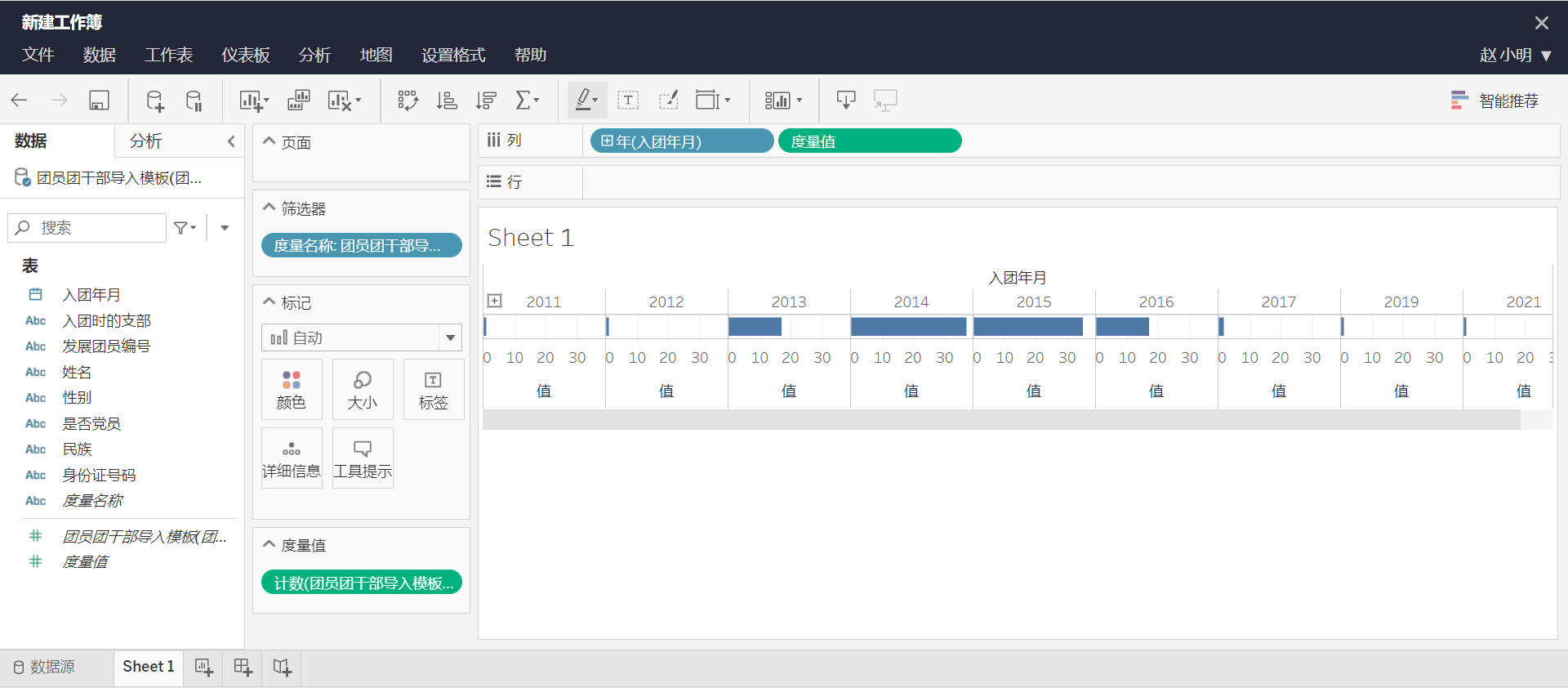
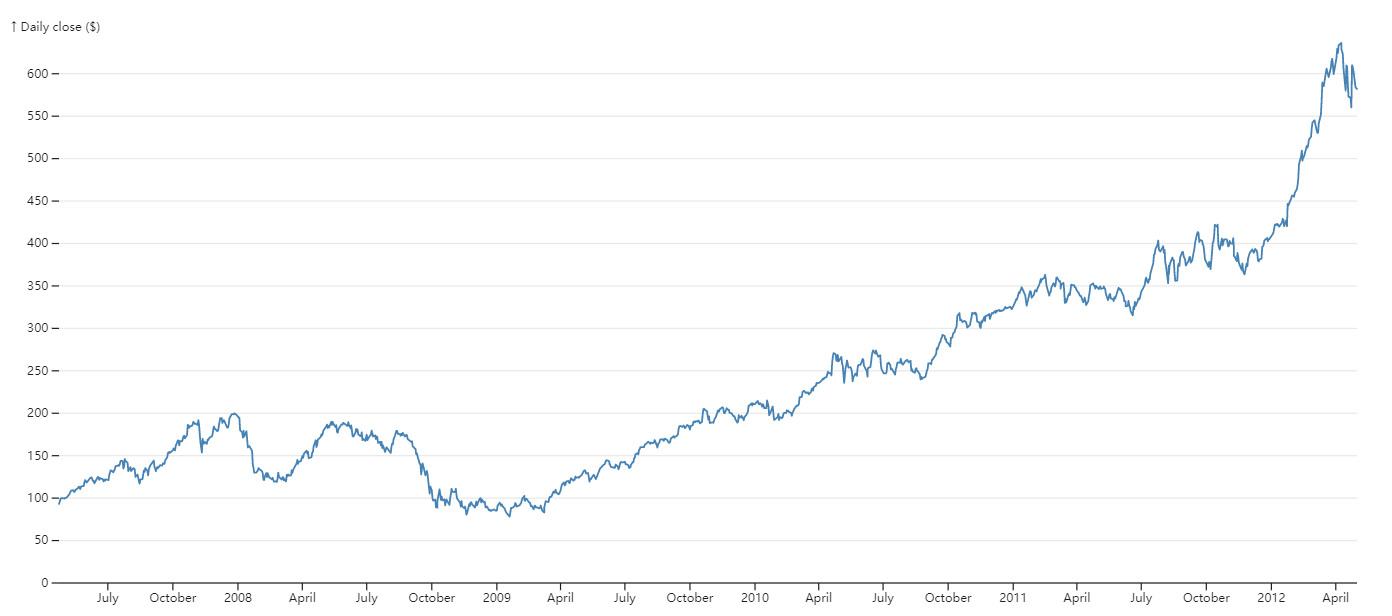
1.使用Tableau对班级同学入团年份进行统计



2.使用echarts.js绘制一个简单的柱状图



3.使用d3.js折线图样例



function LineChart(data, {

x = ([x]) => x, // given d in data, returns the (temporal) x-value

y = ([, y]) => y, // given d in data, returns the (quantitative) y-value

defined, // for gaps in data

curve = d3.curveLinear, // method of interpolation between points

marginTop = 20, // top margin, in pixels

marginRight = 30, // right margin, in pixels

marginBottom = 30, // bottom margin, in pixels

marginLeft = 40, // left margin, in pixels

width = 640, // outer width, in pixels

height = 400, // outer height, in pixels

xType = d3.scaleUtc, // the x-scale type

xDomain, // [xmin, xmax]

xRange = [marginLeft, width - marginRight], // [left, right]

yType = d3.scaleLinear, // the y-scale type

yDomain, // [ymin, ymax]

yRange = [height - marginBottom, marginTop], // [bottom, top]

yFormat, // a format specifier string for the y-axis

yLabel, // a label for the y-axis

color = "currentColor", // stroke color of line

strokeLinecap = "round", // stroke line cap of the line

strokeLinejoin = "round", // stroke line join of the line

strokeWidth = 1.5, // stroke width of line, in pixels

strokeOpacity = 1, // stroke opacity of line

} = {}) {

// Compute values.

const X = d3.map(data, x);

const Y = d3.map(data, y);

const I = d3.range(X.length);

if (defined === undefined) defined = (d, i) => !isNaN(X[i]) && !isNaN(Y[i]);

const D = d3.map(data, defined);

// Compute default domains.

if (xDomain === undefined) xDomain = d3.extent(X);

if (yDomain === undefined) yDomain = [0, d3.max(Y)];

// Construct scales and axes.

const xScale = xType(xDomain, xRange);

const yScale = yType(yDomain, yRange);

const xAxis = d3.axisBottom(xScale).ticks(width / 80).tickSizeOuter(0);

const yAxis = d3.axisLeft(yScale).ticks(height / 40, yFormat);

// Construct a line generator.

const line = d3.line()

.defined(i => D[i])

.curve(curve)

.x(i => xScale(X[i]))

.y(i => yScale(Y[i]));

const svg = d3.create("svg")

.attr("width", width)

.attr("height", height)

.attr("viewBox", [0, 0, width, height])

.attr("style", "max-width: 100%; height: auto; height: intrinsic;");

svg.append("g")

.attr("transform", `translate(0,${height - marginBottom})`)

.call(xAxis);

svg.append("g")

.attr("transform", `translate(${marginLeft},0)`)

.call(yAxis)

.call(g => g.select(".domain").remove())

.call(g => g.selectAll(".tick line").clone()

.attr("x2", width - marginLeft - marginRight)

.attr("stroke-opacity", 0.1))

.call(g => g.append("text")

.attr("x", -marginLeft)

.attr("y", 10)

.attr("fill", "currentColor")

.attr("text-anchor", "start")

.text(yLabel));

svg.append("path")

.attr("fill", "none")

.attr("stroke", color)

.attr("stroke-width", strokeWidth)

.attr("stroke-linecap", strokeLinecap)

.attr("stroke-linejoin", strokeLinejoin)

.attr("stroke-opacity", strokeOpacity)

.attr("d", line(I));

return svg.node();

}