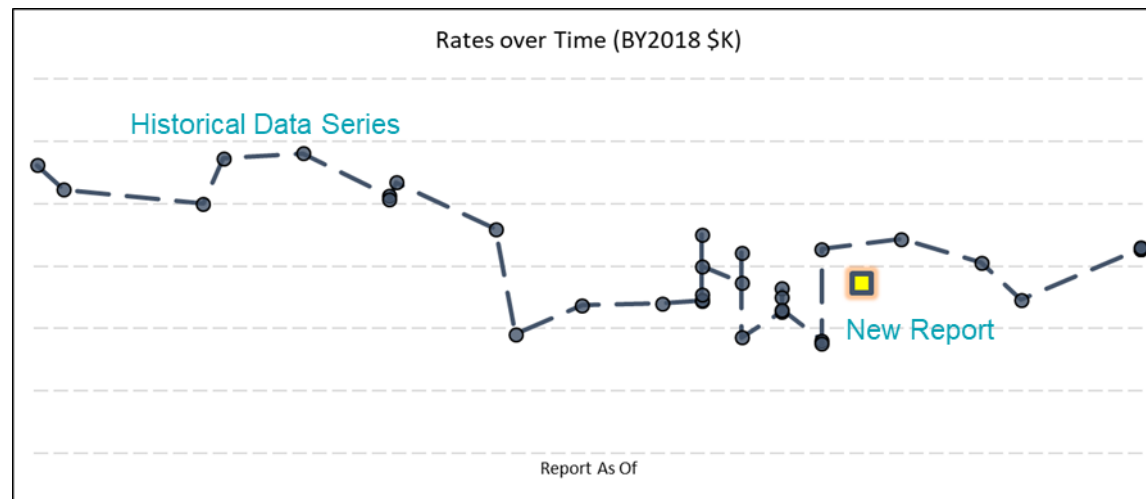


Figure 15: Sample profit/fee rate over time output (research question 1)

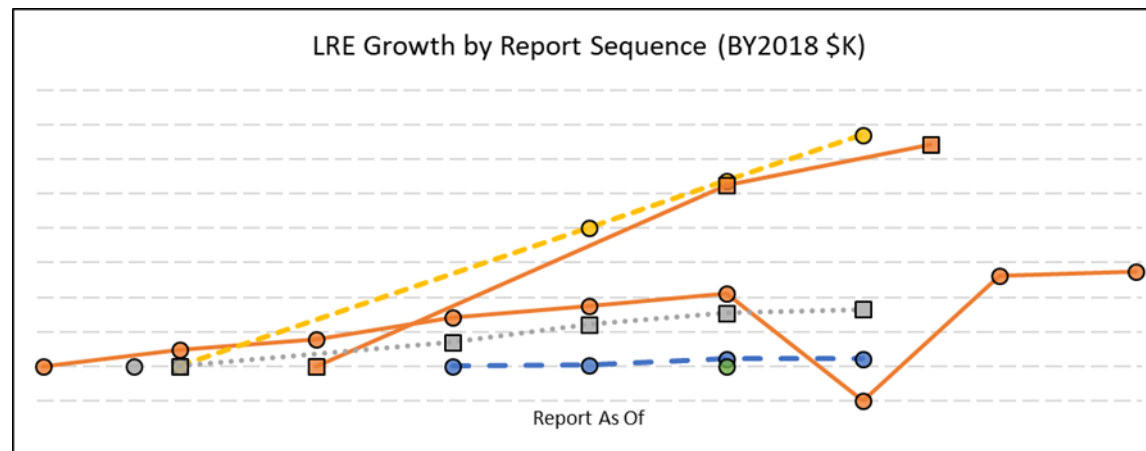


Figure 16: Sample LRE growth over time output (research question 3)

Figure 16 shows how costs grow over time. Each line represents a different CCDR. An increase represents cost or hours growth across a series of submissions.