# Penny for Your Thoughts

## Measuring the Impact of Funding on Educational Outcomes

Cassandra Seney, Elizabeth Chabot, Rani Fields
22nd August 2017

## Objectives and Constraints

The goal of our project is to answer a research question investigating the relationship between funding and academic outcomes for public school students. Is there a direct relationship between funding for education and student achievement? What do we see when we examine the available data?

Answers to questions about educational funding are important to policymakers who determine where our tax money is spent, as well as to parents of school age children and their teachers. With ESSA replacing No Child Left Behind during the Obama Administration the role of evaluating schools and determining how to ‘fix’ them fell largely on each state. Pros and cons existed for each bill, but the question still remained, “How should we measure academic success or educational outcomes within the US?”. With the Trump administration now overturning many of the educational bills passed during the Obama and Bush administration federal, state and local funding of public schools is in flux.

Changes made thus far include less strict review of teacher preparation programs and funding being cut to help states identify failing schools and development of plans to improve schools success rates. With less funding being aloted to reviewing different measures contributing to educational outcomes our team  focused on creating an educational database focused on collecting factors that contribute to educational outcomes and develop measures to determine the success of students, and thus schools. For the purpose of this project we focused first on gathering funding and financial data along with any available educational factors related to public schools K-12. Due to time constraints and poor collection or data quality we settled on CCD and NAEP datasets.

Since this is a public interest topic, we wanted our results to be reproducible by others. So we made sure we used easily accessible publicly available data and a low cost solution. We also want the project to continue to be relevant in the future with the ability to incorporate future data sets.

How much of an impact funding has on children’s success is debatable, but through this project we hope to give better insight on how public school funding is linked to a school’s ability to output successful students. It is also an important question for the future of our country, as a well educated and informed society is better equipped to take a leadership role on the world stage.

# Data Sources

To investigate our research question we required fiscal data measuring the revenue received by public schools annually in each state. We also needed a measure of academic performance. We obtained this data from two sources.

## Common Core of Data

The Common Core of Data (CCD) is the Department of Education's database of public school data nationwide. It collected through surveys conducted annually. We used two data sets from the CCD:

- Fiscal provides details of school funding with breakdown into local/state/federal sources and expenses including teacher salaries and benefits
- Non fiscal includes total numbers of teachers and students for calculations such as pupil teacher ratios and funding per student

## National Assessment of Educational Progress

The National Assessment of Educational Progress (NAEP) is the largest nationally representative and continuing assessment of student academic progress. Beginning with the 2003 assessments, national assessments are conducted every two years in reading and mathematics. Although the test was performed earlier, year 2003 was the first statewide full participation.

We chose to use the 8th Grade NAEP test scores as the data set was more complete. The NAEP test was first administered to 8th grade and then later added to 12th grade so there is a longer history of data for 8th grade.

We also considered using SAT data, but although we submitted a request for access to SAT data, we received no response. The NAEP data proved adequate for our purposes.

We chose to focus on statewide data to provide a more detailed analysis than federal level data would provide and we were able to leverage complete data sets that were available publicly.

We recognise that our definition of educational success is limited to factors measurable in a statistical survey. Outcomes such as teamwork, social leadership skills and a love of learning are difficult to quantify although they are undoubtedly valuable outcomes of a good education. We are also relying on educational assessment standards to measure success which do not necessarily provide a complete picture of academic achievement. Despite these limitations we felt we were able to provide good insight into the relationship between educational success and funding.

# Architecture

Identifying which solution was best suited to our use case began with considering the following characteristics.

## Data Characteristics

- Structured. The source data has a well defined structure with columns corresponding to property values and data types defined in the data layout document.
- Small size.
- Low sink latency and high source latency. Batch updates will occur annually, but we would like to be able to query the data as soon as it is imported.
- Medium quality. Most of the data quality issues are resolved in the transform scripts. Some filtering by constraints will be desirable.
- Complete data.

## Processing Dimensions

- High to low selectivity. Some of our queries are highly selective and others are not.
- Short query time. We expect sub-second responses to our queries.
- Short processing time for batch jobs.
- Medium aggregation allowing us to aggregate over multiple dimensions.
- Advanced joins across multiple tables
- Exact precision handling numeric data

## Other Characteristics

- Low cost. Open source preferred.
- Simple to configure and maintain.

## Preferred Solution

These characteristics matched most closely with a relational database. Our desire for a low cost open source solution placed PostgreSQL at the top of our list. It is relatively easy to administer, stable and well maintained, and supports all the standard SQL constructs as well as some of the more advanced features. It also integrates with R, Tableau and other solutions we were considering as part of our implementation.

## Scalability

Our data sets are small with the potential to grow over time. It is expected that the educational statistical data we use will continue to be generated every year for the foreseeable future. The annual increments impact the number of rows in the tables, but do not greatly increase storage needs. We were able to store all of our database files locally, reducing the network overhead and cost of external storage solutions.

We chose to implement partitioning within the Postgres database to help ensure that query performance remained fast as the number of rows increased over time.

We also focused on appropriate indexing to ensure fast performance of our queries.

## Cloud

We chose to implement our solution in an Amazon Web Services Linux image for ease of collaboration. We also used a github repository to share the solution implementation.

# Schema

We are using third normal form schema on write with row based tables. Constraints are used to validate the data before persisting. The combination of state and year provides a common key to relate the tables. Indexes are defined for faster access. Partitioning is implemented for scalability.

## Tables and Columns

Our Entity Relationship Diagram identified three classes of data - fiscal, nonfiscal and scores, where both fiscal and nonfiscal data influence the scores. For each table the year/state uniquely identifies the entity and provides a link to corresponding data in other tables.

**Fiscal data**

| | |
|---|---|
| survey year | year |
| state | character |
| state revenue | numeric |
| local revenue | numeric |
| federal revenue | numeric |
| total revenue | numeric |
| teacher salaries | numeric |
| teacher benefits | numeric |
| current expenditures | numeric |

**Nonfiscal data**

| | |
|---|---|
| survey year | year |
| state | character |
| total teachers | numeric |
| grade 8 students | numeric |
| grade 12 students | numeric |

| | |
|---|---|
| total students | numeric |

**NAEP scores grade 8**

| | |
|---|---|
| test year | year |
| state | character |
| mathematics score | numeric |
| reading score | numeric |

In addition, these values would allow us to perform queries to calculate the following:

- Total revenue per pupil as total revenue/total students
- Teacher compensation as (teacher salaries+teacher benefits)/total teachers
- Student teacher ratio as total students/total teachers

PostgreSQL has a remarkably enlightened approach to handling numeric data where it does not require the scale and precision to be defined in advance and will adapt based on the data input.

## Constraints

In addition to “not null” modifiers, check constraints have been used to validate important values during import, including the following

- total revenue > 0
- total teachers > 0
- total students > 0
- mathematics and reading scores must be between 0 and 500

Furthermore, the insert trigger checks the state value and will only insert rows which match defined states.
We also defined a domain for the year data type with a constraint which checks that the value for year is valid.

## Indexes

On each table, the combination of year and state has been defined as the primary key. This key relates the tables. It is also used in most of our queries and is therefore valuable for performance reasons.

## Partitioning

We have divided the tables into five state range based partitions. Since most of our queries gather data per state, we implemented 5 range partitions with the states divided evenly.
Using state for the partition range also means that new years of data can be added without repartitioning.
The insert trigger function checks the value for state, and inserts the row into the appropriate partition table. If the state does not match a partition, a notice is issued and the row is skipped. This helps us ensure that only data from states common to all the data sets are imported.

# Extract, Transform and Load

The first step was to look for existing tools which could be used to import the data into PostgreSQL.

- Unlike some of the more expensive commercial database products, Postgres does not provide full featured ETL tools.
- An open source tool called pgloader initially looked promising. It was able to download files from a website and import the csv contents into database tables. Upon closer examination however, it provided no functionality for extracting specific columns from a csv file.
- The PostgreSQL COPY command is generally considered to perform much better than other alternatives (including pandas for instance).

We decided to write our own script to extract and transform the data and then use a sql script to create the schema and populate with data using the COPY command.

## Download

The script downloads zip files containing the data from the CCD website. There is one fiscal and one nonfiscal zipped data file for each year.

The filenames of the data sets do not use a consistent naming pattern. An earlier version of the script attempted to use pattern matching to retrieve the filenames, but due to website redirects the script was attempting to access files not intended. As a safer approach, the script now uses text files containing the URLs of the data sets to be downloaded. Each year as new data is added to the CCD website, the text files can be updated to add the URLs of the new files.

Each file is downloaded to a temporary file, unzipped and extracted then deleted before the next file is retrieved. With this design we only need to store the extracted content of the files.

## Extract and Transform

The most recent fiscal and nonfiscal data files are in tab delimited format with a header row. The extraction process reads each row of the file and parses the rows to output just the required columns into a comma delimited csv file.

We implemented the script to overcome the following challenges:

- the tab delimited data files contained null values
- some field values contained commas
- the format and columns of the nonfiscal data file changed. From 2003 to 2006 the files were in fixed width format. In 2007 the format changed to tab delimited. Starting in 2008 additional columns of demographic data were added to the files.
- the nonfiscal data files from 2003 to 2006 did not contain a header row but subsequent years did have column headers
- some files contained data for extra territories. The CCD include 56 states and territories including District of Columbia, American Samoa, Guam, Northern Mariana Islands, Puerto Rico, Virgin Islands. Some earlier years of nonfiscal data included DoD school

information while later years did not. The NAEP data includes 50 states plus District of Columbia and DoDEA (Department of Defense Education Activity military schools). We chose to use the overlapping data from the 50 states and District of Columbia in our analysis.

### OpenRefine

The NAEP website provides an interactive form to generate csv file data which precludes the possibility of automating the download. The data file layout is designed for aesthetics not analysis:

- multiple header rows
- blank columns
- year values are provided in the first row of each table but not populated in subsequent rows

Fortunately the data for all years is included in a single file. This made it easy to quickly process the file in OpenRefine.The empty rows and columns were removed. The fill down tool was useful for populating the year value in rows where it was missing. The data was exported as a csv file.

### Load

The SQL script which creates the database schema also loads the data from csv files into the tables using the COPY command. The data is checked by constraints on fiscal and state values.

## Execution

The code for our scripts is available in the ETL folder of our project repository with a readme detailing steps for running the scripts.

1. Run ./extract_transform.sh

This process retrieves data files from the National Center for Educational Statistics website based on URLs recorded in the fiscal.txt and nonfiscal*.txt text files in the same directory.
The output of the script is two csv files: fiscal.csv and nonfiscal.csv

While it is running, the extract_transform.sh script displays output as below.

```
[w205@ip-172-31-22-103 ETL]$ ./extract_transform.sh
--2017-08-15 05:12:21--  https://nces.ed.gov/ccd/data/zip/stfis14_1a.zip
Resolving nces.ed.gov... 63.145.228.23, 2001:428:7003:11::23
Connecting to nces.ed.gov|63.145.228.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 33374 (33K) [application/x-zip-compressed]
Saving to: `temp.zip'

100%[=========================================================>] 33,374      44.6K/s   in 0.7s

2017-08-15 05:12:22 (44.6 KB/s) - `temp.zip' saved [33374/33374]

51 fiscal.csv
--2017-08-15 05:12:22--  https://nces.ed.gov/ccd/data/zip/stfis13_1a.zip
Resolving nces.ed.gov... 63.145.228.23, 2001:428:7003:11::23
Connecting to nces.ed.gov|63.145.228.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 34621 (34K) [application/x-zip-compressed]
Saving to: `temp.zip'

100%[=========================================================>] 34,621      93.2K/s   in 0.4s

2017-08-15 05:12:23 (93.2 KB/s) - `temp.zip' saved [34621/34621]

102 fiscal.csv
--2017-08-15 05:12:23--  https://nces.ed.gov/ccd/data/zip/Stfis_1a_txt.zip
Resolving nces.ed.gov... 63.145.228.23, 2001:428:7003:11::23
Connecting to nces.ed.gov|63.145.228.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 35042 (34K) [application/x-zip-compressed]
Saving to: `temp.zip'
```

2. The load.sql script is executed within PostgreSQL: \i /home/w205/205_Project/ETL/load.sql

After creating the schema, the data is loaded from the csv files created by the previous step.

While it is running the load.sql script displays output as below:

```
postgres=# \i /home/w205/205_Project/ETL/load.sql
DROP DATABASE
CREATE DATABASE
psql (8.4.20)
You are now connected to database "w205project".
CREATE DOMAIN
ALTER DOMAIN
CREATE LANGUAGE
ALTER LANGUAGE
SET
psql:/home/w205/205_Project/ETL/load.sql:42: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit
 index "fiscal_pkey" for table "fiscal"
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
```

After the scripts are completed, the tables can be shown using the \dt command.

```
w205project=# \dt
            List of relations
 Schema |     Name      | Type  |  Owner
--------+---------------+-------+----------
 public | fiscal        | table | postgres
 public | fiscal_1      | table | postgres
 public | fiscal_2      | table | postgres
 public | fiscal_3      | table | postgres
 public | fiscal_4      | table | postgres
 public | fiscal_5      | table | postgres
 public | naep8         | table | postgres
 public | naep8_1       | table | postgres
 public | naep8_2       | table | postgres
 public | naep8_3       | table | postgres
 public | naep8_4       | table | postgres
 public | naep8_5       | table | postgres
 public | nonfiscal     | table | postgres
 public | nonfiscal_1   | table | postgres
 public | nonfiscal_2   | table | postgres
 public | nonfiscal_3   | table | postgres
 public | nonfiscal_4   | table | postgres
 public | nonfiscal_5   | table | postgres
(18 rows)
```

The columns, indexes, constraints and triggers for the fiscal table are displayed with \d fiscal.

```
w205project=# \d fiscal
                  Table "public.fiscal"
        Column        |         Type         | Modifiers
----------------------+----------------------+-----------
 survey_year          | year                 | not null
 state                | character varying(2) | not null
 state_revenue        | numeric              | not null
 local_revenue        | numeric              | not null
 federal_revenue      | numeric              | not null
 total_revenue        | numeric              | not null
 teacher_salaries     | numeric              | not null
 teacher_benefits     | numeric              | not null
 current_expenditures | numeric              | not null
Indexes:
    "fiscal_pkey" PRIMARY KEY, btree (survey_year, state)
Check constraints:
    "fiscal_total_revenue_check" CHECK (total_revenue > 0::numeric)
Triggers:
    insert_fiscal_trigger BEFORE INSERT ON fiscal FOR EACH ROW EXECUTE PROCEDURE fiscal_insert_trigger()
```

The columns, indexes, constraints and triggers for the fiscal table are displayed with \d nonfiscal.

```
w205project=# \d nonfiscal
              Table "public.nonfiscal"
     Column      |         Type          | Modifiers
-----------------+-----------------------+-----------
 survey_year     | year                  | not null
 state           | character varying(2)  | not null
 total_teachers  | numeric               | not null
 grade8_students | integer               | not null
 total_students  | integer               | not null
Indexes:
    "nonfiscal_pkey" PRIMARY KEY, btree (survey_year, state)
Check constraints:
    "nonfiscal_grade8_students_check" CHECK (grade8_students > 0)
    "nonfiscal_total_students_check" CHECK (total_students > 0)
    "nonfiscal_total_teachers_check" CHECK (total_teachers > 0::numeric)
Triggers:
    insert_nonfiscal_trigger BEFORE INSERT ON nonfiscal FOR EACH ROW EXECUTE PROCEDURE nonfiscal_insert_trigger()
```

The columns, indexes, constraints and triggers for the fiscal table are displayed with \d naep8.

```
w205project=# \d naep8
                Table "public.naep8"
    Column     |         Type          | Modifiers
---------------+-----------------------+-----------
 test_year     | year                  | not null
 state         | character varying(2)  | not null
 math_score    | numeric               |
 reading_score | numeric               |
Indexes:
    "naep8_pkey" PRIMARY KEY, btree (test_year, state)
Check constraints:
    "naep8_math_score_check" CHECK (math_score >= 0::numeric AND math_score <= 500::numeric)
    "naep8_reading_score_check" CHECK (reading_score >= 0::numeric AND reading_score <= 500::numeric)
Triggers:
    insert_naep8_trigger BEFORE INSERT ON naep8 FOR EACH ROW EXECUTE PROCEDURE naep8_insert_trigger()
```

After the data is loaded the database is small, justifying the storage choices we made.

```
w205project=# select pg_size_pretty(pg_database_size('w205project')) as size, pg_database_size('w205project') as bytes;
  size   |  bytes
---------+---------
 6294 kB | 6444816
(1 row)
```

# Exploratory Data Analysis

We performed exploratory data analysis using R connected to the PostgreSQL database. This helped us better understand our data in preparation for later visualizations. For instance we noticed that there were some states with much higher revenue and that those were also the states with the highest numbers of students, so we decided to look at revenue per student in our visualizations. Similarly we found that remuneration per teacher made more sense that the raw salary plus benefits data given the different numbers of teachers per state. We also found that the data for DC included outlier values and not applicable values so we might want to focus on the 50 states in our final presentation for more consistent data.

The full data exploration report is available as [pdf]() or as [Rmd]() format file.

# Visualization

The first step in the visualization process was deciding what kind of story our data wanted to tell and which queries would best enable us to convey the information contained in the data. Our prior EDA proved helpful as a baseline for existing relationships. The creation of the visualizations also helped to find data quality and data integrity. During the composition of our visualizations we reviewed 3 applications, R Shiny, Tableau Server, and Tableau Public, to output our analysis to the public. We resolved to use Tableau Public and posted our analysis on Tableau Public's site.

## Choice of Visualization Product

For our project, all data is static and incremental uploads will take place every year to 2 years. We do not need to take into consideration streaming data, high velocity or data approaching 'big data' status at the moment. We as a team decided that the best way to output a response to the RQ would be to create a dashboard and story documenting our analysis. The audience we foresee reviewing this analysis will be the general public and our stakeholders.

### *Tableau Server*

We initially explored Tableau Server as an easy to access solution which facilitates collaboration. While this solution is costlier than other solutions we covered due to both hosting and licensing costs, the familiar UI would allow for rapid dashboard development and the unique workbook management system would allow us to work on our visualizations in tandem with various team members. Tableau Server has a stringent list of hosting requirements. Notably, Tableau Server required Windows 2012/2016 Server, two physical cores, and 15 GB of RAM[1].

While Tableau Server would have been the best solution despite the cost, issues associated with the installation process preventing us from pursuing this route.

### *R Shiny*

R Shiny requires a small amount of time to learn how to code if you have experience with R. Due to ample tutorial resources, it should take the average R user no more than 2 days to learn the system. if you lack R knowledge then it could take weeks. For this project, however, all team members should be versed in R which is why this solution was considered.

If working on a team that does not code in R, Plotly, Bokeh, Seaborn are other considerations for those who code in Python, while D3 visualizations take the same amount of time if able to

[1] "Shiny Server Professional v1.5.3 Administrator's Guide"

create JavaScript, HTML, CSS. There is also a myriad of other strictly JavaScript solutions such as HighCharts and Chart.js. What these solutions boils down to is building up a visualization or dashboard from code and packages. It requires one to know enough in one of the above languages as well as the statistical knowledge to create a dashboard or visualization. Additional time and effort comes from coding each visualization and having to establish the composition of the dashboard with care, however, once a baseline is established components can be copy and pasted. Despite the issues with programmatic databases, we found it useful to attempt to use Shiny due to its collaboration potential.

Our team spent 48 hours working with both Tableau Public and R Shiny to create a base dashboard with visualizations to see what would be the best option for this current project. In R Shiny we were able to create two base functional dashboards with limited visualizations each, while we were able to create a multitude of dashboards in Tableau Public.

### *Tableau Public*

Tableau Public for our team provided an easier solution for the creation of visualizations and dashboards. We were able to create three dashboards and over ten visualizations. Tableau won in speed of creation over R Shiny. Additionally, in a real world setting we would be able to create new visuals quickly and easily to present to stakeholders. R Shiny would be able to create more detailed visuals with more options for statistical analysis. If this was not a project we would take into consideration in more detail what our stakeholders were requesting and if the analysis required more advanced statistics we would most likely move to R Shiny or create an additional layer (possibly presentation layer) in R Studio before outputting the data to Tableau.

## Tableau Server vs. R Shiny: a Detailed Look

As Tableau Server and R Shiny were our primary solutions, we've compiled a detailed look into the differences between the two technologies. The primary areas we will be covering are set-up and maintenance issues, the costs associated with each product, the flexibility of each product, and each product's conduciveness to advanced analytics and automation.

On the next page you will find a table documenting each one of these key areas in detail. Whichever product appears to offer the stronger solution for that area will be color-coded green. Please note that we are judging each area strictly in the context of our project. That is, we weigh time-related risks with disproportionate importance.

*A point-by-point look at R Shiny vs Tableau Server*

| Area | R Shiny | Tableau Server |
|---|---|---|
| **Set-up / Maintenance** | • Likely will need to deal with R dependencies<br>• R updates can break Shiny<br>• Low set-up time unless issues with R arise | • Difficult to set up<br>• Tableau Server stability and maintenance risks are currently unknown however individual reviews[2] imply Tableau products to be low risk |
| **Costs** | • Open source<br>• An accessible free version is available<br>• Commercial solutions are available if certain features are required | • $293/mo for around-the-clock hosting, does not include EBS hosting. Includes licensing (5 users) and VM costs.<br>• A more detailed cost analysis can be found below. |
| **Flexibility** | • More options for bringing in multiple sources<br>• Creating visualizations is not as simple as drag-and-drop<br>• Can copy template widgets with ease: just copy and paste! | • Cumbersome UI makes simple tasks difficult<br>• UI-based dashboard design allows for rapid development |
| **Advanced Analytics & Automation** | • Dashboards are engineered in R, a premier analytical language<br>• Programmed dashboard: allows for extensive automation, especially with regards to complex analysis | • Large in-house suite of analytics features such as k-means clustering and zip code handling<br>• Recently added Python support[3]<br>• Due to the recently added Python API, more analysis needs to be performed to gauge advanced analytics and automation ability. |

At this time, we cannot make any claims about load and performance. We do know that both Tableau Server and Shiny (open source) do not have a cap on concurrent dashboard viewers.

### *Tableau Server: Financial Analysis*

In this section we will briefly analyze Tableau Server's potential costs. As other solutions are low-cost options and the Tableau Server option has the potential to become unreasonable costly, extreme care was taken in ensuring we chose only the instance type required. The primary source of cost is from the hardware and OS requirements for Tableau. As of the start of

[2] "Review of Tableau by a Real User" by Bhosale, Dev
[3] "Leverage the power of Python in Tableau with TabPy" by Beran, Bora

our project, Tableau Server mandated a Windows OS. Tableau also required two physical cores, but it might be possible to work with virtual cores in lieu of physical cores. Experimentation is required to see how Tableau behaves in this situation. Tableau also required a number of other specifications, but the remaining specifications (e.g. RAM) were found to not hold a large financial footprint.

In the table below, we provide three columns of interest: Cost (24-hr), Cost (12-hr), and Cost (8-hr). In order to limit costs, we were looking at the option of only running the host VM for a limited number of hours a day. Enabling and disabling the VM would be handled by a lightweight script. Please note that one team member has extensive experience in creating such scripts so knowledge and skill were not an issue.

*Windows VM costs, 8/20/2017, all costs are 30-day approximates assuming the VM runs for x hours a day. All data was pulled from the official AWS site.*

| Instance | Risks Associated | Cost (24-hr) | Cost (12-hr) | Cost (8-hr) |
|---|---|---|---|---|
| **t2.xlarge** | • Requires EBS for storage | $164.88 | $82.44 | $54.96 |
| **m4.xlarge** | • Requires EBS for storage<br>• Insufficient physical cores, unknown issues likely | $276.48 | $138.24 | $92.16 |
| **m4.2xlarge** | • Requires EBS for storage<br>• Requires strict active hours management to limit costs | $552.96 | $276.48 | $184.32 |

### *Summary*

We chose Tableau Public over R Shiny due to time constraints and to build off of existing, functional, and refined work done in Tableau. This became a boon for us: due to this change, we were able to find a number of DQ issues through our EDA with Tableau Public.

## Designing and Building Dashboards

Building the dashboards required an understanding of the visualization capabilities of Tableau and how we wanted to probe into the data to uncover the answer to our research question.

We focussed on the following:
1.Strength of relationship between funding per student and test scores in math and reading per state
2. Regional variation and differences of strong vs weak funding and academic outcome relationships

3.Outliers

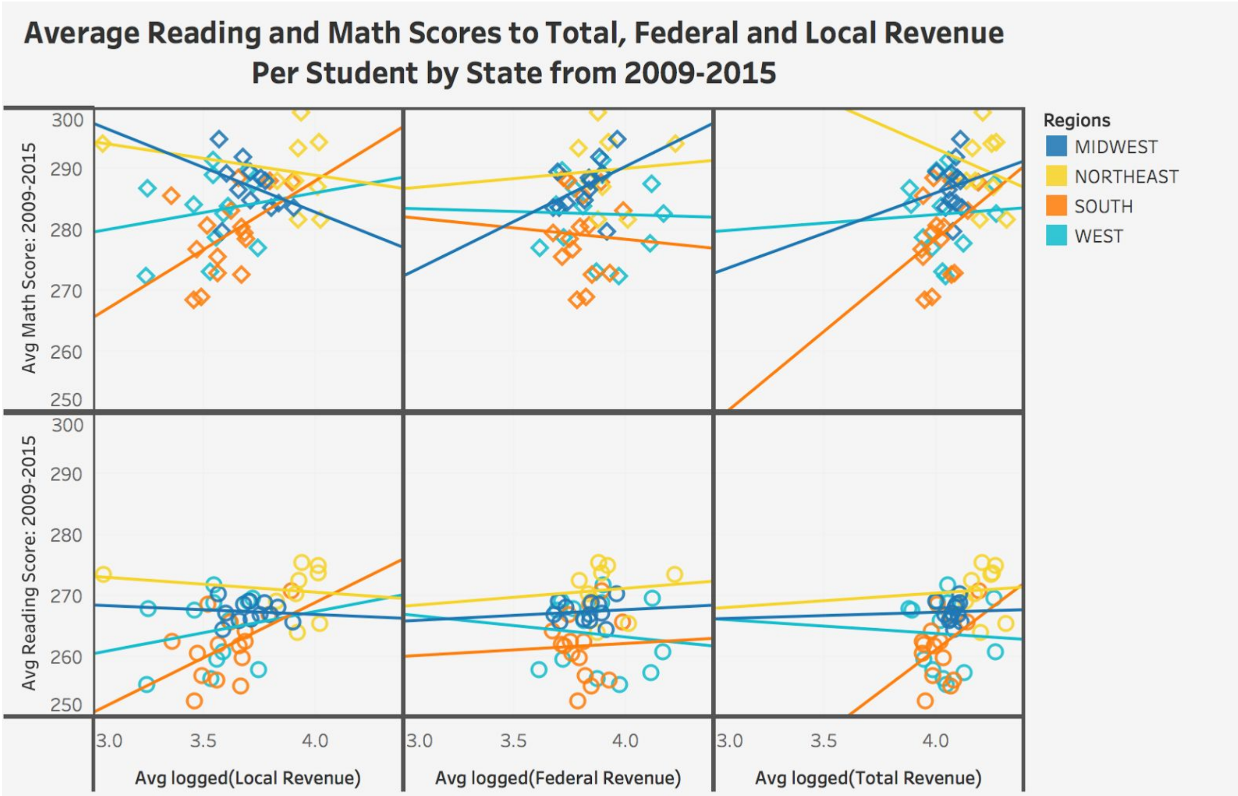
## Visualizations

We enriched our visualizations with proficiency and percentile rankings for both reading and math scores, US Regions and US districts. Additionally, we filtered and excluded fields and instances with poor data quality within Tableau and then implemented in our Postgres tables.
Metrics to note created in Tableau/Postgres:
Average Math and Reading Score across 2009-2013
Logged Total Revenue per Student
Logged Federal Revenue per Student
Logged Local Revenue per Student
Correlation between Math, Reading, Local Revenue and Total Revenue per Student
Teacher to Student Ratio
Revenue per Teacher

# Results

Our analysis found that there is a complex relationship between funding and academic outcomes. With evidence of a positive linear trend between total revenue and math/reading scores across states it would be unreasonable to dismiss funding as factor in academic outcomes for students. However this relationship was not statistically significant, but it is practically significant.

After a thorough review of the data through EDA and descriptive stats we formed the following conclusions:
While a relationship between between state funding and math/reading scores exists it is not a strong one due to unique outliers such as HI, AR, and AK, but there are strong and variable relationships between regional US educational funding and math/reading scores.



Average of loglocal, average of logfed and average of logtot vs. average of Math Score and average of Reading Score. Color shows details about Regions. Details are shown for state. The view is filtered on Exclusions (Regions,state), which keeps 49 members.

Additional resources were added to the dashboard to enrich the analysis:

- NAEP ranking of percentile
- Basic-proficient rank score
- US regions
- US Districts

Considered for future enrichment:

- Attributes of students, teachers, districts and states
- Median income of each state
- Political leanings of each state and district
- Ranked educational laws by state, region and country

Our research raised some additional questions:

Why are states like Alaska, Hawaii, Arkansas and Vermont outliers in local and total funding? While the south has shown the most improvement over the years, why, according to our data, is it consistently behind in math/reading scores and funding from other states?

The western states seem to be divided into high performing Northwestern states to low performing southwestern states in regards to math/reading scores. When thinking of policy changes should this region be subdivided at all times to ensure that we do not ultimately create laws that would increase performance from the southwest, but decrease performance from the northwest?
Could the Northwest be shown as performing better than the other regions due to a larger population and while it tends to allot more revenue for teachers than other regions could this be due to a higher standard income than other regions?

# Future Evolution of the Project

Given the opportunity to expand on our existing project, we would look at additional factors which contribute to academic outcome unrelated to educational funding as well as drilling down into more specific uses of educational funding to obtain insight.

## Identifying Other Factors Influencing Academic Outcomes

In this project we focused on local, state and federal revenue sources and test scores as our primary metrics. We also looked at class sizes and teacher salaries as results of funding contributing to academic success. It would be useful to examine other ways in which funding is used. Some examples include:

- property expenditure and maintenance
- facilities acquisition
- food services
- staff training
- support staff type and numbers

## Incorporating Additional Data Sources

Enrichment data sets which could provide some further insights include demographic data which may be obtained from the Center for Economic Studies census bureau or the Bureau of Labor Statistics Consumer Expenditure Survey to provide additional context to the educational outcomes.

## Handling an Evolving Dataset

When our project expands, we need to ensure our visualization infrastructure can handle any changes which might occur in data size as well as data variety. Assuming our project were to continue, we would likely onboard a variety of data sources which would provide us both finer-grained data as well as new data sources. These have been identified as the primary projects for how our data could change, that is, we expect changes in both volume and variety.

As our data grows more nuanced, access to a UI could prove useful to discovering new trends in our data. Due to this, if our data variety increases, Tableau variants pose substantial benefit due to Tableau's high accessibility and ease of creating dashboards.

If our data were to grow in volume, Tableau variants are likely able to bear the brunt up due to how Tableau handles data, especially in-memory data. If we reach a point, however, that the amount of memory Tableau consumes renders Tableau an unusable solution, we could opt for other solutions. For instance, we might find it useful to pre-compute some of our data to reduce the memory footprint.

## Analyzing Our Data in New Ways

Finer-grained data means we have more methods to analyze our data. Notably, it could be of use to employ more advanced cluster analysis methods and algorithms to identify trends which are not so readily available via the methods we use today. For instance, we might find it prudent to cluster students by socioeconomic data and check if there are other overarching trends present among those clusters which can be used to further refine our models.

We might also find use in the apriori algorithm as a niche alternative to traditional methods. If the apriori algorithm proves useful, Neo4j could also come into play to data mine our associations and how they interplay with each other.

# Works Cited

1. Gupta, Anand. "R Shiny v Tableau: Dawn of Graphics." LinkedIn, 25 Apr. 2016, https://www.linkedin.com/pulse/r-shiny-v-tableau-dawn-graphics-anand-gupta Accessed 21 Aug. 2017.
2. Beran, Bora. "Leverage the power of Python in Tableau with TabPy." *Tableau Software*, Tableau Software, 4 Nov. 2016, https://www.tableau.com/about/blog/2016/11/leverage-power-python-tableau-tabpy-62077. Accessed 21 Aug. 2017.
3. Bhosale, Dev. "Review of Tableau by a Real User." Unbiased reviews from the tech community, 22 Nov. 2016, https://www.itcentralstation.com/product_reviews/tableau-review-37875-by-dev-bhosale Accessed 22 Aug. 2017.
4. Jain, Rashmi. "A beginner's tutorial on the apriori algorithm in data mining with R implementation." *HackerEarth Blog*, HackerEarth, 3 Mar. 2017, https://www.itcentralstation.com/product_reviews/tableau-review-37875-by-dev-bhosale. Accessed 21 Aug. 2017.
5. "Shiny Server Professional v1.5.3 Administrator's Guide." Shiny Server v1.5.3 Configuration Reference, http://docs.rstudio.com/shiny-server/#system-requirements. Accessed 21 Aug. 2017.

6. “Amazon EC2 Instance Types – Amazon Web Services (AWS).” Amazon Web Services, Inc., https://aws.amazon.com/ec2/instance-types/. Accessed 20 Aug. 2017.
7. “EC2 Instance Pricing – Amazon Web Services (AWS).” Amazon Web Services, Inc., https://aws.amazon.com/ec2/pricing/on-demand/. Accessed 20 Aug. 2017.
8. “Is there a limit on the number of concurrent connections per server for Shiny Server?” RStudio Support, https://support.rstudio.com/hc/en-us/articles/218294957-Is-there-a-limit-on-the-number-of-concurrent-connections-per-server-for-Shiny-Server-. Accessed 22 Aug. 2017.