Boxplot

<http://bl.ocks.org/jensgrubert/7789216>

<http://informationandvisualization.de/blog/box-plot>

add new type plot for vega

vvage-lite>src>compile>mark

boxplot :68

modeskew = (mean- median)/stdev;

<http://mathworld.wolfram.com/PearsonsSkewnessCoefficients.html>

ROBPCA : skewness + outlinear

Outliner

https://www.youtube.com/watch?v=bas3-Ue2qxc

Final goal:

abstraction level

learning user behavior

# Scagnostics

<script type="text/javascript" src="https://idatavisualizationlab.github.io/Scagnostics2018/scagnostics/build/js/scagnostics.min.js"></script>

msg += "<br/>1. Outlying score: " + scag.outlyingScore + ", outlying edge cut point: " + scag.outlyingUpperBound;

//Skewed

msg += "<br/>2. Skewed score: " + scag.skewedScore;

//Sparse

msg += "<br/>3. Sparse score: " + scag.sparseScore;

//Clumpy

msg += "<br/>4. Clumpy score: " + scag.clumpyScore;

//Striated

msg += "<br/>5. Striated score: " + scag.striatedScore;

//Convex

msg += "<br/>6. Convex score: " + scag.convexScore;

//Skinny

msg += "<br/>7. Skinny score: " + scag.skinnyScore;

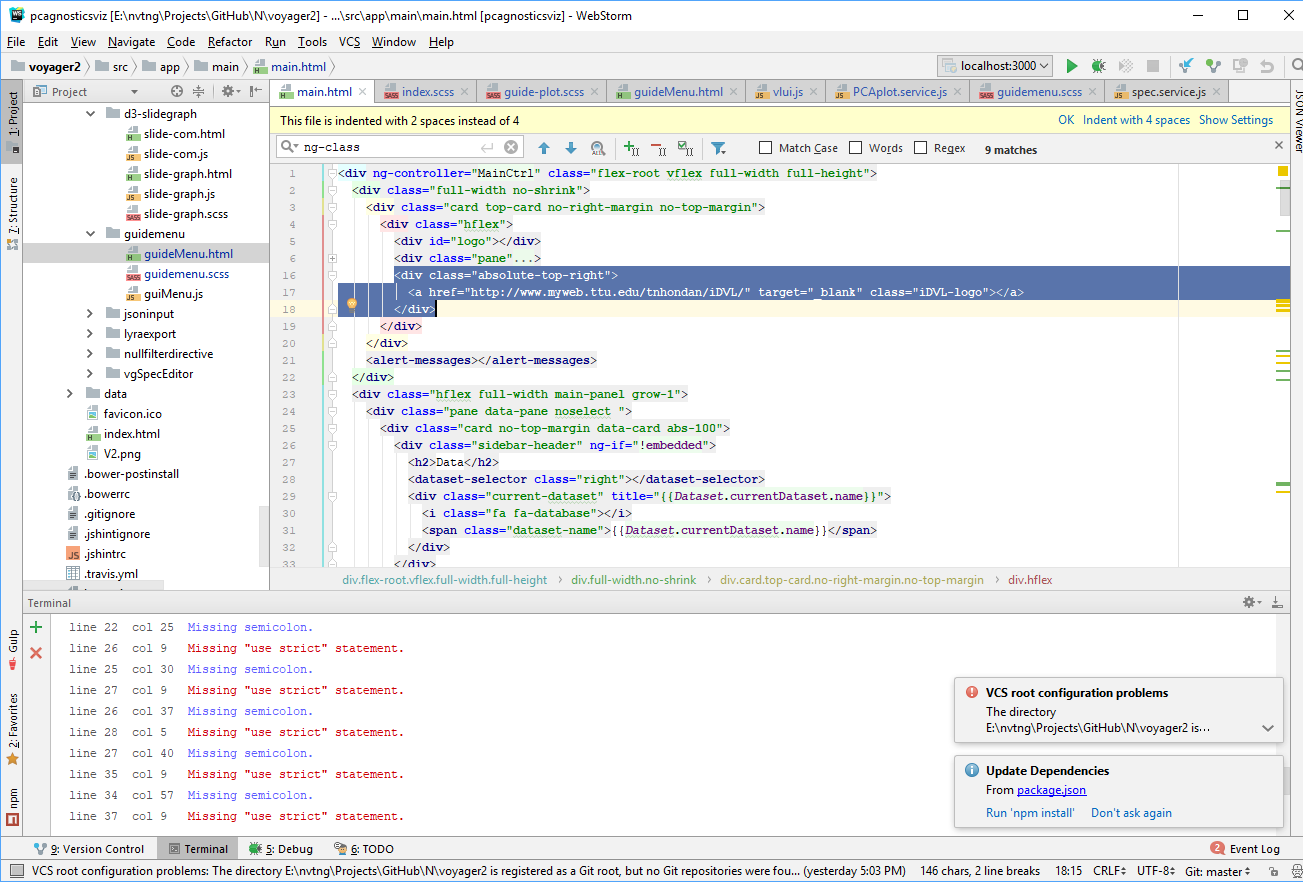
//Stringy

msg += "<br/>8. Stringy score: " + scag.stringyScore;

//Monotonic

msg += "<br/>9. Monotonic score: " + scag.monotonicScore;

AE43AF



itemprop="mainContentOfPage"

[https://sope.prod.reuters.tv/program/rcom/v1/scroll?edition=us&pageid=USKCN1MQ1LN](https://sope.prod.reuters.tv/program/rcom/v1/scroll?edition=us&pageid=USKCN1MP2I2)

world news: **USKCN1M**P0Y6

**USKCN1M**Q2PM

<https://sope.prod.reuters.tv/program/rcom/v1/related-videos?pageid=USKCN1MP2I2>

[https://ieeexplore.ieee.org/abstract/document/6741396 2014](https://ieeexplore.ieee.org/abstract/document/6741396%202014)

// R stemDocument

stemDocument

install.package(‘tm’)

<https://spin.js.org/#?lines=10&length=38&width=52&radius=13&scale=0.25&corners=1&speed=1&rotate=62&animation=spinner-line-fade-quick&direction=1&color=%23ff8040&fadeColor=transparent&top=48&left=50&shadow=0%200%202px%20transparent>

log collector Netflix

<https://medium.com/netflix-techblog/scalable-logging-and-tracking-882bde0ddca2>

hive

<https://www.tutorialspoint.com/hive/hive_installation.htm>

<http://localhost:3000/#/page?userid=1>

$location.search().userid

Build with rollup with config

rollup -c

backup code

**var** datain =Dataset.**data**.map(**function** (d){  
 **var** dd = fieldDefs.map(**function**(f){**return** d[f.**field**] });  
 dd.**data** = dd;  
 **return** dd;});  
**var** bin = binnerN()  
 .startBinGridSize(40)  
 .isNormalized(**false**)  
 .minNumOfBins(1)  
 .maxNumOfBins(datain.**length**)  
 .data([]).updateRadius(**true**).binType(**"evenbin"**);  
bin.data(datain)  
 .**calculate**();  
color.domain(d3.*extent*(bin.bins.map(**function**(b) {**return** b.**length**})));  
**var** opacitys = d3.scale.*linear*().domain(color.domain()).range([0.3,1]);  
scatterData[0].**marker**.**opacity**=[];  
*//scatterData[0].type = "surface";*bin.bins.forEach(**function**(d) {  
 **var** point = bin.normalizedFun.scaleBackPoint(d.**val**);  
 *//var matrizz = scatterData[0].x.map(function(e){return 0;});  
 //scatterData[0].x.push(point[0]);  
 //scatterData[0].y.push(point[1]);  
 //scatterData[0].z.push(point[2]);  
 // matrizz.push(point[2]);  
 // scatterData[0].z.push(matrizz);* scatterData[0].**marker**.**size**.push(scaleXs(bin.**binRadius**/2));  
 scatterData[0].**marker**.**color**.push(color(d.**length**));  
 scatterData[0].**marker**.**opacity**.push(opacitys(d.**length**));  
 **var** text = fieldset[0] + **": "** + point[0] + **"<br>"**;  
 text += fieldset[1] + **": "** + point[1] + **"<br>"**;  
 text += fieldset[2] + **": "** + point[2];  
 scatterData[0].**text**.push(text);  
});  
scatterData.push({  
 **alphahull**: 2,  
 **color**:maincolor(0.3),  
 **opacity**: 0.05,  
 **type**: **'mesh3d'**,  
 **x**: [],  
 **y**: [],  
 **z**: []  
});  
datain.forEach(**function**(d){  
 scatterData[1].**x**.push(d[0]);  
 scatterData[1].**y**.push(d[1]);  
 scatterData[1].**z**.push(d[2]);  
});  
scatterData.push({  
 **alphahull**: 3,  
 **color**:maincolor(0.4),  
 **opacity**: 0.1,  
 **type**: **'mesh3d'**,  
 **x**: scatterData[1].**x**,  
 **y**: scatterData[1].**y**,  
 **z**: scatterData[1].**z**});  
scatterData.push({  
 **alphahull**: 4,  
 **color**:maincolor(0.2),  
 **opacity**: 0.15,  
 **type**: **'mesh3d'**,  
 **x**: scatterData[1].**x**,  
 **y**: scatterData[1].**y**,  
 **z**: scatterData[1].**z**});  
scatterData.push({  
 **alphahull**: 5,  
 **color**:maincolor(0.5),  
 **opacity**: 0.2,  
 **type**: **'mesh3d'**,  
 **x**: scatterData[1].**x**,  
 **y**: scatterData[1].**y**,  
 **z**: scatterData[1].**z**});  
scatterData.push({  
 **alphahull**: 6,  
 **color**:maincolor(0.6),  
 **opacity**: 0.25,  
 **type**: **'mesh3d'**,  
 **x**: scatterData[1].**x**,  
 **y**: scatterData[1].**y**,  
 **z**: scatterData[1].**z**});  
scatterData.push({  
 **alphahull**: 7,  
 **color**:maincolor(0.7),  
 **opacity**: 0.3,  
 **type**: **'mesh3d'**,  
 **x**: scatterData[1].**x**,  
 **y**: scatterData[1].**y**,  
 **z**: scatterData[1].**z**});  
  
*// bin.bins.forEach(function(d,i) {  
// for (var j =i+1;j<scatterData[0].x.length;j++)  
// scatterData[0].z[i].push(0);  
// })*

backup even bin

*// const box* **var** x = [0, 0, 1, 1, 0, 0, 1, 1];  
 **var** y = [0, 1, 1, 0, 0, 1, 1, 0];  
 **var** z = [0, 0, 0, 0, 1, 1, 1, 1];  
 **var** i = [7, 0, 0, 0, 4, 4, 2, 6, 4, 0, 3, 7];  
 **var** j = [3, 4, 1, 2, 5, 6, 5, 5, 0, 1, 2, 2];  
 **var** k = [0, 7, 2, 3, 6, 7, 1, 2, 5, 5, 7, 6];  
 **var** *boxscale* = **function** (x, y, z, position,radius) {  
 *// we will forego other checks for to limit the length of the example* **var** range\_x = [position[0]-radius,position[0]+radius];  
 **var** range\_y = [position[1]-radius,position[1]+radius];  
 **var** range\_z = [position[2]-radius,position[2]+radius];  
 x = x.map(**function**(e, i) {  
 **return** range\_x[e];  
 });  
  
 y = y.map(**function**(e, i) {  
 **return** range\_y[e];  
 });  
  
 z = z.map(**function**(e, i) {  
 **return** range\_z[e];  
 });  
  
 **return** {**x**: x, **y**: y, **z**: z};  
 }  
**var** datain =Dataset.**data**.map(**function** (d){  
 **var** dd = fieldDefs.map(**function**(f){**return** d[f.**field**] });  
 dd.**data** = dd;  
 **return** dd;});  
**var** bin = binnerN()  
 .startBinGridSize(40)  
 .isNormalized(**false**)  
 .minNumOfBins(1)  
 .maxNumOfBins(datain.**length**)  
 .data([]).updateRadius(**true**).binType(**"evenbin"**);  
bin.data(datain)  
 .**calculate**();  
color.domain(d3.*extent*(bin.bins.map(**function**(b) {**return** b.**length**})));  
**var** opacitys = d3.scale.*linear*().domain(color.domain()).range([0.3,1]);  
scatterData[0].**marker**.**opacity**=[];  
bin.bins.forEach(**function**(d) {  
 **var** point = bin.normalizedFun.scaleBackPoint(d.**val**);  
 scatterData[0].**x**.push(point[0]);  
 scatterData[0].**y**.push(point[1]);  
 scatterData[0].**z**.push(point[2]);  
 scatterData[0].**marker**.**size**.push(scaleXs(bin.**binRadius**/2));  
 scatterData[0].**marker**.**color**.push(color(d.**length**));  
 scatterData[0].**marker**.**opacity**.push(opacitys(d.**length**));  
 **var** text = fieldset[0] + **": "** + point[0] + **"<br>"**;  
 text += fieldset[1] + **": "** + point[1] + **"<br>"**;  
 text += fieldset[2] + **": "** + point[2];  
 scatterData[0].**text**.push(text);  
})