function replaceNulls(data) {

for (var i=0; i<data.length; i++) {

if (data[i].countofcompleteddate === 'null') {

data[i].countofcompleteddate = '0';

}

if (data[i].numberofremainingitems === 'null') {

data[i].numberofremainingitems = '0';

}

// convert to int

data[i].countofcompleteddate = parseInt(data[i].countofcompleteddate);

data[i].numberofremainingitems = parseInt(data[i].numberofremainingitems);

}

}

function getThroughput(data) {

var throughput = [];

for (var i = 0; i < data.length; ++i) {

throughput.push(data[i].countofcompleteddate);

}

return throughput;

}

function time\_for\_jobs(items, throughput) {

// Randomly selects a number of jobs

var count = 0;

var loop = true;

while (loop){

items -= throughput[Math.floor(Math.random()\*throughput.length)]

count += 1;

if (items <= 0){

loop = false;

}

}

return count;

}

function monte\_carlo(iterations, items, throughput) {

// Check if the items and throughput arrays are empty or undefined

if (!items || items.length === 0 || !throughput || throughput.length === 0) {

console.log("Error: Items array or throughput array is empty or undefined.");

return [];

}

var output = [];

for (var i = 0; i < iterations; i++) {

output.push(time\_for\_jobs(items, throughput));

}

return output;

}

function freq\_table(output){

// Creates a frequency table from MC output

var result = {};

for (var i=0; i<d3.max(output); i++){

result[i] = 0; // Initialise to zeros

}

for(var i = 0; i < output.length; ++i) {

if (!result[output[i]]) {

result[output[i]] = 0;

}

++result[output[i]]; // Increment

}

return result;

}

function format(data){

// Gets monte carlo frequency table in correct plotting format

var final = [];

Object.entries(data).forEach(function(key, val) {

final.push({

"num" : key[0],

"freq" : key[1]

});

});

return final;

}

function sortNumber(a, b) {

return a - b;

}

function validation(samples){

samples.sort(sortNumber);

var group1Sum = 0;

var group1Count = 0;

var group2Sum = 0;

var group2Count = 0;

var range = Math.max.apply(null, samples) - Math.min.apply(null, samples);

for(var i = 0; i < samples.length; ++i){

if (i % 2 == 0) {

group1Sum += samples[i];

group1Count ++;

} else {

group2Sum += samples[i];

group2Count ++;

}

}

var group1avg = group1Sum / group1Count;

var group2avg = group2Sum / group2Count;

var avgError = Math.abs(group1avg - group2avg);

var stability = (avgError / range) \* 100;

return stability.toFixed(1);

}

function calc\_median(numbers) {

// median of [3, 5, 4, 4, 1, 1, 2, 3] = 3

var median = 0, numsLen = numbers.length;

numbers.sort(sortNumber);

if (

numsLen % 2 === 0 // is even

) {

// average of two middle numbers

median = (numbers[numsLen / 2 - 1] + numbers[numsLen / 2]) / 2;

} else { // is odd

// middle number only

median = numbers[(numsLen - 1) / 2];

}

return median;

}

function calc\_percentile(numbers, perc) {

var numsLen = numbers.length;

var percentile\_index = (perc \* numsLen)-1;

numbers.sort(sortNumber);

//percentile = Math.ceil((numbers[Math.floor(percentile\_index)] + numbers[Math.ceil(percentile\_index)]) / 2);

var percentile = Math.ceil(numbers[Math.floor(percentile\_index)] + (numbers[Math.ceil(percentile\_index)] - numbers[Math.floor(percentile\_index)]) \* percentile\_index % 1);

return percentile;

}

function x\_ticks(x\_vals, delta){

// removes every delta'th element for x axis

var domain = [];

for (var i=0; i<x\_vals.length; i=i+delta){

domain.push(x\_vals[i]);

}

return domain;

}

// Define margin, width and height of box

var margin = {top: 20, right: 75, bottom: 50, left: 80},

width = pbi.width - margin.left - margin.right,

height = pbi.height - margin.top - margin.bottom;

var bar\_gap = 0.1;

// Define scales

var x = d3.scale.ordinal()

.domain([0,width])

.rangeRoundBands([0, width], bar\_gap, 0);

var y = d3.scale.linear()

.range([height, 0]);

// Create chart based on margin, width, height

var svg = d3.select("#chart")

.attr("width", width + margin.left + margin.right)

.attr("height", height + margin.top + margin.bottom)

.append("g")

.attr("transform", "translate(" + margin.left + "," + margin.top + ")");

// Read in data

pbi.dsv(function(data) {

// Initialize parameters

replaceNulls(data);

var throughput = getThroughput(data);

var projection\_items = Math.max.apply(Math, data.map(function(o) { return o.numberofremainingitems;})); // select max as nulls are replaced with 0s.

var iterations = 10000;

// Run the simulation

var stability = validation(throughput);

var mc = monte\_carlo(iterations, projection\_items, throughput);

var ft = freq\_table(mc);

var histogram = format(ft);

// Determine percentiles

var median = calc\_median(mc);

var seventy\_percent = calc\_percentile(mc, 0.7);

var seventy\_five\_percent = calc\_percentile(mc, 0.75);

var eighty\_five\_percent = calc\_percentile(mc, 0.85);

var ninety\_five\_percent = calc\_percentile(mc, 0.95);

// Domains of each axis

x.domain(Object.keys(ft));

y.domain([0, d3.max(Object.values(ft))]);

var \_interval = Math.round(Object.keys(ft).length / 7);

// Add axes to graph

svg.append("g")

.attr("class", "x axis")

.attr("transform", "translate(0," + height + ")")

.call(d3.svg.axis()

.tickValues(x\_ticks(Object.keys(ft), \_interval)) // Determines x-axis skips (arg 2)

.scale(x)

.orient("bottom"));

// text label for the x axis

svg.append("text")

.attr("transform",

"translate(" + (width/2) + " ," +

(height + margin.top + 20) + ")")

.style("text-anchor", "middle")

.text("Weeks to complete "+projection\_items.toString()+" work items | Stability = " + stability.toString() + "% (keep below 25%)")

.style("font-size", "12px")

.attr("class", "label xAxisLabel");

svg.append("g")

.attr("class", "y axis")

.call(d3.svg.axis()

.scale(y)

.orient("left"));

// text label for the y axis

svg.append("text")

.attr("transform", "rotate(-90)")

.attr("y", 0 - margin.left)

.attr("x",0 - (height / 2))

.attr("dy", "1em")

.style("text-anchor", "middle")

.text("Frequency")

.style("font-size", "12px")

.attr("class", "label yAxisLabel");

// Add data to the graph from Monte Carlo histogram

svg.selectAll(".bar")

.data(histogram)

.enter()

.append("rect")

//.attr("class", "bar")

.attr("x", function(d) { return x(d.num)})

.attr("width", x.rangeBand())

.attr("y", function(d) { return y(d.freq)})

.attr("height", function(d) { return height - y(d.freq); })

.attr("class", function(d) {

if (d.num <= median) {

return "bar barRed";

} else if (d.num > median & d.num <= seventy\_percent) {

return "bar barOrange";

} else if (d.num > seventy\_percent & d.num < eighty\_five\_percent) {

return "bar barYellow";

} else {

return "bar barGreen";

}

});

// Add the percentile lines on the graph

// var group = svg.append("g");

// var line = d3.svg.line()

// .x(function(d){ return d.x})

// .y(function(d){ return d.y})

// var zeroth\_pxl = x(histogram[0].num); // Pixel position of first bar

// var pxl\_diff = x(histogram[1].num) - zeroth\_pxl; // Rate of pixel increase per bar

// var bar\_centre = pxl\_diff/2;

// Percentile data

// var data\_50 = [ // 0.5 added to px\_diff for the mean

// {x: zeroth\_pxl+bar\_centre+(pxl\_diff\*median), y:10},

// {x: zeroth\_pxl+bar\_centre+(pxl\_diff\*median), y:height}

// ]

// var data\_85 = [ // 0.85 added to px\_diff for the 85th %ile

// {x: zeroth\_pxl+bar\_centre+(pxl\_diff\*eighty\_five\_percent), y:10},

// {x: zeroth\_pxl+bar\_centre+(pxl\_diff\*eighty\_five\_percent), y:height}

// ]

// var data\_95 = [ // 0.95 added to px\_diff for the 95th %ile

// {x: zeroth\_pxl+bar\_centre+(pxl\_diff\*ninety\_five\_percent), y:10},

// {x: zeroth\_pxl+bar\_centre+(pxl\_diff\*ninety\_five\_percent), y:height}

// ]

// Percentile lines

// group.selectAll("path")

// .data([data\_50, data\_95, data\_85])

// .enter()

// .append("path")

// .attr("d", line)

// .attr("stroke", "black")

// .attr("fill", "none")

// .attr("stroke-width", 1)

// .attr("stroke-dasharray", "5,5")

// Labels for 50th percentile

// group.append("text")

// .attr("dx", zeroth\_pxl+bar\_centre+(pxl\_diff\*median)-10) // Minus 10 for centering

// .attr("dy", 0)

// .text("50%")

// .style("font-size", "10px")

// .attr("class", "label fiftyPerc");

// Labels for 85th percentile

// group.append("text")

// .attr("dx", zeroth\_pxl+bar\_centre+(pxl\_diff\*eighty\_five\_percent)-10) // Minus 10 for centering

// .attr("dy", 0)

// .text("85%")

// .style("font-size", "10px")

// .attr("class", "label eightyFivePerc");

// Labels for 95th percentile

// group.append("text")

// .attr("dx", zeroth\_pxl+bar\_centre+(pxl\_diff\*ninety\_five\_percent)-10) // Minus 10 for centering

// .attr("dy", 0)

// .text("95%")

// .style("font-size", "10px")

// .attr("class", "label ninetyFivePerc");

});