Integrating AMP Data Through Databases

Bruce Taylor, Rocket Science Consulting

## Goals and Issues

The AMP mission simulation and optimization engine both receives its input and delivers its output to a collection of data structures arranged as a large tree of comma-separated-value (CSV) files. Broadly, the input files contain:

1. Static configuration data such as airports, seaports, aircraft configurations, etc.
2. Simulation-specific configuration data such as aircraft, ships, and trucks available
3. Mission-specific information about cargo and passenger requirements

The output files contain the final results of the AMP simulation and optimization execution. Roughly, they contain:

1. Final location and state of all cargo and passenger requirements
2. Dates of delivery for all requirements
3. Disposition of all aircraft, ships, trucks, etc.

These CSV data structures have been part of the AMP architecture for a long time, and are deeply embedded in the AMP program logic. In this context of AMP execution, they work efficiently.

But increasingly, the AMP data needs to be ingested by other, follow-on analytical tools such as the Excel spreadsheet program, the Tableau data visualization tool, or analytic programs written in Java, R, or MatLab languages. In this role, the CSV files are problematic because

* The logical purpose of each CSV file is not clearly defined.
* There is no readily available data dictionary that defines the contents of each file
* The inter-relationship among the files is complex and not clearly documented.

These restrictions mean that analysts and programmers who wish to further analyze the AMP output data must go through some forensic analysis to locate their needed data and to combine that data appropriately.

## Project Overview

The AMP Data Integration project investigates a solution to this problem that does not require the massive re-engineering of the AMP engine. Briefly, the strategy of the project is:

* Describe the input and output data in the AMP Java code, closest to the point where it is used or produced, and under the maintenance of the AMP application programmers who understand it best.
* Export the AMP data definitions along with the AMP data files when the AMP simulation data is saved. The data definitons are written as representation neutral metadata files in industry standard JSON format, suitable for consumption by follow on analysis programs.
* An independent “AMP Database Tool” reads the metadata files, analyzes their content and interrelationships, and uses that data to create a standard SQL database schema that exactly mirrors the data definitions in the AMP metadata.
* The same AMP database tool is used to read the information from the CSV files and store it in the tables created earlier. The result of these two database operations is an industry standard SQL database containing all the information needed for follow-on analysis.

## Data Description Through Embedded Metadata

The data integration process begins with the AMP application developers. For the critical set of Java classes in AMP, the developer includes a Java “annotation” similar to that shown in below. Annotations are a Java mechanism that allows the developer to encode information about the class outside of the Java code itself. In this case, the annotation is designed to capture all the information required for a data dictionary entry:

* Identity of the data source – this normally maps to a CSV file and to a database table
* Name of the SQL table
* Name of the Java class that governs this data within AMP
* Version of AMP in which this data was introduced
* Definition of each column for the table. The column definition includes:
  + Human readable name of the column
  + Database name of the column
  + Description of the column’s purpose and semantics
  + Logical type of the column’s data
  + Maximum length of textual data
  + Whether the data is required or optional
* (Optional) the set of columns that define the “primary key” of the table. The concatenation of these columns uniquely identifies a logical entity in the table.
* (Optional) the set of “foreign key” columns whose data refers to data in other tables.  
  Databases use the combination of primary and foreign keys to enforce “referential integrity” of the database, and tools like Tableau use them to provide navigational assistance, and to combine tables appropriately.

@ClassMetadata(path="/tplan/aircraftTypes/", tablename="AircraftTypes", filename="aircraft

@FieldMetadata(name="DefMOG", description="Default Parking MOG to be used with associated

@FieldMetadata(name="DefHrsHome", description="Default mission home hours"),

@FieldMetadata(name="DefHrsUpload", description="Default mission upload hours"),

@FieldMetadata(name="DefHrsEnroute", description="Default mission enroute hours"),

@FieldMetadata(name="DefHrsDownload", description="Default mission download hours"),

@FieldMetadata(name="DefHrsRecovery", description="Default mission recovery hours")

@FieldMetadata(name="Name", description="Name of this aircraft")

@FieldMetadata(name="CivMil", isForeignKey=true, foreignKey="CivMilType", description="Whe

@FieldMetadata(name="CargoClass", isForeignKey=true, foreignKey="CargoClassType", descript

@FieldMetadata(name="Prefix", description="Generated Tail Numbers will begin with this pre

@FieldMetadata(name="Substitute", description="Aircraft Type used to compute utilization f

@FieldMetadata(name="DirtyConfig", description="Specified the dirty aircraft configuration

@FieldMetadata(name="Size", isForeignKey=true, foreignKey="AircraftSizeType", description=

@FieldMetadata(name="MaxPayloadRange", description="The range of the aircraft at maximum c

@FieldMetadata(name="MaxFuelLoadRange", description="The maximum cargo load of the aircraf

@FieldMetadata(name="MaxRange", description="The range of the aircraft with maximum fuel a

@FieldMetadata(name="MaxFuelLoad", description="The weight of the maximum effective fuel l

@FieldMetadata(name="MaxCargoPayload", description="The maximum cargo load for the aircraf

@FieldMetadata(name="CargoPayloadMaxFuel", description="The maximum cargo load of the airc

@FieldMetadata(name="FuelBurnRate", description="The burn rate in pounds per hour for this

@FieldMetadata(name="MaxOperatingTime", description="Maximum flying time in hours before a

@FieldMetadata(name="CargoRestricted", description="Whether or not oversize and outsize lo

@FieldMetadata(name="FuelCommodity", isForeignKey=true, foreignKey="Name", description="Th

@FieldMetadata(name="PalletCapacity", description="Pallet capacity of this aircraft")

@FieldMetadata(name="LayoutType", foreignKey="CompartmentLayoutType", isForeignKey=true, d

@FieldMetadata(name="Tanker", description="Tanker of this aircraft")

@FieldMetadata(name="WingType", isForeignKey=true, foreignKey="WingType", description="Win

@FieldMetadata(name="HeloSpeedSling", description=" Speed for rotary wing AC with exterior

@FieldMetadata(name="HeloSpeedNormal", description="Speed for rotary wing AC with standard

@FieldMetadata(name="FlyHomeLegs", description="Is the model flying position / depositioni

@FieldMetadata(name="ReconfigurationTime", description="Time taken to reconfigure aircraft

@FieldMetadata(name="litterCapacity", description="Tanker of this aircraft")

@FieldMetadata(name="BaggagePalletThreshold", description="The baggage pallet threshold am

@FieldMetadata(name="MaxPoundsPerBaggagePallet", description="The maximum pounds per bagga

Figure - Java Data Definition Annotations

## Creating Metadata Files

In the process of compiling the AMP source code, the Java compiler reads the AMP data annotations and converts them to an equivalent metadata format expressed in JSON notation. JSON is an industry-standard representation for structured data that is less formal than XML, but carries less overhead and is easier to read, write, and manipulate. It is widely used in applications, and there are many JSON processing packages available for a wide variety of languages and application programs.

These metadata files are stored in the same directory structure that contains the CSV data files, so they are readily accessible to follow-on analysis programs that need to read and interpret the information in the CSV files. An example of a translated AMP metadata file is shown in below. Notice that all the information supplied by the AMP programmer in the Java annotation is carried into the metadata file; this fidelity ensures that all information available to the programmers is also available to the analysts and applications that consume the AMP data. In essence, the metadata files represent a very complete data dictionary that can be updated automatically for every release of AMP, and used with confidence by analysts and applications.

{

    "id": "/tplan/aircraftTypes/aircraftTypes.csv.meta",

    "title": "AircraftTypes",

    "scenario": "a-metadata-test",

    "classname": "amp.domain.actype.AircraftType",

    "ampversion": "14.13.0 alpha1",

    "tableSchema": {

        "columns": [

            {

                "name": "Name",

                "titles": "Name",

                "description": "Name of this aircraft",

                "datatype": "string"

            },

            {

                "name": "CivMil",

                "titles": "CivMil",

                "description": "The civilian or military type",

                "datatype": "string"

            },

            {

                "name": "CargoClass",

                "titles": "CargoClass",

                "description": "The type of the cargo class",

                "datatype": "string"

            },

Columns Deleted for Clarity

            {

                "name": "LayoutType",

                "titles": "LayoutType",

                "description": "The type of layout for the compartment",

                "datatype": "string"

            }

        ],

        "primaryKey": [

            "Name"

        ],

        "foreignKeys": [

            {

                "columnReference": "CivMil",

                "reference": {

                    "schemaReference": "/tplan/civMils.csv.meta",

                    "columnReference": "CivMilType"

                }

            },

Columns Deleted for Clarity

            {

                "columnReference": "LayoutType",

                "reference": {

                    "schemaReference": "/tplan/compartmentLayoutOptions.csv.meta",

                    "columnReference": "CompartmentLayoutType"

                }

            }

        ]

    }

}

Figure - Translated AMP Metadata File

## Creating Database Schemas

While the information in the AMP metadata files can be used in a number of ways, the important application for this project is to create a standard SQL database containing the AMP data. This database can be used by itself for data analysis, or as input to further analytic tools like Excel or Tableau. The AMP database is created and populated by a new custom application, the AMP Database Tool. The AMP Database Tool goes through four steps to transform each metadata file into an SQL Table, and shown in below:

1. Delete the old version of the table, if it exists.
2. Create the table as named in the metadata, with a representation of all its columns
3. Create database indexes for the primary key and foreign key reference columns. Indexes help the database engine optimize cross-table operations and enforce referential integrity.
4. Add a primary key to tables whenever it is required in the metadata.
5. Add foreign keys to the tables wherever it is required in the metadata.

DROP TABLE IF EXISTS AircraftTypes;

CREATE TABLE AircraftTypes (

   `Name` VARCHAR(50) COMMENT '`Name` of this aircraft',

   CivMil VARCHAR(50) COMMENT 'The civilian `or` military `type`',

   CargoClass VARCHAR(50) COMMENT 'The cargo class',

   Prefix VARCHAR(50) COMMENT 'Prefix of this aircraft',

   DefMOG DECIMAL(10,2) COMMENT 'DefMOG of this aircraft',

   Substitute VARCHAR(50) COMMENT 'Substitute of this aircraft',

   MaxPayloadRange INT COMMENT 'MaxPayloadRange of this aircraft',

   MaxFuelLoadRange VARCHAR(50) COMMENT 'MaxFuelLoadRange of this aircraft',

   MaxRange INT COMMENT 'MaxRange of this aircraft',

   MaxFuelLoad BIGINT COMMENT 'MaxFuelLoad of this aircraft',

   MaxCargoPayload BIGINT COMMENT 'MaxCargoPayload of this aircraft',

   CargoPayloadMaxFuel BIGINT COMMENT 'CargoPayloadMaxFuel of this aircraft',

   FuelBurnRate INT COMMENT 'FuelBurnRate of this aircraft',

   FuelCommodity VARCHAR(50) COMMENT 'The commodity `name`',

   MaxOperatingTime INT COMMENT 'MaxOperatingTime of this aircraft',

   CargoRestricted BOOLEAN COMMENT 'CargoRestricted of this aircraft',

   DefHrsHome DECIMAL(10,2) COMMENT 'DefHrsHome of this aircraft',

   DefHrsUpload DECIMAL(10,2) COMMENT 'DefHrsUpload of this aircraft',

   DefHrsEnroute DECIMAL(10,2) COMMENT 'DefHrsEnroute of this aircraft',

   DefHrsDownload DECIMAL(10,2) COMMENT 'DefHrsDownload of this aircraft',

   DefHrsRecovery DECIMAL(10,2) COMMENT 'DefHrsRecovery of this aircraft',

   Size VARCHAR(50) COMMENT 'The size of the aircraft',

   DirtyConfig VARCHAR(50) COMMENT 'DirtyConfig of this aircraft',

   PalletCapacity INT COMMENT 'PalletCapacity of this aircraft',

   Tanker BOOLEAN COMMENT 'Tanker of this aircraft',

   ReconfigurationTime DECIMAL(10,2) COMMENT '`Time` taken `to` reconfigure aircraft `into`…

   WingType VARCHAR(50) COMMENT 'The `type` of wing',

   HeloSpeedSling INT COMMENT ' Speed `for` rotary wing AC `with` exterior loads (aka …

   HeloSpeedNormal INT COMMENT 'Speed `for` rotary wing AC `with` standard `load` `or` empty…

   litterCapacity INT COMMENT 'Tanker of this aircraft',

   FlyHomeLegs BOOLEAN COMMENT '`Is` the model flying position / depostioning legs `or` will…

   BaggagePalletThreshold INT COMMENT 'FIXME',

   MaxPoundsPerBaggagePallet INT COMMENT 'FIXME',

   LayoutType VARCHAR(50) COMMENT 'The `type` of layout `for` the compartment'

) ENGINE = INNODB;

Figure - Creation of SQL Schema from Metadata

## Importing AMP Data into Databases

The final step in converting the AMP data into an SQL database is to import data from the CSV files into their equivalent SQL tables. This function is also performed by the AMP Database tools through its “populate” mechanism as shown in below. For each metadata file, the tool locates its associated CSV file and, for each record of the CSV file, creates a new row in the associated SQL table. During the process, it checks for:

* Correct matching of columns between the CSV file and the SQL table
* Correct matching or conversion of datatypes between the CSV file and the SQL table
* All rows of the CSV file were loaded correctly into the SQL table.

LOAD DATA LOCAL INFILE 'fedResults/aircraftType.csv'

 INTO TABLE AircraftType

 COLUMNS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'

 IGNORE 1 ROWS

 (Model,FedTime,`Name`,Size,WeightCapacity,PaxCapacity,OnloadTime,OffloadTime,

BlockSpeed500,BlockSpeed1000,BlockSpeed1500);

 /fedResults/aircraftType.csv.meta: read 5 records from fedResults/aircraftType.csv

 /fedResults/aircraftType.csv.meta: loaded 5 rows into AircraftType

Figure - Populating SQL Tables From CSV Files

## Performing Analytic Calculations on AMP Databases

It is possible to perform basic analyses on the AMP database through the SQL function operators. below illustrates the use of SQL to calculate:

* The mission week
* The required arrival time as a mission day
* The actual arrival time as a mission day
* The difference between required and actual times as number of days late
* The product of days late and tonnage as Days/Tons late

Since the SQL creates a database view, this calculation is available through the database and is automatically updated as the underlying data changes. To a consuming analyst or application, using this view is identical to using a table of the database.

CREATE OR REPLACE VIEW `DayTonsLate` AS

SELECT DISTINCT

  CargoId,

  RLN,

  RequiredTime,

  ArrivalTime,

  FLOOR(FedTime/7) as FedWeek,

  FLOOR(RequiredTime) as RequiredDay,

  FLOOR(ArrivalTime) as ArrivalDay,

  Amount,

  Units,

  OnloadGeoloc,

  OffloadGeoloc,

  (FLOOR(ArrivalTime-RequiredTime)) as DaysLate,

  (FLOOR(ArrivalTime-RequiredTime)\*Amount) as DayTonsLate

FROM `RequiredVsArrivedView`;

Figure 5 - Calculating Days/Tons Late Using SQL

## Analyzing AMP Data Using Excel

Once the AMP data is stored in a database, with appropriate analytic views defined, it is available for analysis through a variety of applications. below show an Excel spreadsheet that accesses the data in the “DayTonsLate” view defined in above.

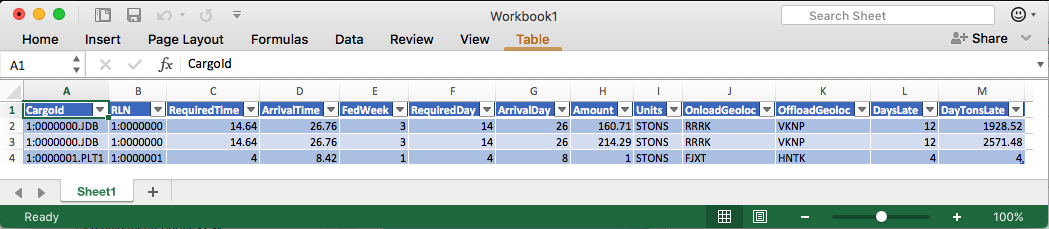


Figure - Excel Access to Day/Tons Late Data

The data imported into Excel from the AMP database can be used as is, or can be input to further analysis within Excel. For example, below illustrates analyzing the overall days/tons late per offload location using a simple Excel pivot table. Further analysis is limited only by the analyst’s familiarity with Excel.

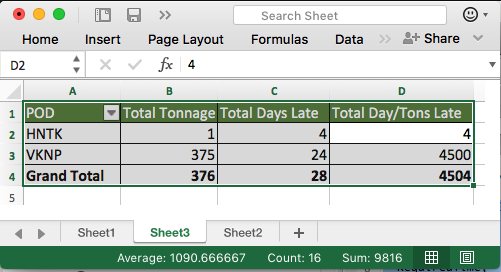


Figure - Pivot Table Analysis of Days/Tons Late

## Analyzing AMP Data Using Tableau

The Tableau tool is a widely accepted application for light analysis and visualization of tabular data, and it has extensive features for reading, combining, and displaying information in SQL databases. This feature makes it ideal for further analysis of AMP data, when simpler tools like Excel have insufficient features. In an experiment, Tableau was able to connect easily to the AMP database created in earlier steps, and was able to produce two useful displays with minimal effort.

below illustrates a stacked bar chart of days/tons late per offload port, over the course of the TPFFD calendar weeks. It is clear from the graph that all ports were busy during the initial surge of the mission (weeks 3-8) but that the two busiest of the ports (HNTS and VNTP) were significantly overloaded, while the smaller ports were able to keep up with their demands. Further, it shows that VNTP continued to have problems with late tonnage throughout the TPFFD calendar period.

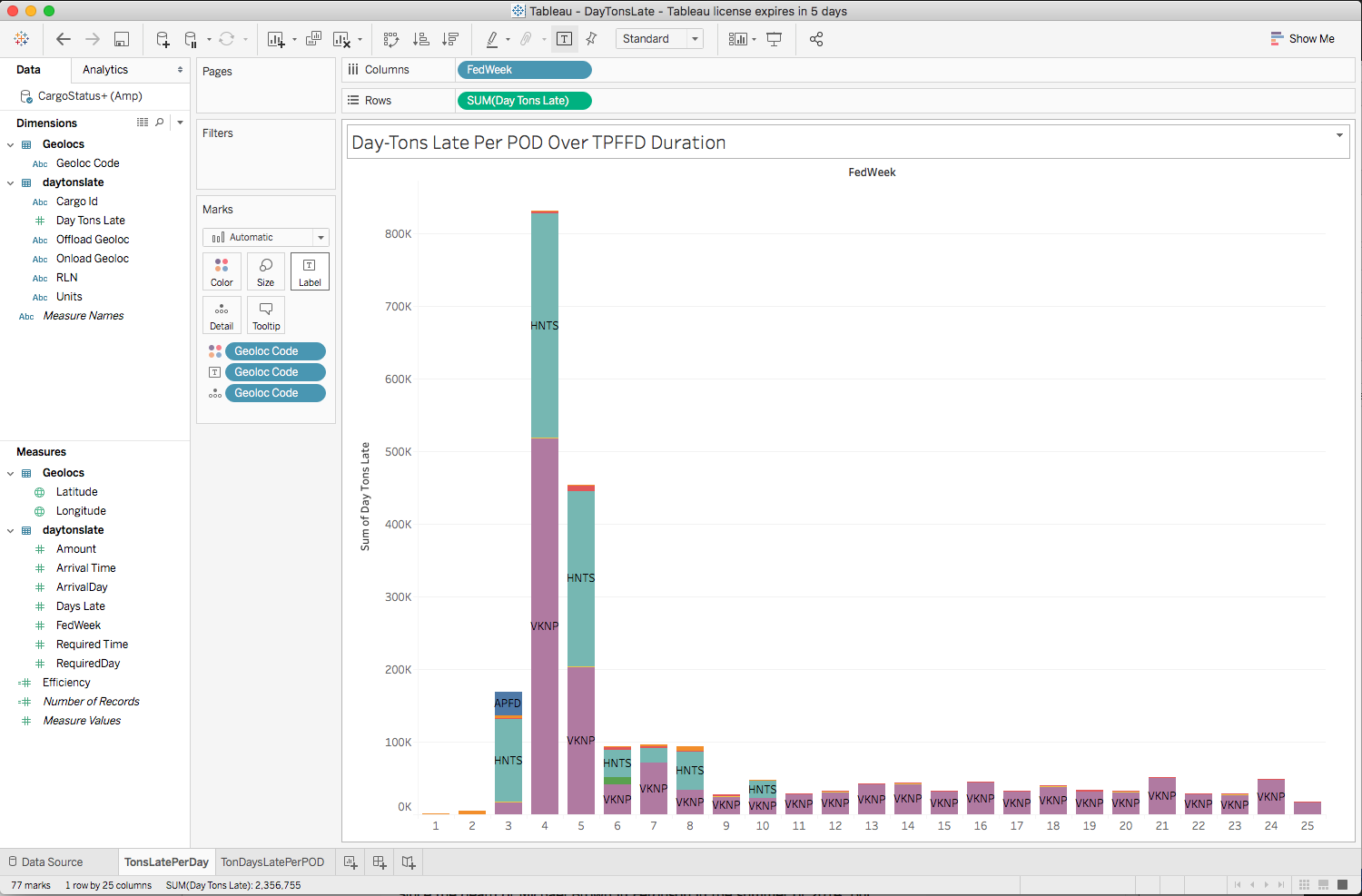


Figure - Tableau Calculation of Day/Tons Late per POD

below shows another analysis of the day/tons late metric: plotting two variables on a map of the offload ports. In the figure,the size of the circle at each port is proportional to its average day/tonnage. The color of its circle is a function of a simple “efficiency” metric:

**port efficiency = average daily tonnage / average days late**

This display indicates that the two ports with the largest volume, VKNP and JEAH, had adequate to good effiency, moving large volumes of daily tonnage with minimal days late. However, the two air ports, HNTK and FUQN operated very inefficiently, moving smaller volumes of daily cargo with greater average daily delays.

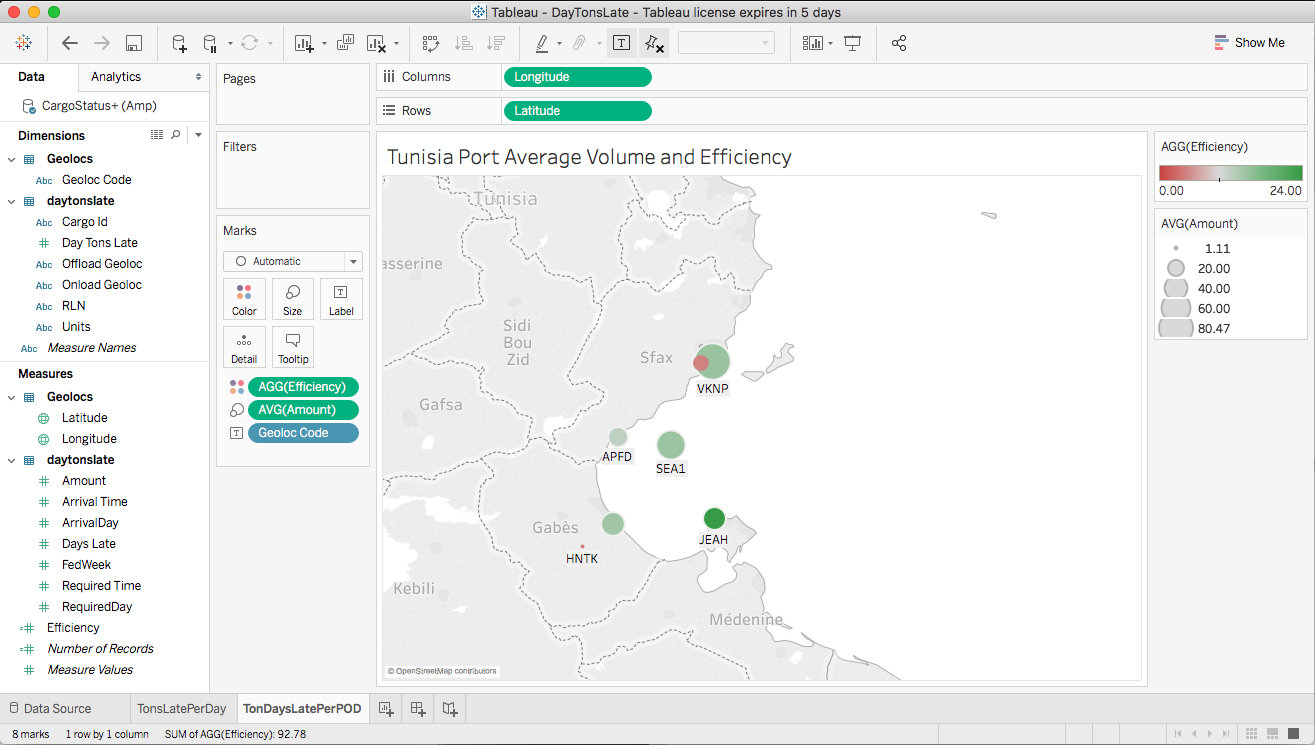


Figure - Tableau Display of Average Port Volume and Efficiency Metric

The two examples shown, and others not shown, indicate that the Tableau tool is an effective application for analyzing and visualizing AMP data from an SQL database. Its characteristics:

* Very easy connectivity to the database
* Intuitive tools for combining tables and views and selecting data of interest
* Ability to further filter data based on user-selected values
* Ability to include “calculated fields” such as the efficiency metric that do not appear in the database
* Easy visualization of the resulting data in a rich set of formats, including geolocation formats such as mapping.

## Analyzing AMP Data with the R Language

The R language is a standard tool for implementing advanced analysis of data, with libraries that support analysis of:

* parametric and non-parametric data,
* linear and non-linear regression
* simple variance and multivariate analysis
* data clustering
* Bayesian analysis
* Multi-dimensional array computation
* Data quality analysis
* Signal detection
* Autocorrelation analysis
* Geographic computations
* and many more

illustrates the simplicity of importing data from SQL databases into the R workbench, where the heavy analysis can take place. illustrates the import of the Day/Tons Late data from the “DayTonsLate” view defined in above. The R database package can perform appropriate data type conversions from a wide variety of formats, as shown by the lines in red below.

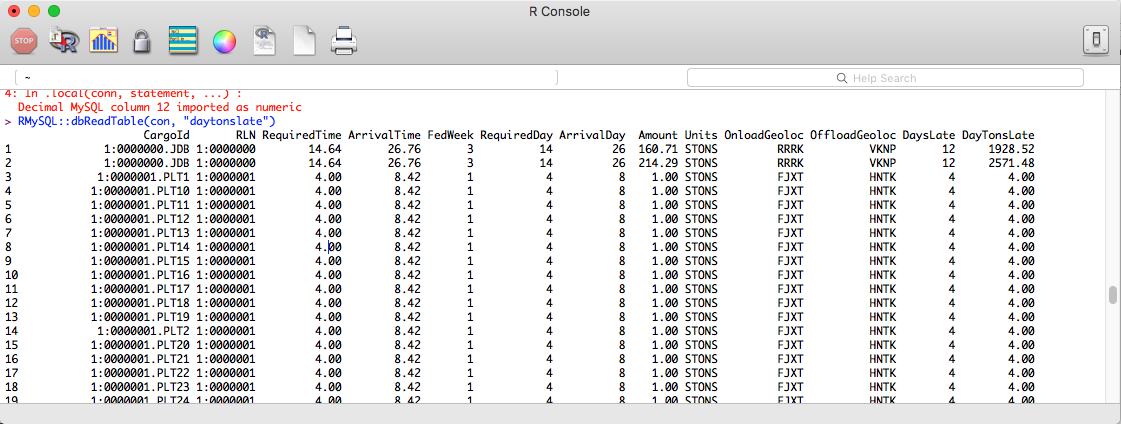


Figure - Reading AMP Data for Analysis Using R

Once the AMP data has been read from the SQL database into the R workspace, it can be summarize, analyzed, and displayed in a number of ways, limited only by the analyst’s familiarity with the R statistical packages. below illustrates the simplest analysis: simple summaries of each of the numeric colums by min, max, mean, median, and quartiles.

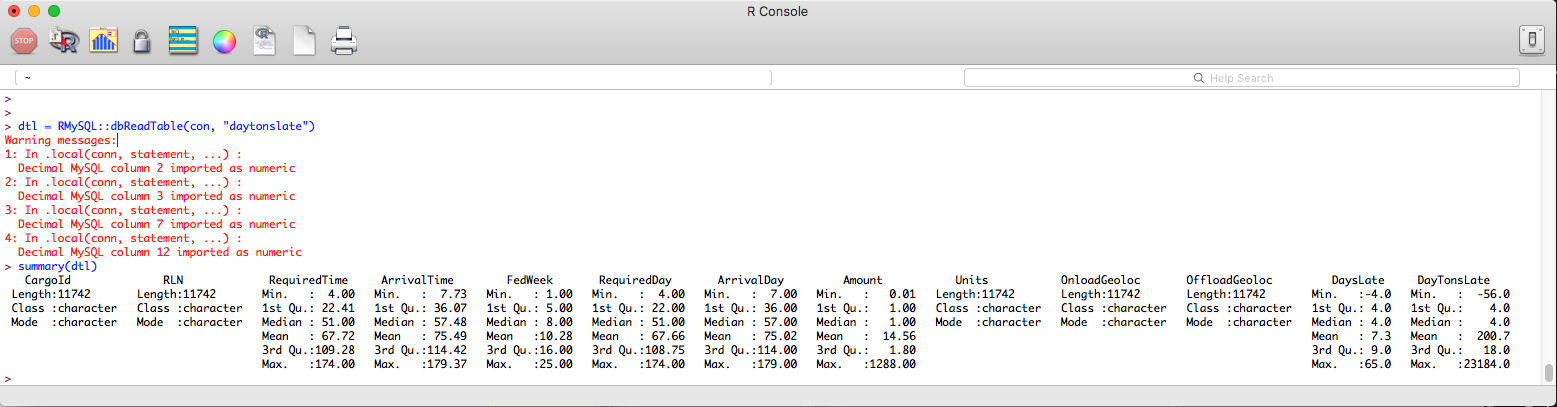


Figure - Simple Summary Analysis of Day/Tons Late Using R

## Conclusions

The motivation behind the AMP Data project was to make AMP’s input and output data available to analysts and to follow-on applications without having to rewrite those portions of the AMP code that depend on the existing tree of CSV files. The proposed solution was to document the structure of the data at the earliest possible moment: in the code of the Java classes that manage the AMP data. This “data dictionary” information is then carried throug to metadata files stored along with the CSV files. Then, the combination of the CSV files containing the data, and the metadata files defining their contents, provides enough information to make further analyis possible.This study validated the metadata approach, indicating that :

* it causes very little perturbation to the AMP code
* the metadata can be managed from one AMP release to the next
* the metadata contains sufficient information to reconstruct the AMP data in a different context

As an extension of the metadata approach, an AMP Database Tool provided two critical functions: transforming the AMP metadata descriptions into a standard SQL database schema, and importing the AMP CSV file data into the created database. From that neutral SQL platform, further analysis becomes possible and even easy. This study demonstrated successful follow-on analysis using:

* Excel spreadsheets for simple analysis
* Tableau for data visualization
* R for advanced analytic calculations.

We conclude that the metadata approach to AMP data sharing is efficient and effective, and should rank first among the options for further development.