# DAVE

Data Analytical Visualization with Ease

James HyunSeung Hong (hh2473)
Min Woo Kim (mk3351)
Fan Yang (fy2207)
Chen Yu (cy2415)

Programming Languages and Translators
COMS W4115, Fall 2015
Project Proposal

# Introduction

DAVE is a programming language optimized for data retrieval, manipulation, analysis and visualization. We designed DAVE with cross-dataset operations in mind, which is fairly common yet complicated to achieve with current toolsets in today's data-intensive tasks. Operators would be able to use DAVE to validate datasets, incorporate (parts of) datasets from different sources, split up oversized datasets, conduct statistical analysis and visualize critical data.

# Motivation

For the last few decades developers have created a variety of tools to help with data management and analysis. SQL and R language, for example, are two of the most prominent tools in this field, being widely utilized by business analysts, social scientists, statisticians and many more. Popular as they are, there still remain certain scenarios where both tools are imperfect. For instance, social scientists today often need to incorporate data of distinct formations from different sources for their statistical analysis. SQL may work well for this job, yet it is designed for relational databases, a system which is not suitable for most statistical tasks. R language, on the contrary, is fully optimized for statistical analysis with all the functions and libraries, but it lacks the simplicity and usability of SQL in terms of managing data. That is exactly why we are building DAVE to address this problem. By implementing **rec, fld and tbl** data structures and adopting a C-style syntax, our language offers great assistance in cross-dataset operations; the addition of statistical analysis functions also grants DAVE capabilities to analyze and visualize data. As the libraries keep expanding, DAVE is sure to achieve more.

# Language Overview

## Datatypes and Operators

DAVE has mainly four fundamental kinds of datatypes and three DAVE-specific types. The first four are the character string, 32-bit integer, boolean and 32-bit floating point number. The new data types, tbl, rec and fld are implemented to enable a statistical manipulation of data.

For the operators, DAVE supports arithmetic operators, relational operators, logical operators, and assignment operators in a similar fashion as other commonly used programming languages.

### List of Datatypes

| | |
|---|---|
| str | sequence of characters, closed with single or double quotes |
| int | 32-bit integer |
| float | 32-bit floating-point number |
| bool | boolean value |
| tbl | data table that consists of collection of fields and records |
| rec (record) | sequential collection of heterogeneous variables |
| fld (field) | sequential collection of homogeneous variables |
| none | null type |

### List of Operators

| | |
|---|---|
| - + / * % ^ | minus, plus, divide, multiply, modular, exponent |
| = | assignment to an identifier |
| > | greater than |
| >= | greater than or equal to |
| < | less than |
| <= | less than or equal to |
| == | equal to |
| != | not equal to |
| && | and |
| \|\| | or |
| ! | not |
| if, else if, else | conditional |

| | |
|---|---|
| for, while | loop, iteration |
| /* */ | comment, all characters inside /* and */ are considered comments |

## Functions

DAVE's built-in functions comprise primarily of file I/O stream, data manipulation, and graphical modeling. Some commonly used statistical methods such as linear/quadratic regression are also included.

### List of Functions

| | |
|---|---|
| load(str filepath,<br>str[] field_list,<br>int[] record_list) | to load a table from a path.<br>argument:<br>filepath: str - path of file to load table from (required)<br>field_list: str[] - list of field names to fetch (default all)<br>record_list: int[] - list of record indexes to fetch (default all) |
| save(str filepath,<br>tbl table,<br>str option) | to save a table to a path<br>argument:<br>filepath: str - path of file to save table to (required)<br>table: tbl - table to save to a file (required)<br>option: str - 'a' to append to file, 'o' to overwrite (default 'o') |
| printf(str, additional arguments) | to print its argument to standard output |
| plotf(str, additional arguments) | to plot the corresponding diagram |
| comp(fld, fld) or comp(rec, rec) | to compare two fields or records with identical data types |
| append(tbl, fld/rec) | to add new entities following in the end of table |
| max(fld), min(fld), sum(fld) | to return the max, min, or sum of the records in given field |
| linreg(fld x, fld y) | to calculate the linear regression of fld x and fld y, and return a new fld with revelant parameters (slope, intercept, t-value, confidence level, $R^2$, etc.) |
| plotline(int k, int b) | to plot the corresponding straight line, with k is the slope and b is the intercept |
| quadreg(fld x, fld y) | to calculate the quadratic regression of fld x and fld y, and return a new fld with constants in the quadratic equation |

### Visualization

DAVE supports a built-in plot function which formats its argument into a python executable data set and prints the diagram as an image file. The central idea is to enable efficient plotting of DAVE-specific data types, fld. An example of visualization is provided below.

# Example Program

Here's an example program for DAVE language, which includes the process for importing data from the original text file, performing data analytics, visualizing data, adding new data, and saving the updated data.

### Database Example (w4115_roster.txt)

```
<!TABLETYPE dave>
str name; int age; str gender; bool is_enrolled; int score;
Emily; 24; F; true; 90;
James; 22; M; true; 80;
Min Woo; 23; M; true; 95;
Jenny; 19; F; false; 65;
<ENDTABLE>
```

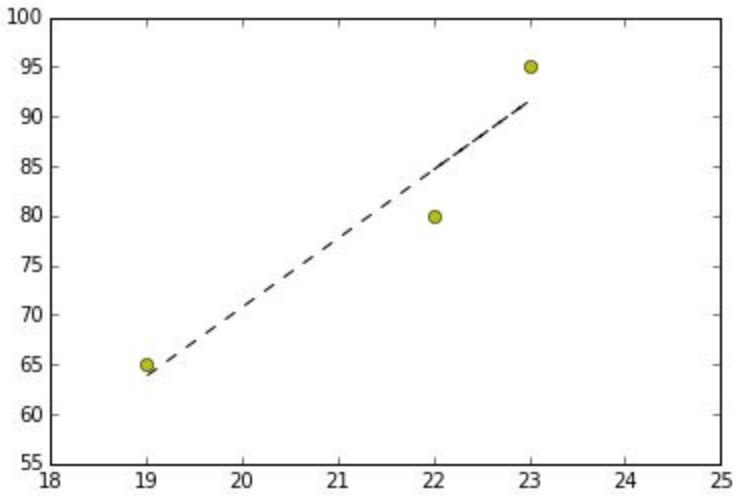
### Input:

```
int main(int argc, char **argv) {
    /* read database from w4115_roster.txt above,
        only for fields 'age', age, 'score' and records 1 to end */
    tbl roster = read('w4115_roster.txt', ['name', 'age', 'score'], [1:]);
    printf("%tbl", roster);

    /* visualize linear regression on the selected data */
    fld ages = roster.age;
    fld scores = roster.score;
    fld[] result = linreg(ages, scores);
    plotf("%fld %fld", result[0], result[1]);

    /* create new record for a student,
        with name 'Michael' and age from first index of ages */
```

```
    rec new_student = {name: ‘Michael’, age: ages[0]};

    /* add new student record to the table */
   append(roster, new_student);

    /* save updated roster to w4115_roster_v2.txt */
    save(‘w4115_roster_v2.txt’, roster);

    return 0;
}
```

## Output:

```
> str name; int age; int score;
   James; 22; 80;
   Min Woo; 23; 95;
   Jenny; 19; 65;

>
```




## w4115_roster_v2.txt:

```
<!TABLETYPE dave>
str name; int age; int score;
James; 22; 80;
Min Woo; 23; 95;
Jenny; 19; 65;
Michael; 22;;
<ENDTABLE>
```