

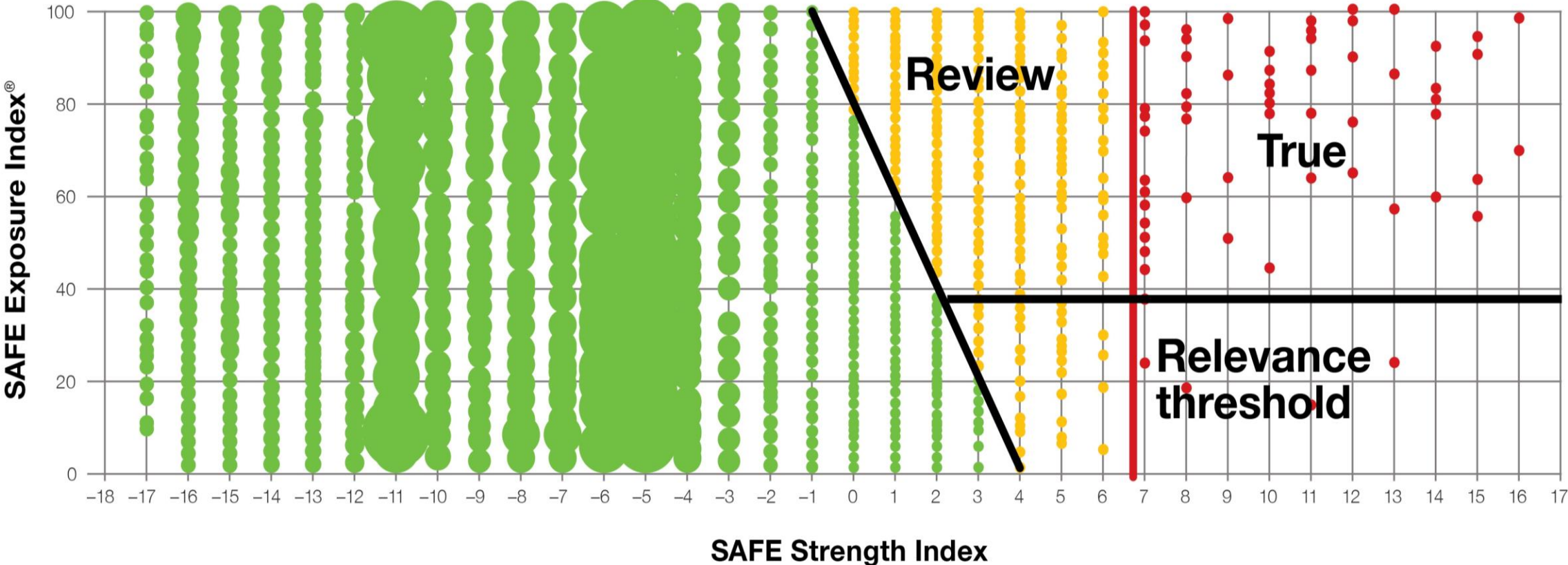
A day in the life of a functional data scientist

Richard Minerich, Director of R&D at Bayard Rock
@Rickasaurus

SAFE BANKING SYSTEMS PYRAMID OF RISK



Projecting onto a 2D Plane



The Pairwise Entity Resolution Process

Blocking

- Two Datasets (Customer Data and Sanctions)
- Pairs of Somehow Similar Records

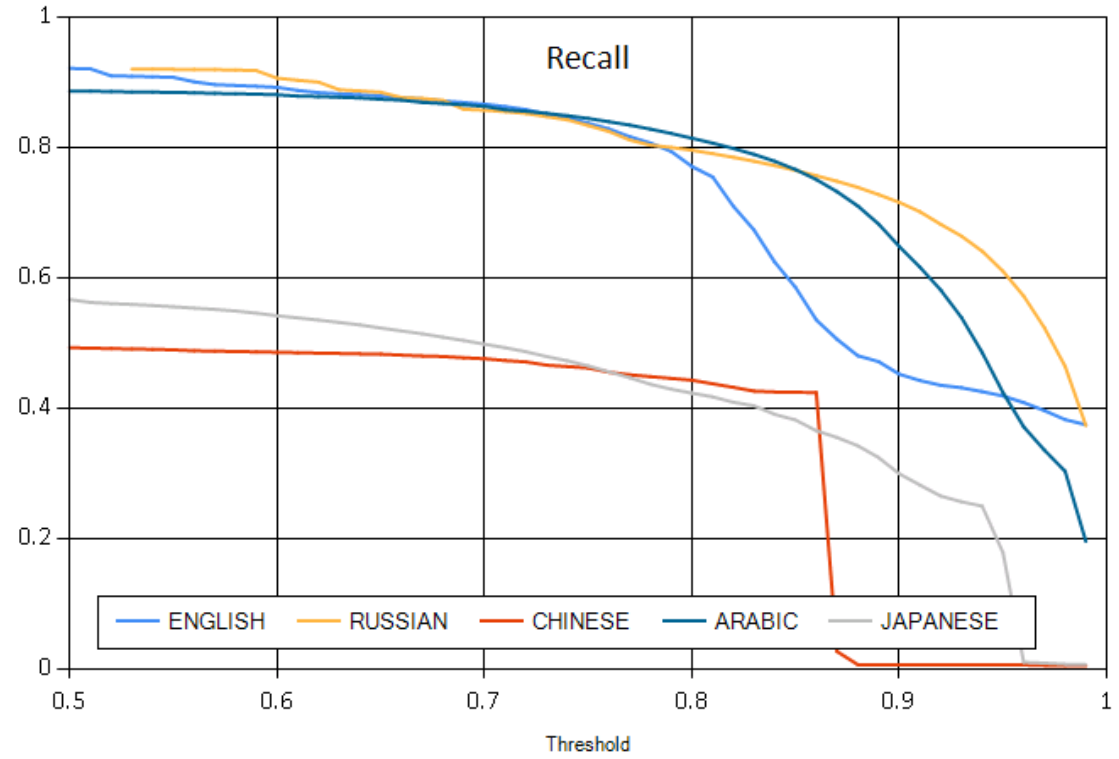
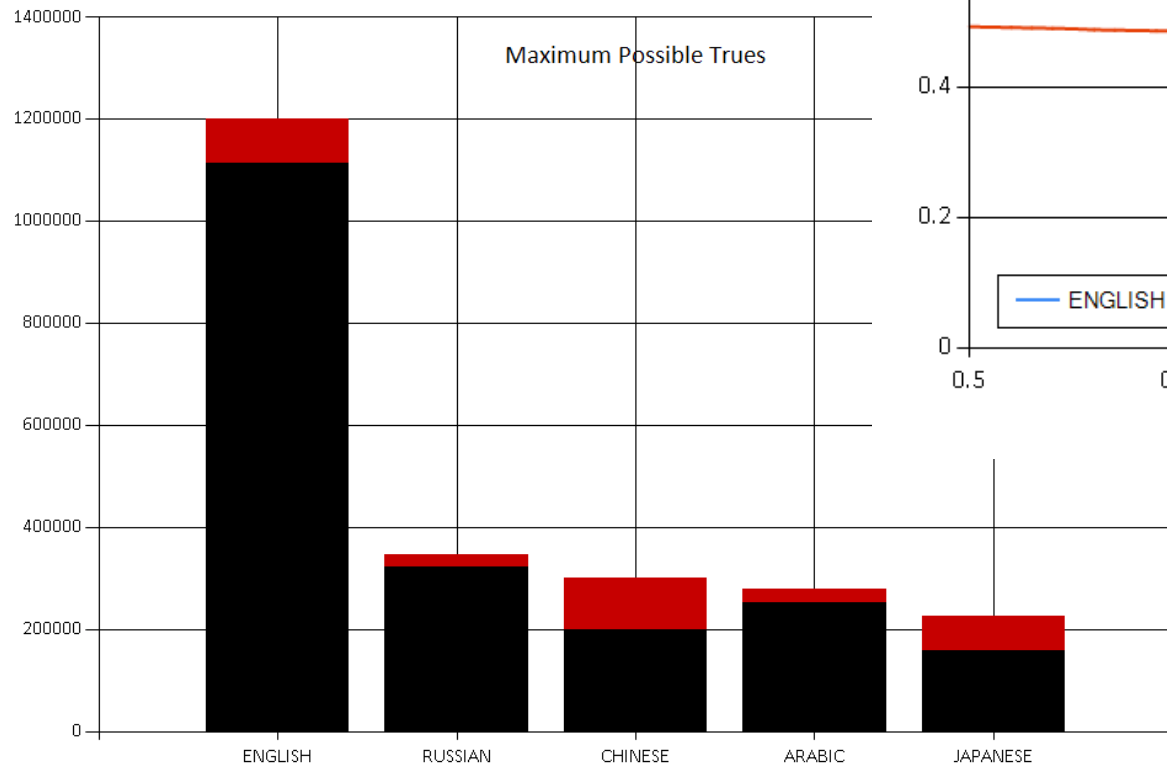
Scoring

- Pairs of Records
- Probability of Representing Same Entity

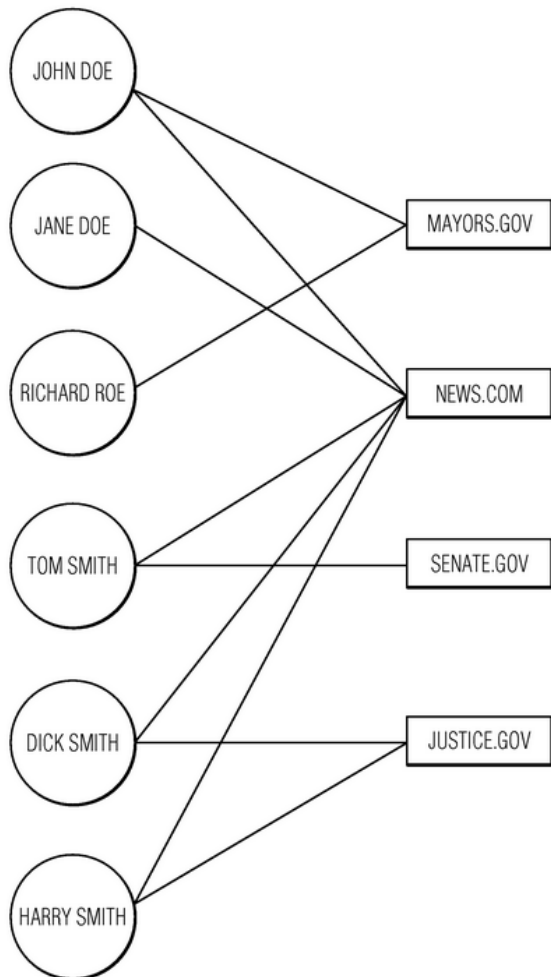
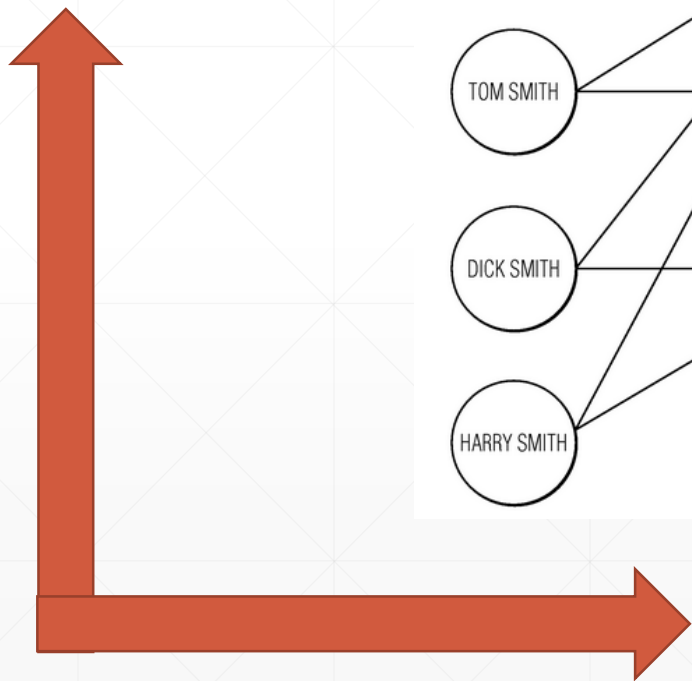
Review

- Records, Probability, Similarity Features
 - True/False Labels (Mostly by Hand)
-

Blocking

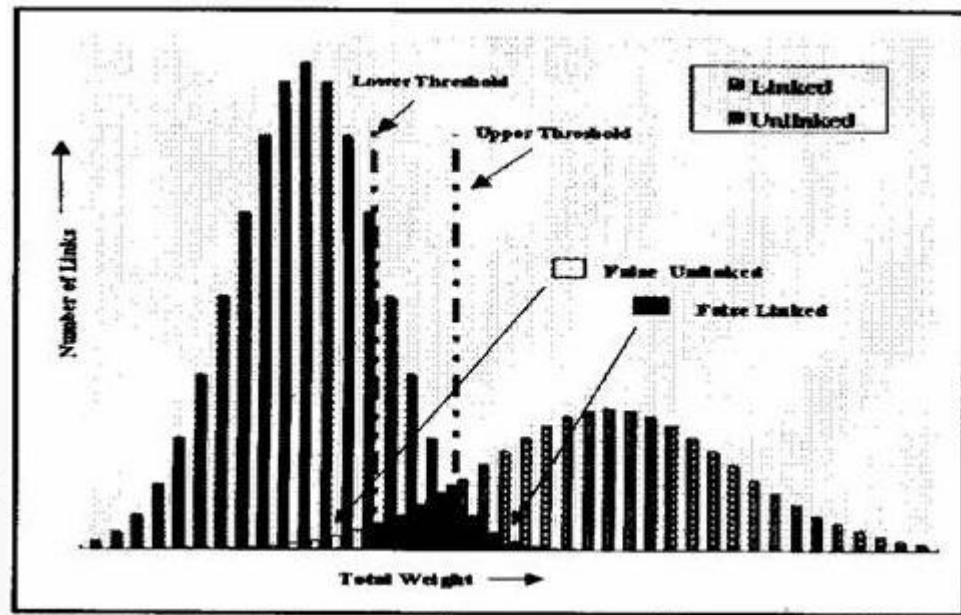


Likely to Launder Money

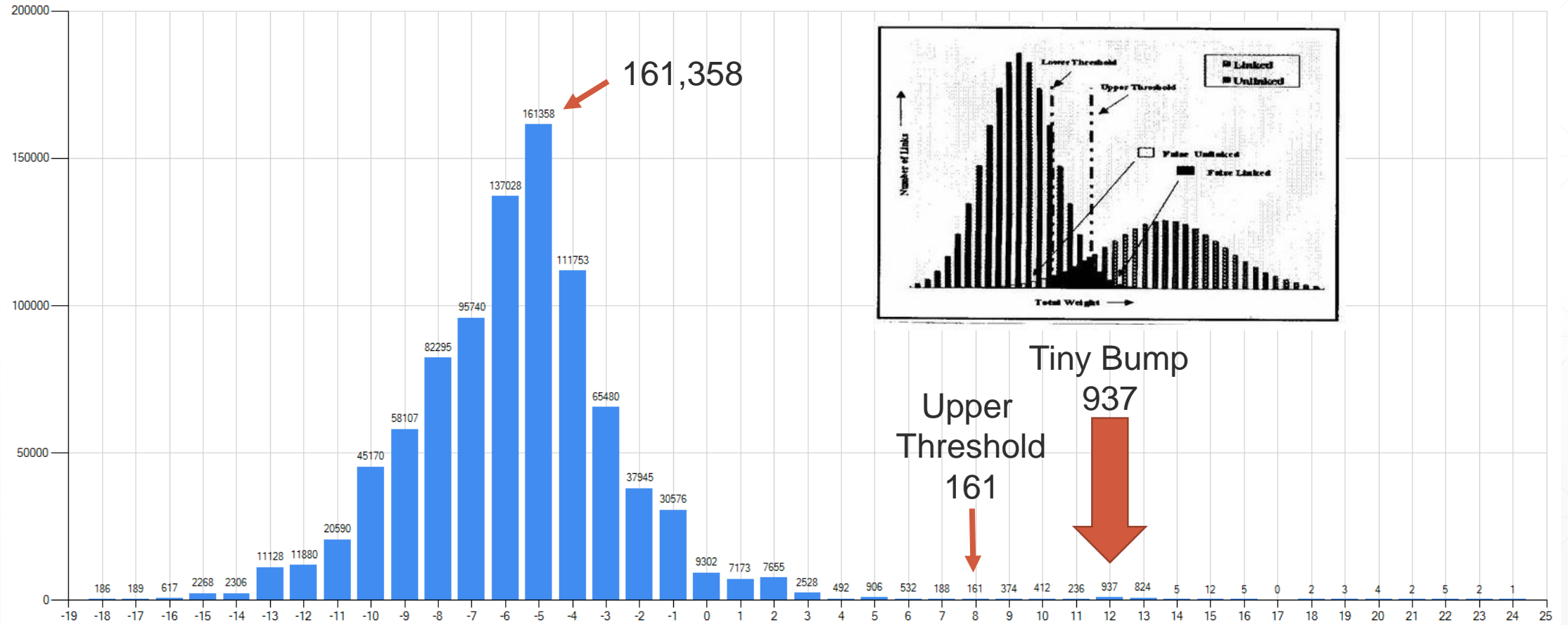


Probably the Same Person

Scoring: Risk vs Probability (The Ideal)

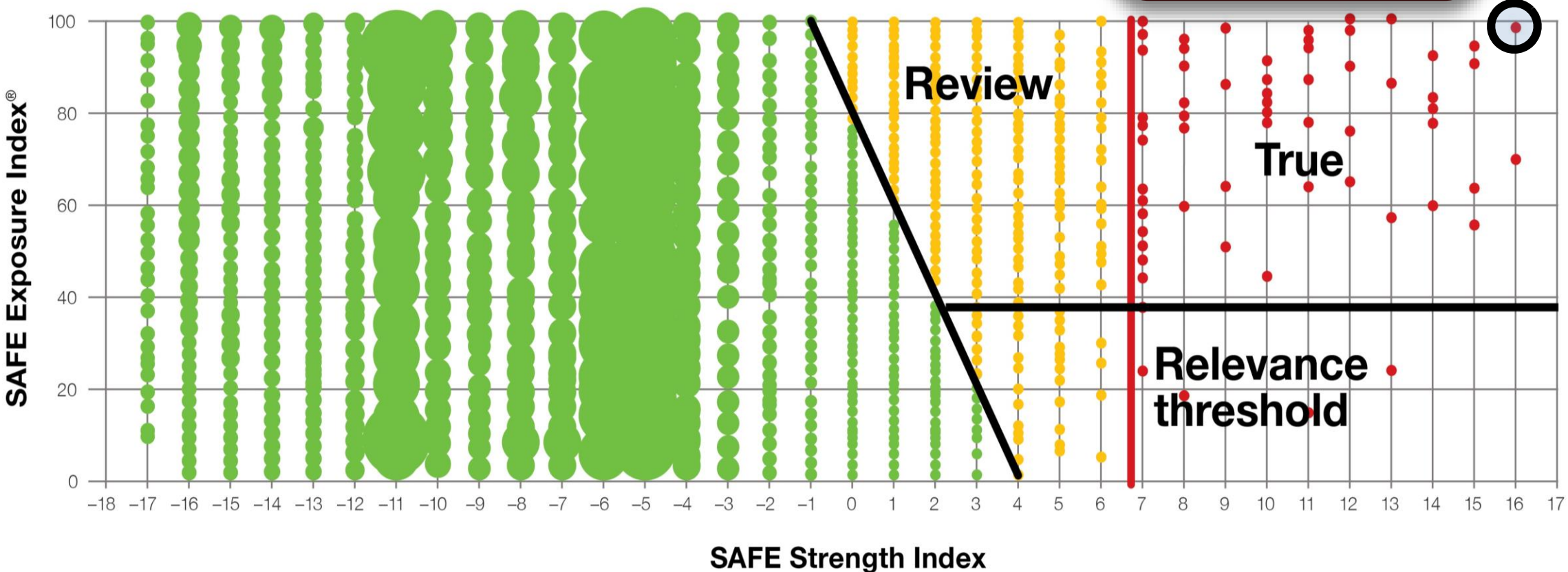


The Reality (Dominated by Garbage)



Let's dig into a single point

Jimmy Cournoyer
EI: 95/ SI:16



Aliases

- COURNOYER,COSMO
- COURNOYER,SUPERMAN

Alternative Spellings

Occupations

Position	Start Date	End Date
the Sinaloa Cartel and Hells Angels Motorcycle Club	2002	Oct 2011
Alleged kingpin of an international narcotics and money laundering enterprise		
comprised of the Montreal Mafia (Rizzuto Clan)		































Company(ies) reported in sources below: (3)

UID	Name	Country	EI
466172	SINALOA CARTEL	MEXICO	100
644456	HELLS ANGELS MOTORCYCLE CLUB	CANADA	100
1146414	RIZZUTO CLAN	CANADA	100

Citation Network (Safe View)

Information Source(s) (10)

to

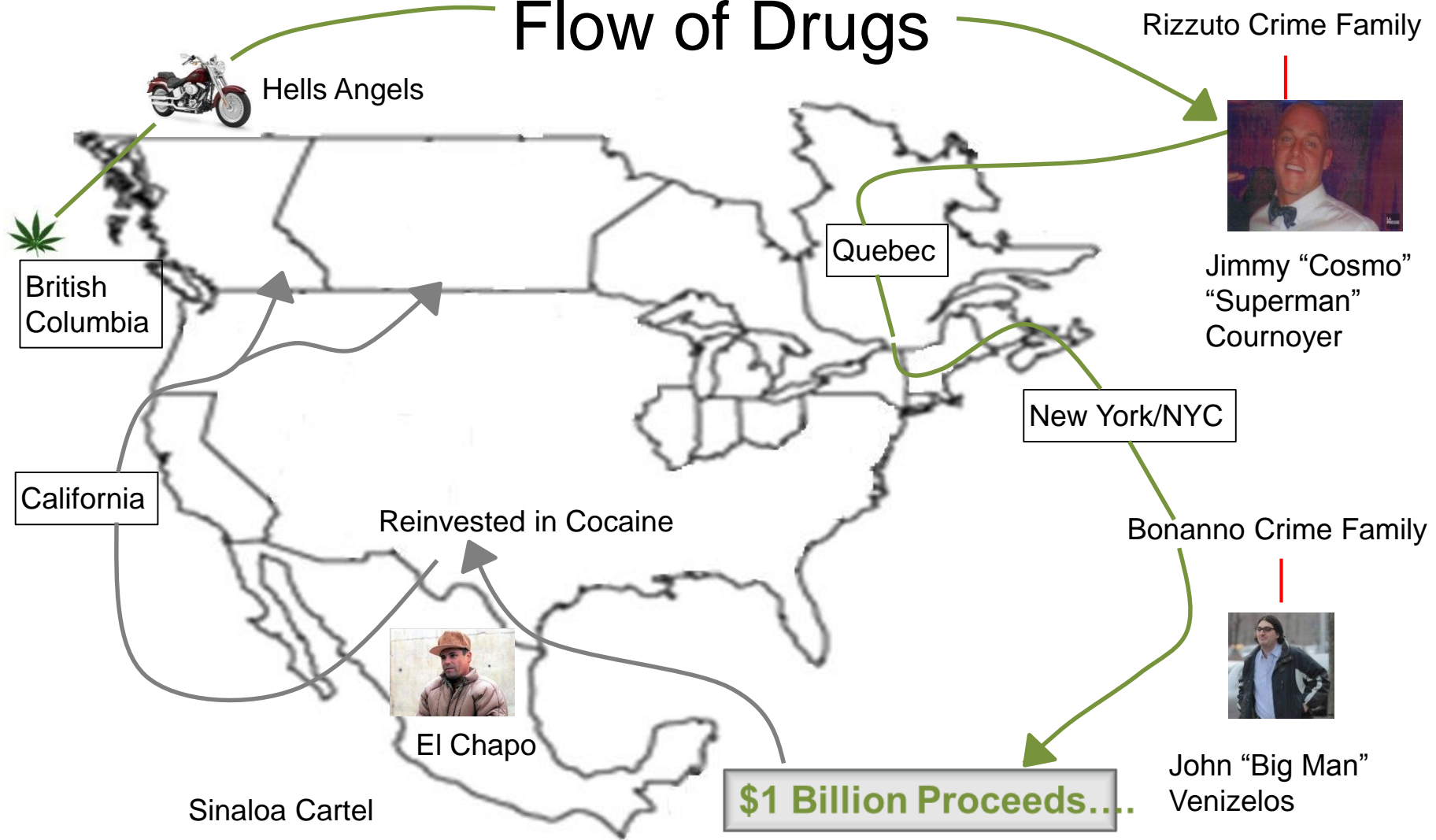
R	Date	P	C	F	I	A	Link
	2013/05/29					@	Public Access to US Court Electronic Records / USA
	2013/01/24					@	Federal Bureau of Prisons / USA
	2013/01/24					@	The Daily Mail News Online / United Kingdom
	2013/01/24					@	The Montreal Gazette News Online / Canada
	2013/01/24					@	The New York Post News Online / USA
	2012/03/21					@	Canoe 24 Hours Montreal News Online / Canada
	2012/03/21					@	Department of Justice / USA
	2012/03/21					@	The Cyberpress News Online / Canada
	2012/03/21					@	The Laval Courier News Online / Canada
	2012/03/21					@	The New York Post News Online / USA

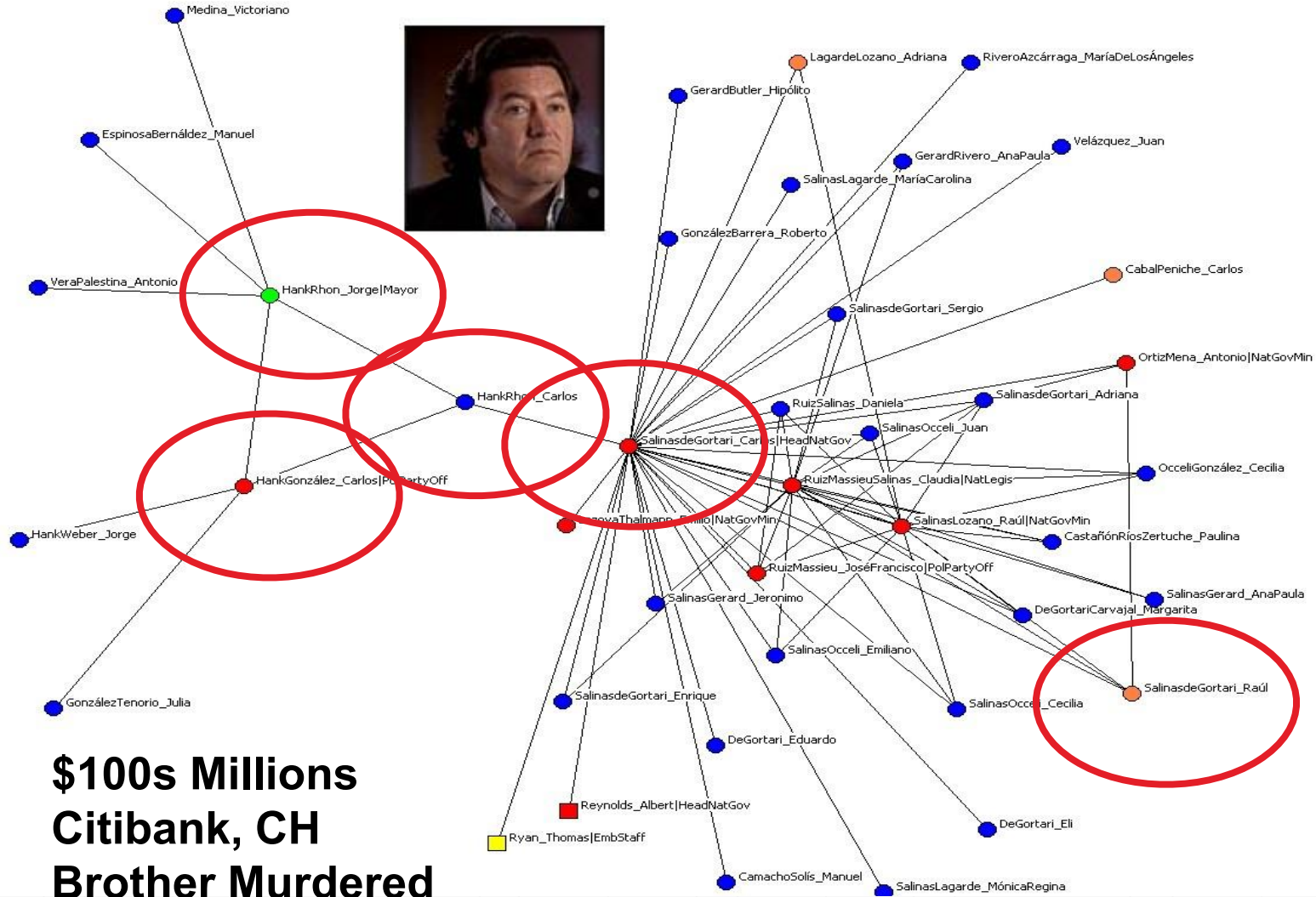
Relationship Network (Safe View)

Reported to be linked to: (12)

UID	Name	Country	Relationship	EI
18723	GUZMAN LOERA, Joaquin	MEXICO		100
240136	CHAJCHIC, German	USA		93
1281023	COURNOYER, Luc Normand	CANADA		83
1445785	CURATOLA, Dominick Vincent	USA		73
1445789	DECRESENZO, Michael	USA		71
1700695	RACINE, Mario Oliver	CANADA		95
1700698	PAISSE, Patrick	CANADA		93
1700709	CASTILLO MEDINA, Jose Alejandro	CANADA		77
1704437	GALEBI, Bobby	UNKNOWN		55
1939824	TALONI, Alessandro	USA		76
1939846	VENIZELOS, John	USA		91
2032556	TASCHETTI, John R	USA		55

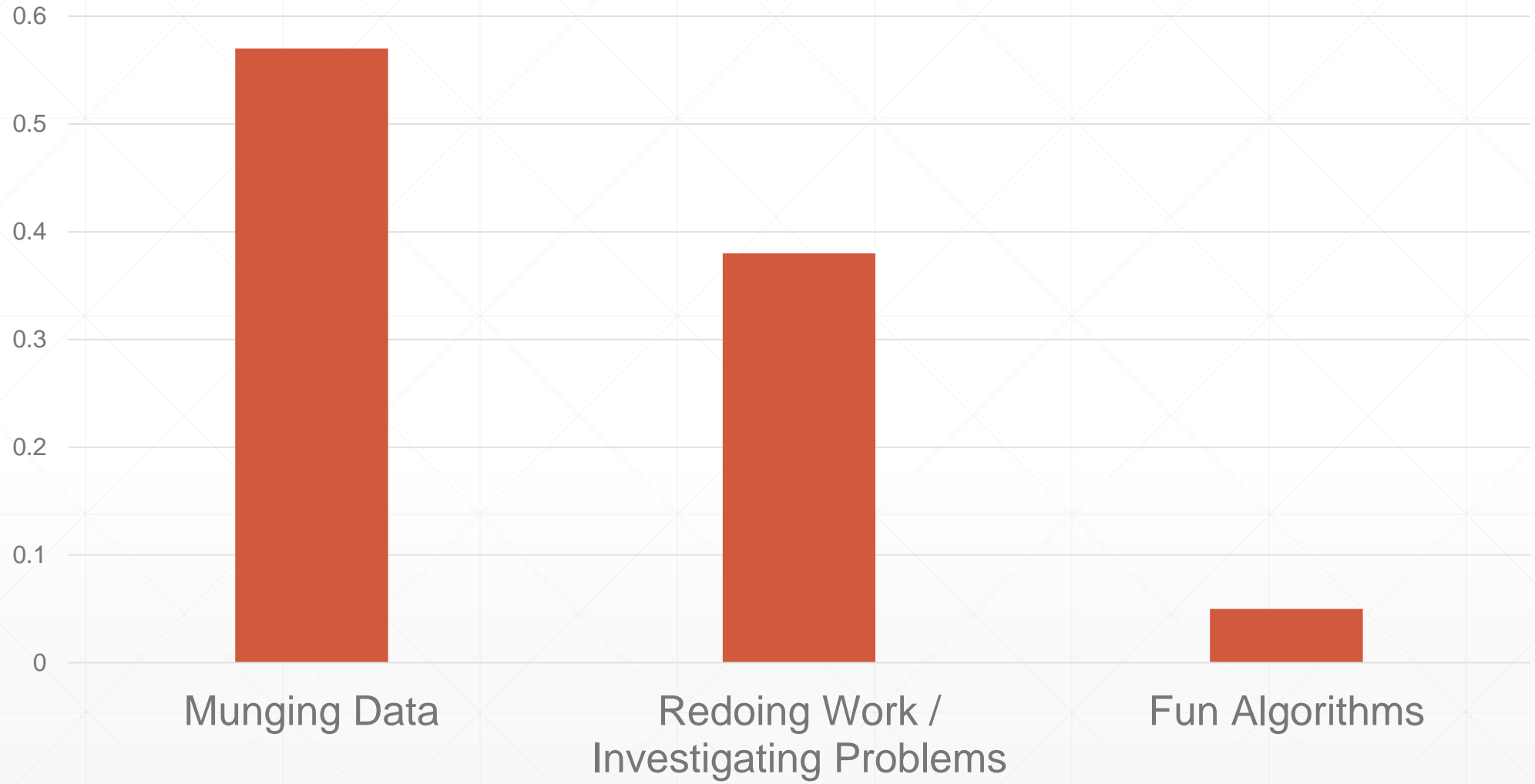
Flow of Drugs





**\$100s Millions
Citibank, CH
Brother Murdered**

% Time Spent



Disgustingly Bad but Fairly Large Datasets

- Both Wide (many fields) and Tall (many records)
- From different systems (different encodings)
- Missing data
- Poorly merged data
- Extra data
- Non-unique IDs

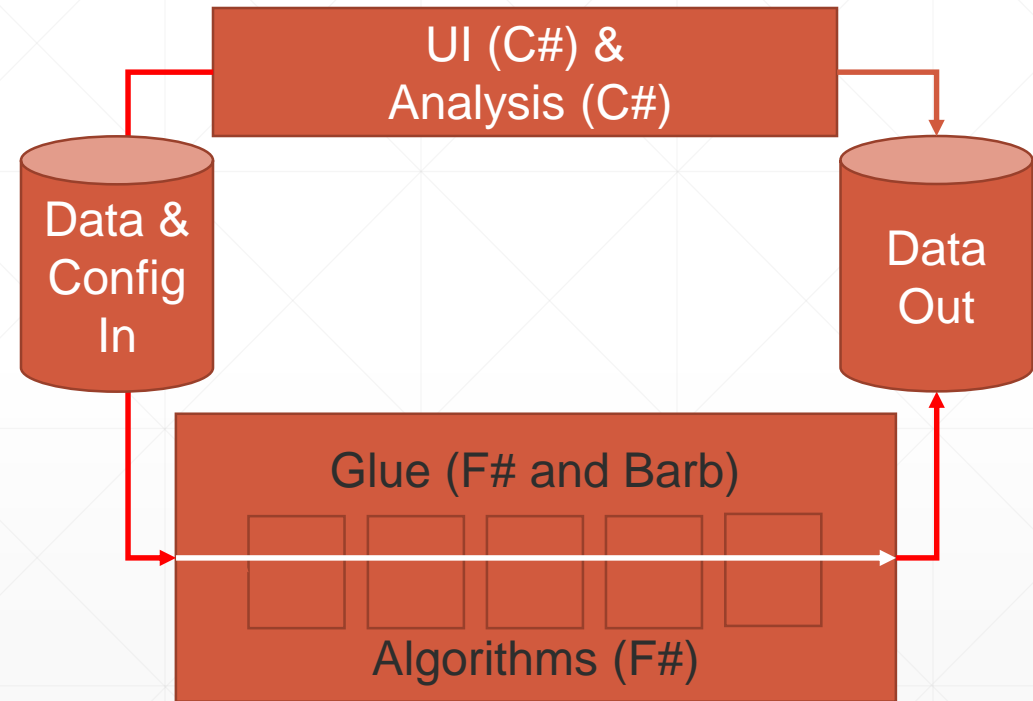
Every client is awful in a completely different way.

NAME	LARRY O BRIAN
STATE	CANADA
CITY	121 Buffalo Drive, Montreal, Quebec H3G 1Z2
ADDRESS	NULL
ZIP	00000
DOB	10/24/80; 1/1/1979

SAM – Building for Bad Data

- Lazy Pure Functional Core
- Programmable Data Cleaning
- Programmable ETL
- Ad-Hoc Behaviors

All with an F# Core and Barb for scripting.



Other Kinds of Problems (sometimes even my fault)

- Extra / Missing Data (e.g. incorrect subset or incorrect joins)
- Wrong version of data (e.g. bad sync in SQL)
- Bad configuration of dependencies

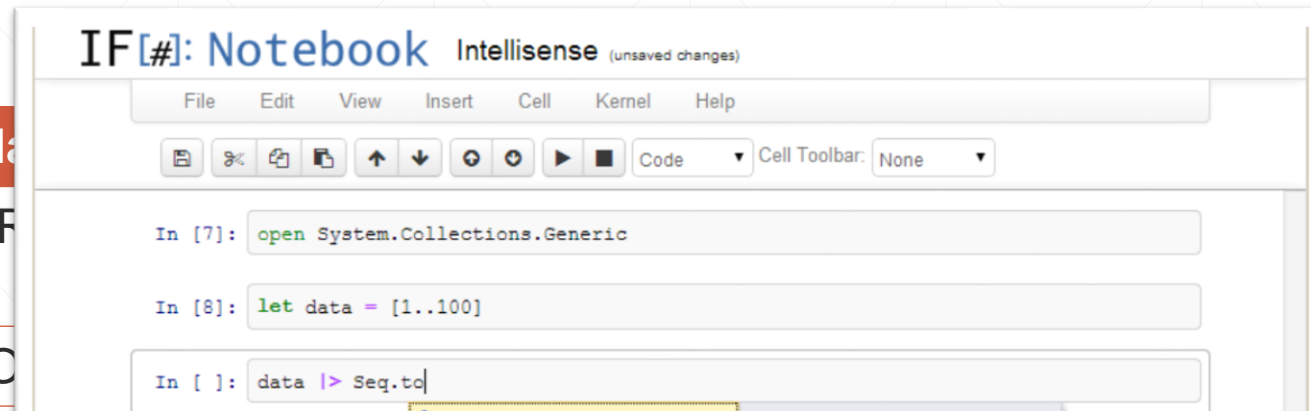
The data lives in a locked down environment and so feedback cycles are slow.

Lesson: Be Paranoid

F# Tools From Bayard Rock

<http://github.com/BayardRock>

Tokens	Class
Pegasus Airlines	ORGANIZATION
Istanbul	LOCATION
Sochi	LOCATION
Russia	LOCATION
Turkey	LOCATION
Transportation Ministry	ORGANIZATION



IF[#]: Notebook Intellisense (unsaved changes)

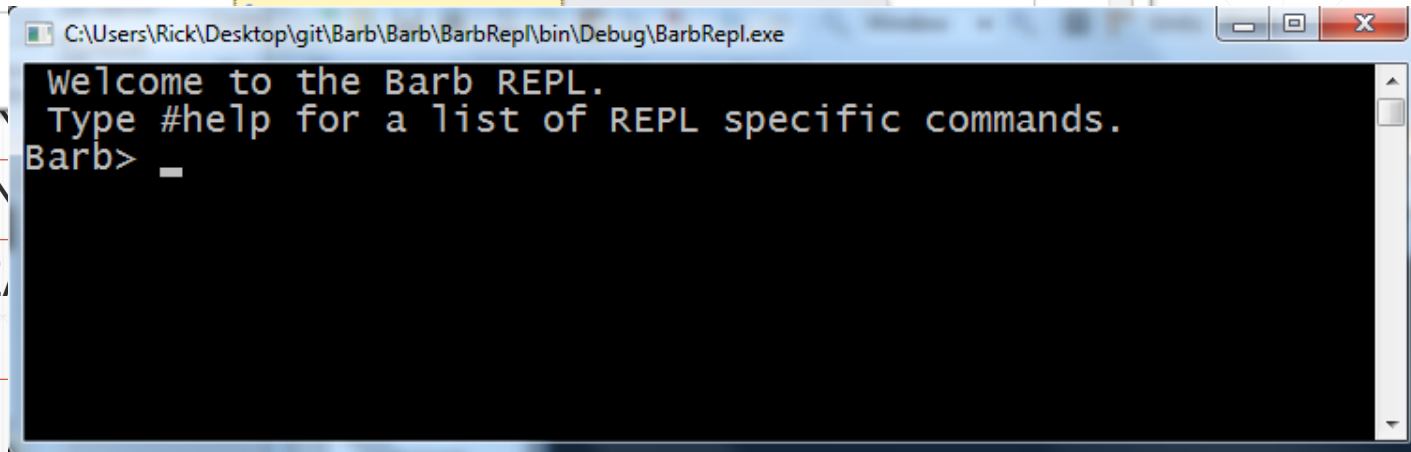
File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
In [7]: open System.Collections.Generic
```

```
In [8]: let data = [1..100]
```

```
In [ ]: data |> Seq.tc|
```



C:\Users\Rick\Desktop\git\Barb\Barb\BarbRepl\bin\Debug\BarbRepl.exe

```
Welcome to the Barb REPL.  
Type #help for a list of REPL specific commands.  
Barb> _
```

FSharpWebIntellisense

```
1 type Person = { FirstName: string; LastName: string }
2 let p = { FirstName = "Robert"; LastName = "Dole" }
3 p.
```

fx Equals
FirstName
fx GetHashCode
fx GetType
LastName
fx ToString

Person.FirstName: string



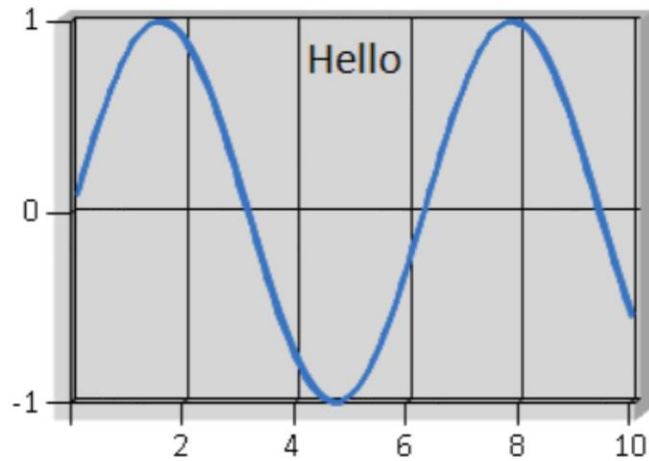
```
1 open System
2 open System.IO
3 open System.Text
4
5 derp
6
7 let hexdump byteCount (bytes: byte[]) =
8     let totalLength = Array.length bytes
9     let (|Char|Ctrl|) c = char bytes.[c] |> fun ch -> if Ch
10    let (|TooLong|_|) c = if c < totalLength then None else
11    let rec printLine pos =
12        if pos < totalLength then
13            let length = min byteCount (totalLength - pos)
14            let rec printByte =
```

<https://github.com/BayardRock/FSharpWebIntellisense>



iFSharp Notebook

```
In [12]: Chart.Line(data) |> Chart.WithTitle("Hello") |> Chart.wi
```



- fx* With3D
- fx* WithDataPointLabels
- fx* WithLegend
- fx* WithMarkers
- fx* WithStyling
- fx* WithTitle
- fx* WithXAxis
- fx* WithXAxis?

```
In [ ]: let x = errors + 1  
x.|
```

The value or constructor 'errors' is not defined



Barb, a simple .net record query language

Name.Contains "John" and (Age > 20 or Weight > 200)

The image shows two overlapping windows. The left window is a terminal titled 'C:\Users\Rick\Desktop\git\Barb\Barb\BarbRepl\bin\Debug\BarbRepl.exe' with the text: 'Welcome to the Barb REPL. Type #help for a list of REPL spec Barb>'. The right window is 'Safe Alert Manager 1.0.0.0' showing a table of records. The table has columns: ID, CustName, EI, SI, and Category. The first record is selected, showing its details in a 'Directly Accessible Record Fields' pane on the right.

ID	CustName	EI	SI	Category
15802	B HASHIM ABU BAKAR	97	9	Some(True)
	VOLOKHO YURY NIKOLAEVICH	96	8	Some(True)
	OULD MOHAMED EL MOCTAR	86	8	Some(True)
	MUBARAK GAMAL HOSNY	100	8	Some(True)
	HASHEM AHMED HASHEM	92	8	Some(True)
	DELOCHE YVES MICHAEL	89	8	Some(True)
	MAZHAR SALEEM KHAN	93	8	Some(True)
	AHMEDOU MOHAMED LEMINE OULD	90	7	Some(True)
	MIYAHARA TAKESHI	96	7	Some(True)
	OULD AHMEDOU MOHAMED LEMINE	97	7	Some(True)
	RAZA ABID KHAN	96	7	Some(True)

Directly Accessible Record Fields

Field	Contents
CustName	B HASHIM ABU BAKAR
ProvName	BIN HASHIM ABU BAKA
SI	9
EI	97
NFQ	6
MNI	3
DOB	0
LOC	3
QAL	2
IDM	0
BP	-5
ADD	0

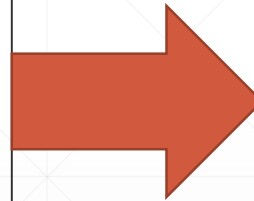
Scanning Alerts from: C:\Users\Rick\Desktop\SBS\Alert Files\FAA_RELFORCB_TOP750K.xml
With Settings: C:\TFS\BayardRock\Project Greif\Safe Alert Suite 2.6\Binaries\Default\alertsettings.json
No reasonable customer date of birth format found.

<https://github.com/Rickasaurus/Barb>



MITIE Dot Net (a wrapper for MIT's MITIE)

A **Pegasus Airlines** plane landed at an **Istanbul** airport Friday after a passenger "said that there was a bomb on board" and wanted the plane to land in **Sochi, Russia**, the site of the Winter Olympics, said officials with **Turkey's Transportation Ministry**.



Tokens	Classification
Pegasus Airlines	ORGANIZATION
Istanbul	LOCATION
Sochi	LOCATION
Russia	LOCATION
Turkey	LOCATION
Transportation Ministry	ORGANIZATION

<https://github.com/BayardRock/MITIE-Dot-Net>

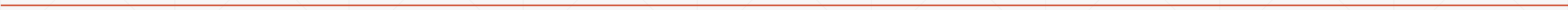
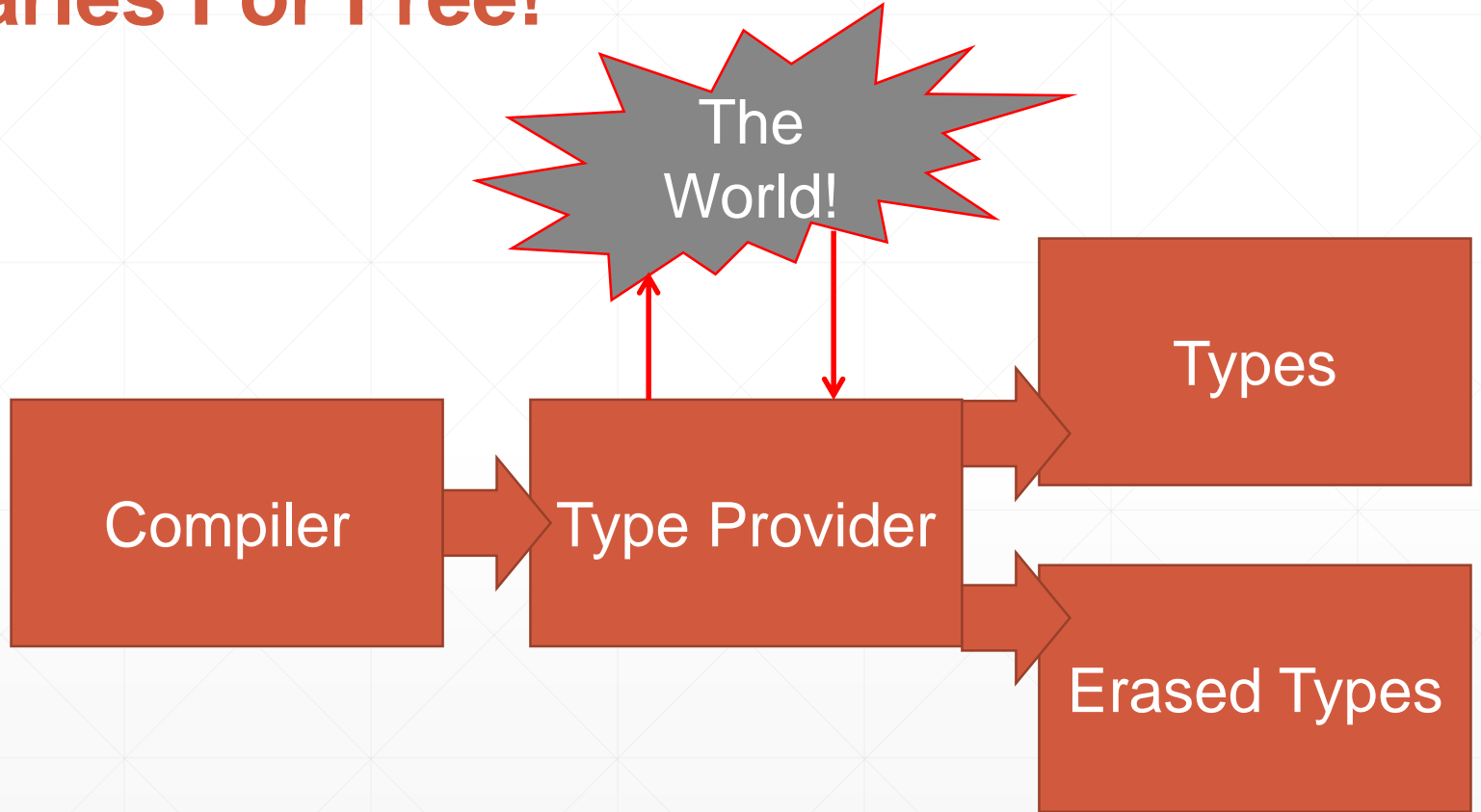
Other F# Community Tools (Not by Us)

- Data Type Providers (SQL, OData, CSV, etc..)
 - Language Type Providers (R, Matlab, Python soon)
 - Deedle (like Pandas but for F#)
 - F# Charting
-

The Magic of Type Providers

```
type Netflix = ODataService<"http://odata.netflix.com">  
  
let avatarTitles =  
    query { for t in netflix.Titles do  
            where (t.Name.Contains "Avatar")  
            sortBy t.Name  
            take 100 }
```

Type Providers! Libraries For Free!





Deedle (Like Python's pandas but for F#)

- Designed with Data Type Providers in Mind
- Interops with the R Type Provider





But what about algorithmic code?

Ranking vs Regression

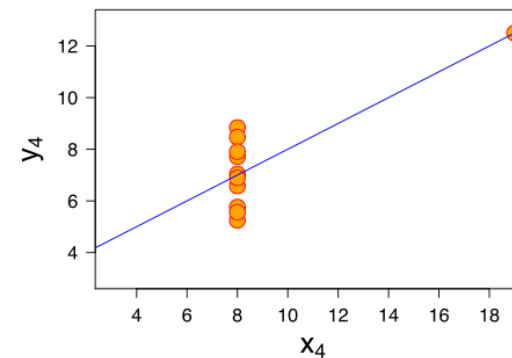
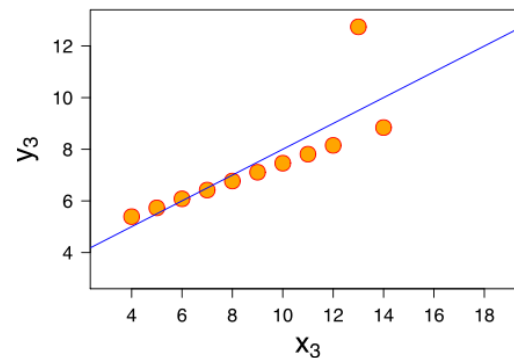
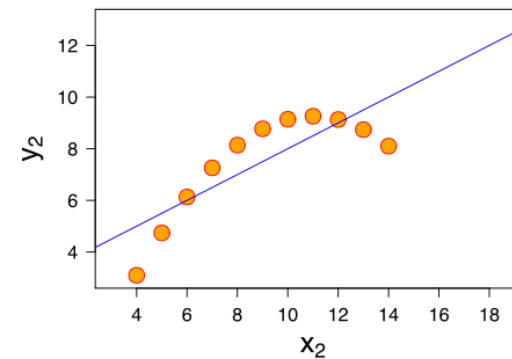
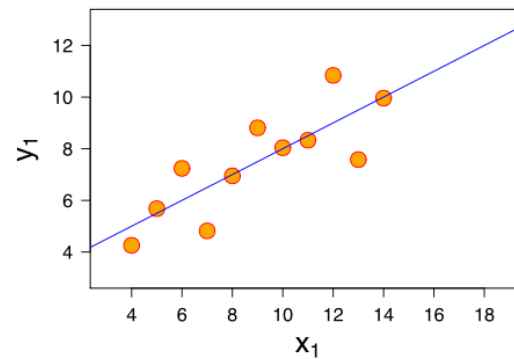
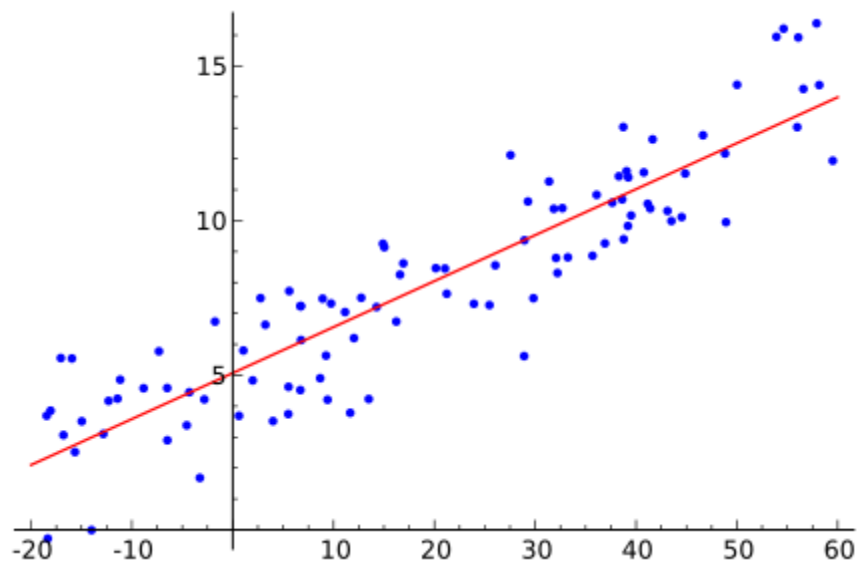
- Regression - you're trying to guess a number, only distance matters
 - May do a very bad job at ordering
- In Ranking you're trying to figure out some order, only order matters
 - May do a very bad job at providing a meaningful number

Example: You're a doctor with 20 spots open and 100 patients who want to see you today, which method would be the best for selecting 20?

Regression

$$y = X\beta + \varepsilon$$

y is labels
 X is features
 β is weights
 ε is errors



“OLS” Regression via Gradient Descent in F#

```
let gdStep (X: Matrix<float>) (y: Vector<float>) (θ: Vector<float>) (α: float) =  
    let nsamples = float y.Count in θ - ((X * θ - y) * X * (α / nsamples))  
  
let thetas X y alpha theta =  
    let rec calcNext theta =  
        seq {  
            let theta' = gdStep X y theta alpha  
            yield theta'  
            yield! calcNext theta'  
        }  
    calcNext theta  
  
let newThetas X y alpha = thetas X y alpha (DenseVector.zeroCreate(X.ColumnCount))
```

Simple Ranking? You Can Use Regression.

- The features are the difference in would-be regression features
- The value to predict is the difference in rank

Select 2 labeled samples randomly => (x_1, y_1) (x_2, y_2)

$$x = x_1 - x_2$$
$$y = y_1 - y_2$$

	Sample 1	Sample 2	Result
Names?	1	1	0
Addresses?	1	0	1
DOB?	0	1	-1
Same Person?	0	0	0

Simple Ranking in F#

```
let makeRankingSamples (X: Matrix<float>) (y: Vector<float>) (nsamples: uint32) =  
    let rnd = new System.Random()  
    let inline rndIdx () = rnd.Next(0, X.RowCount - 1)  
    [| for i = 0u to nsamples - 1u do  
        let idx1, idx2 = rndIdx (), rndIdx ()  
        let x = X.Row(idx1) - X.Row(idx2) // The difference in features  
        let y = y.[idx1] - y.[idx2]      // The difference in rank  
        yield x, y  
    |]  
  
let XSamples, ySamples =  
    let samples = makeRankingSamples X y 1000u  
    samples |> Seq.map fst |> Seq.toList |> DenseMatrix.ofRowVectors,  
    samples |> Seq.map snd |> DenseVector.ofSeq
```

Combined Ranking and Regression – D. Sculley

You can improve your regression with ranking, and your ranking with regression.

The best of both worlds!

METHOD	AdSet1	
	AUC Loss	MSE
REGRESSION	0.133	0.084
RANKING	0.132	0.094
CRR	0.132	0.084

Table 3: Click-Prediction Results. The CRR method gives “best of both” performance on click-prediction data, giving MSE equivalent to the regression-only method with 0.8% less AUC Loss.

Combined Regression and Ranking

D. Sculley
Google, Inc.
Pittsburgh, PA USA
dsculley@google.com

ABSTRACT

Many real-world data mining tasks require the achievement of two distinct goals when applied to unseen data: first, to induce an accurate preference *ranking*, and second to give good regression performance. In this paper, we give an efficient and effective Combined Regression and Ranking method (CRR) that optimizes regression and ranking objectives simultaneously. We demonstrate the effectiveness of CRR for both families of metrics on a range of large-scale tasks, including click prediction for online advertisements. Results show that CRR often achieves performance equivalent to the best of both ranking-only and regression-only approaches. In the case of rare events or skewed distributions, we also find that this combination can actually improve regression performance due to the addition of informative ranking constraints.

Categories and Subject Descriptors

H.3.3 [Information Systems]: Information Storage and Retrieval—*Information Search and Retrieval*

General Terms

Algorithms, Measurement, Performance

Keywords

ranking, regression, large-scale data

1. INTRODUCTION

This paper addresses the real-world scenario in which we require a model that performs well on two distinct families of metrics. The first set of metrics are regression based metrics, such as Mean Squared Error, which reward a model for predicting a numerical value y' that is near to the true target value y for a given example, and penalize predictions far from y . The second set are rank-based metrics, such as Area under the ROC curve (AUC), which reward a model

for producing predicted values with the same pairwise ordering $y'_1 > y'_2$ as the true values $y_1 > y_2$ for a pair of given examples.

In many settings good performance on both families together is needed. An important example of such a setting is the prediction of clicks for sponsored search advertising. In real-time auctions for online advertisement placement, ads are ranked based on $bid + pCTR$ where $pCTR$ is the predicted click-through rate (CTR) of an ad. Predicting a good ranking is critical to efficient placement of ads. However, it is also important that the $pCTR$ not only give good ranking value, but also give good regression estimates. This is because online advertisements are priced using next-price auctions, in which the price for a click on an ad at rank i is based on the expected $bid + pCTR$ for the ad at the next lower rank [1]. In such a setting, it is critical that the actual $pCTR$ estimate be as accurate as possible to enable fair and efficient pricing. Thus, CTR prediction must be performed with both ranking and regression performance in mind.

1.1 Ranking vs. Regression

At first, it may appear that simply learning a good regression model is sufficient for this task, because a model that gives perfect regression will, by definition, also give perfect ranking. However, a model with near-perfect regression performance may yield arbitrarily poor ranking performance. Consider the case of an extreme minority-class distribution, where nearly all examples have target value $y = 0$ and only a small fraction have value $y = 1$. Here, it is possible to achieve near-perfect regression performance by always predicting 0; this gives a useless ranking. Even in less extreme cases, small regression errors can cause large ranking errors.

Similarly, it is easy to see that even a perfect ranking model may give arbitrarily poor regression estimates. Scores produced by a perfect ranking model may include arbitrary order-preserving transformations of the true target values, resulting in predictions that are useless as regression values.

1.2 Benefits of Combination

In this paper, we propose to address these issues using a combined objective function that optimizes regression-based and rank-based objectives simultaneously. This combination guards against learning degenerate models that perform well on one set of metrics but poorly on another. In our experiments, we find that the combined approach often gives “best of both” performance, performing as well at regression as a regression-only method, and as well at ranking as a ranking-only method.

Additionally, we find that the combined approach can ac-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'10, July 25–28, 2010, Washington, DC, USA
Copyright 2010 ACM 978-1-4503-0055-1/10/07 ...\$10.00.

Combined Ranking and Regression – D. Sculley @ Google, Inc

```
let makeSamples (X: Matrix<float>) (y: Vector<float>) ( $\alpha$ : float) (nsamples: uint32) =  
    let rnd = new System.Random()  
    let inline rndZ () = rnd.NextDouble()  
    let inline rndIdx () = rnd.Next(0, X.RowCount - 1)  
    [| for i = 0u to nsamples - 1u do  
        let z = rndZ ()  
        yield  
            if z <  $\alpha$  then // Regression  
                let idx = rndIdx () in  
                X.Row(idx), y.[idx]  
            else // Ranking  
                let idx1, idx2 = rndIdx (), rndIdx ()  
                let x = X.Row(idx1) - X.Row(idx2)  
                let y = y.[idx1] - y.[idx2]  
                x, y  
    |]
```

Thank You!

Check out the NYC F# User Group:
<http://www.meetup.com/nyc-fsharp>

Find out more about F#:
<http://fsharp.org>

Contact me on twitter:
[@Rickasaurus](https://twitter.com/Rickasaurus)

