# Data Visualization- Chronic Disease Report Generator (v0.1.8)

This document details the components of the Chronic Disease Report Generator v0.1.8. Some of the documentation comes straight from the code.

## Current Features:

* Reads CSV reports from PSS specific to diabetes
* Displays % patients meeting quality criteria
  + Horizontal bar charts with all criteria shown
* Can import multiple reports to show trends
  + Reports displayed as line charts by quality criteria – 1 criteria at a time
* Can filter by Doctor
* Can save charts as png files

Table of Contents

[Data Visualization- Chronic Disease Report Generator (v0.1.8) 1](#_Toc387912141)

[Current Features: 1](#_Toc387912142)

[Screenshots 3](#_Toc387912143)

[Files and Functions 5](#_Toc387912144)

[reportGenerator.js 5](#_Toc387912145)

[~~initializeReportGenerator (mode)~~ 5](#_Toc387912146)

[readFiles(files) 5](#_Toc387912147)

[readSingleFile(index, mode) 5](#_Toc387912148)

[cleanParsedData() 6](#_Toc387912149)

[userInterface.js 7](#_Toc387912150)

[addSidePanels() 7](#_Toc387912151)

[toggleSelected() 7](#_Toc387912152)

[~~toggleDataLabels()~~ //not used 7](#_Toc387912153)

[calculateAndGenerate() 7](#_Toc387912154)

[dataManipulation.js 8](#_Toc387912155)

[filterData() 8](#_Toc387912156)

[calculateDataSnapshotMode() 8](#_Toc387912157)

[calculateDataTrackingMode() 8](#_Toc387912158)

[calculateCountDiabeticMeasure(fileIndex, measureIndex) 8](#_Toc387912159)

[generateVisualization.js 9](#_Toc387912160)

[clearCanvas() 9](#_Toc387912161)

[generateVisualizationSnapshowMode() 9](#_Toc387912162)

[generateVisualizationTrackingMode() 9](#_Toc387912163)

[toggleDataLabels() 9](#_Toc387912164)

[Bugs and Issues 10](#_Toc387912165)

[Feature Requests 11](#_Toc387912166)

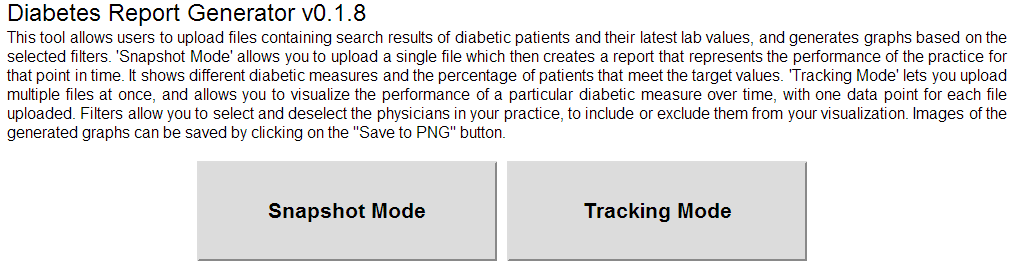
Screenshots

Figure 1. Initial welcome screen with option to select modes “snapshot” or “tracking”.

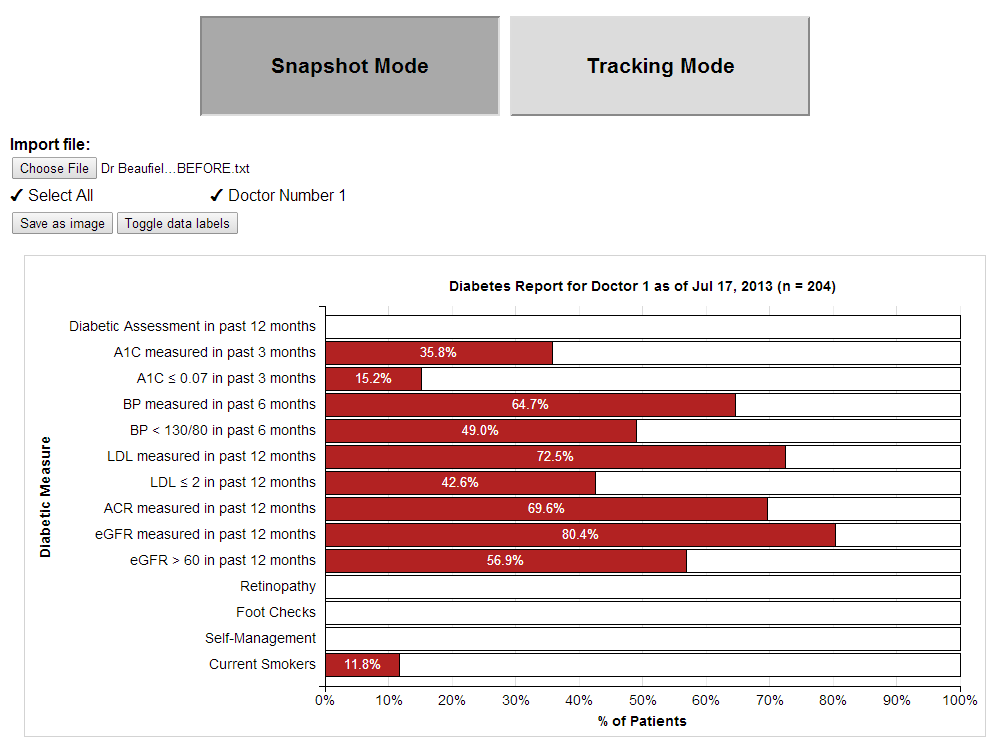


Figure 2. Snapshot mode selected, with one report selected. This report contains information for only one doctor.

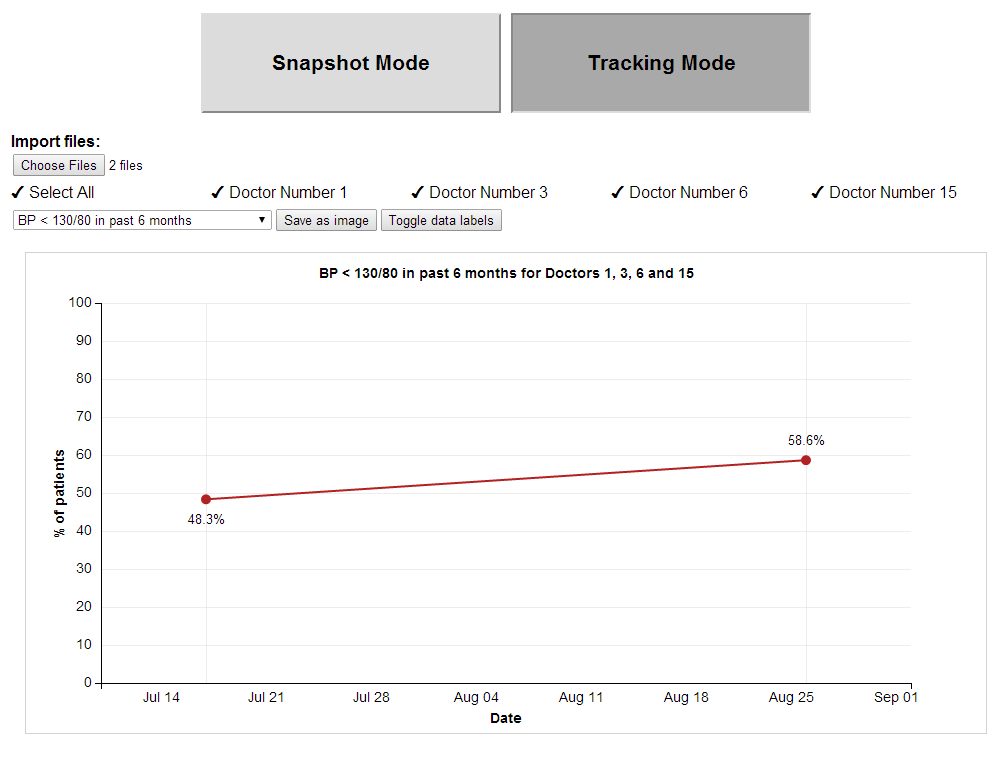


Figure 3. Tracking mode selected. Two reports were selected. The chart shows an increase in diabetic patients with their blood pressure in normal ranges.

# Files and Functions

## reportGenerator.js

Sets up the interface for the appropriate mode (“single” or “tracking”), adds the GUI elements to read in files. Reads, parses and cleans the files, calls appropriate functions with parsed data to generate the graphs.

initializeReportGenerator (mode) //mode one of “tracking” or “snapshot”

* Sets appropriate appearance of **buttonSnapshot** and **buttonTracking**
* Sets contents of **sectionImport** div inside of **sidePanel** to nothing. sectionImport contains a button to load files. \*\*No check to make sure .csv file\*\*
* Removes imported sections **physicianSection, measuresSection, settingsSection, canvasSVG**
* Adds **sectionImport** HTML which allows the ability to import files (single or multiple)
  + Calls readFiles with the files selected by the user
* Created **canvasSVG** from d3.js, sets height, width, and border.

### readFiles(files)

* Checks to make sure files were selected
* No error checking (!!)
* Uses function readSingleFile. Passes file and current mode (“tracking” or “snapshot”)

### readSingleFile(index, mode)

* Checks to make sure more files need to be read, if not calls:
  + cleanParsedData()
  + addSidePanels() //userInterface.js
  + filterData() //dataManipulation.js
  + if “snapshot”
    - calculateDataSnapshotMode() //dataManipulation.js
    - clearCanvas() //generateVisualization.js
    - generateVisualizationSnapshotMode()//generateVisualization.js
  + else if “tracking”
    - calculateDataTrackingMode() //dataManipulation.js
    - clearCanvas() //generateVisualization.js
    - generateVisualizationTrackingMode()//generateVisualization.js
  + Create FileReader(), use readAsText(file) with the d3.csv parser
  + Initialize the arrayLastModifedDate for that file from the time the file was last modified (i.e. the report was generated)

### cleanParsedData()

**Function**: Ensure Current Date is one of the columns, otherwise add it and insert a date value (arrayLastModifiedDate). If Current Date is one of the columns, parse the date format to a different format.

* for each file read in:
  + remove the header
  + for each row in the file:
    - if the last column is empty, remove it. If not, function ends (??)
    - is there a Current Date column?
      * NO: add it to the header and insert arrayLastModifiedDate as the date in each row
      * YES: ignore header, parse current date to different format (defined in file)

## userInterface.js

It creates different sections of the panel to select different filters, such as selecting by physician, selecting by diabetic measure, and the settings menus. The filters in each panel are created dynamically to only contain information that exists in the imported files. It also contains the code to update the filters based on user input, such as changing the classes of the filters based on onClick events fired in the document.

### addSidePanels()

* Removes previous GUI elements and replaces them with appropriate elements. Namely:
  + physicianSection – at top of chart, allows user to select which physicians to display date for.
  + settingsSection – contains buttons to save chart as image and toggle data labels on and off
  + measuresSection – in “tracking” mode only – dropdown menu which allows user to select which measure to display on the chart

1. physicianSection
   1. unordered list generated from physician numbers read in files, sorted by doctor #
      1. set onClick action to **toggleSelected()**
2. measuresSection (only in “tracking” mode)
   1. set onChange to **calculateAndGenerate()**
   2. dropdown generated manually from global variables
3. settingsSection
   1. add two buttons to save an image and toggle date labels (calls **toggleDateLabels()**)

### toggleSelected()

* updates CSS elements to reflect whether they are currently checked depending on which item was selected (whether the “select all” checkbox or the individual physicians’ checkboxes)
* updates arraySelectedPhysicians to reflect changes
  + Current done by inspecting the HTML and doing string comparisons (!?) **may need to change**
* filterData()
* calculateAndGenerate()

### toggleDataLabels() //not used

* if labels on chart are visible, remove them using d3 APIs.
* If labels on chart are not visible:
  + Enter the data element of the canvas and add a text element containing the label

### calculateAndGenerate()

* if “snapshot”
  + calculateDataSnapshotMode() //dataManipulation.js
  + clearCanvas() //generateVisualization.js
  + generateVisualizationSnapshotMode()//generateVisualization.js
* else if “tracking”
  + calculateDataTrackingMode() //dataManipulation.js
  + clearCanvas() //generateVisualization.js
  + generateVisualizationTrackingMode()//generateVisualization.js

## 

## dataManipulation.js

### filterData()

Filters data based on the selected physicians and populates 'arrayFilteredData' with the filtered rows from each CSV file.

* Create a new variable to hold arrayFilteredData
* Loop through each row in arrayCleanedData
* Push the header onto arrayFilteredData
* For each row in arrayCleanedData:
  + Get the doctor number
  + Get doctor index from unique physician array
  + Check if the doctor is selected in arraySelectedPhysicians
  + If the doctor is selected, add this row to arrayFilteredData.
* Return arrayFilteredData

### calculateDataSnapshotMode()

Calculates data for Snapshot mode. Loops through each patient of filtered physicians and increment diabetic measure counters as needed. The function **calculateCountDiabeticMeasure(fileIndex, measureIndex)** is used to calculate the number of patients in each category. An array, **arrayCalculatedData**, is created containing the row names and calculated values.

### calculateDataTrackingMode()

Calculates data for Tracking mode. Retrieves index of diabetic measures drop down menu and create array of counts for that measure.

arrayDates is created which gets the data from the first row of the Current Date column in each file.

### calculateCountDiabeticMeasure(fileIndex, measureIndex)

Functions for calculating patients counts for specific diabetic measures.

**fileIndex**: index into **arrayFilteredData** corresponding to a specific file. 0 if only one file

**measureIndex**: index into each row (i.e. patient)

* loops through each row of a file specified by **fileIndex**.
* Selects an element in the row specified by **measureIndex**
* If the element meets the inclusion criteria, adds it to a **count** variable
* Returns the **count** for storage in **arrayCalculatedData**

## generateVisualization.js

Accepts the **arrayCalculatedData** variable and uses the d3.js library to generate the visualization.

### clearCanvas()

removes the **canvasSVG** element from the interface and replaces it with a fresh one.

### generateVisualizationSnapshowMode()

* Configures chart axis (scale and ticks).
* Adds bars based on the values in the **arrayCalculatedData** variable.
* Adds labels to each bar display % of patients meeting each criteria.
* Adds a title and axis labels to the graph

### generateVisualizationTrackingMode()

* Generates range of dates in files using **arrayDates**
* Configures xaxis with date range, yaxis with 0-100. Adds tick lines.
* Add lines between data points (.dataPointConnector) and data points (.dataPoint)
* Adds mouse over effects to the data points.
  + **TODO**: Add a graph for the specific date when a data point is clicked.
* Add x and y axis labels, add a title.
* Add data point labels.

### toggleDataLabels()

similar function exists in userInterface.js. this function may supercede it.

* If data labels exist, remove them
* If not, check mode (“snapshot” or “tracking”)
* In snapshot, generate labels containing % on target and % off target. Only display labels for % on target
* In tracking, generate and add labels.

# Bugs and Issues

1. Some files do not produce a chart with no errors or warnings displayed to user
2. Use of many global variables, potentially unnecessary and confusing. May need to change to support multiple report types.

# Feature Requests

1. Add support for other chronic diseases such as hypertension
   1. Wrap diabetes part in higher level user interface
   2. **OR**, duplicate program with support for hypertension instead (at different urls?)
2. Automate finding and selecting of text files to reduce need for user input
3. Add the ability to generate “snapshot” charts by click on points in “tracking” chart.
4. Resize canvas as a percentage of window?