Automated Quadratic Characterization of Flow Cytometer Instrument Sensitivity * (flowQB Package: Advanced Processing Using Data NIH3)

Faysal El Khettabi
Terry Fox Laboratory
British Columbia Cancer Agency
Vancouver, BC, Canada
fkhettabi@bccrc.ca
faysal.el.khettabi@gmail.com

April 12, 2014

1 Licensing

2 Introduction

We developed an automated approach to determine a cytometer's detection efficiency (Q) and background illumination (B) independent of a pre-defined bead set, based on Kmeans clustering and quadratic regression methods. We used a quadratic formulation for modelling Q and B that offers more physical insight about cytometry sensitivity than its truncated linear version [1, 3, 2], by considering the term $CV_{intrinsic}$ as part of the problem to solve. Using Kmeans in place of manual gating enabled an automated analysis to calculate Q, B and $CV_{intrinsic}$ quantities objectively and in a time-efficient manner. We validated our approach on flow cytometry sensitivity datasets through comparison to Q, B and $CV_{intrinsic}$ values obtained by manual analysis. The fully automated results ensure adequate analysis for the flow cytometry sensitivity datasets. Our approach is implemented through the R/Bioconductor package *flowQB*.

*This project was supported by the Terry Fox Foundation and the Terry Fox Research Institute and by grant 700374 from the Canadian Cancer Society, Toronto.

- **Methods:** We propose a collection of R generic functions to calculate Q, B and $CV_{intrinsic}$ quantities objectively and in a time-efficient manner. We have implemented these functions in the Bioconductor package flowQB. We illustrate their use in this draft.
- **Results** We hope that these proposed R generic functions will become the base for the development of many tools to calculate Q, B and $CV_{intrinsic}$.
- **keywords** Flow Cytometry, High Throughput, Doublet, Instrument Sensitivity, Kmeans, Mean Fluorescence Intensity (MFI), Molecules of Equivalent Soluble Fluorochrome (MESF), linear and quadratic regressions, Q (detector efficiency) , B (background light level).

Auto-Gating of Bead Populations

First read the data which is in a specific folder. Our data is in flowQB-extdata folder:

```
> rm(list=ls(all=TRUE))
> library("flowQB")
> File= system.file("extdata","NIH3.fcs",package="flowQB")
> library(flowCore)
> Data=read.FCS(File)
```

We used the logicle transformation to transform the data as previously described in [4]. Automatically, it estimates the logicle transformation parameters using Data and the Markers:

```
> Markers=c("FSC-A","SSC-A","B710-A","B515-A","G780-A","G710-A","G660-A","G610-A"
+ ,"G560-A", "R780-A","R710-A","R660-A","V800-A","V705-A","V655-A"
+ ,"V605-A","V585-A","V565-A","V545-A","V450-A")
> lgcl <- estimateLogicle(Data, channels = Markers)
> TransformedData <- transform(Data, lgcl)
```

Get the logicle Transformed MFI for each event:

```
> TransformedMFI=data.frame(exprs(TransformedData))
```

Get the original MFI for each event

```
> OriginalMFI=data.frame(exprs(Data))
```

To identify bead singlets, we used the approach of gating on FSC & SSC (see [3, 2] for more information on doublet discrimination effects on Q and B calculations). The parameters $(a1, b1)$ gate on FSC and $(a2, b2)$ gate on SSC should be defined by the user. The side scattering index is 3 and the forward scattering index is 1. We set the $(a1, b1)$ gate on FSC and $(a2, b2)$ gate on SSC as:

```
> a1=3.3
> b1=3.6
> a2=2.5
> b2=2.85
> SingletsIndices=which(TransformedMFI[,1]> a1 & TransformedMFI[,1] < b1
+  & TransformedMFI[,3]> a2 & TransformedMFI[,3] < b2)
```

Distribution Resolution Metric

The set of markers that infer the existence of the actual bead populations,

```
> nClusters=7
```

in the data are determined by using the distribution resolution metric [1, 5], $R_d = \frac{\mu_1 - \mu_2}{\sigma_1 + \sigma_2}$, which is the difference in the means μ_1 and μ_2, of two populations divided by the sum of the standard deviations of the distributions, σ_1 and σ_2. The user can select only a set of markers for assessment, in this example, we selected all markers in Data and we get their MFI for each considered singlet event. To facilitate the identification of the bead sub-populations, we will perform 2D Kmeans auto-gating on the given marker and side scattering (SSC). The index 3 is associated to the SSC in the original Data.

```
> MarkersForAssessment=c(3,4,5,6,7)
> TSSINGLETS=TransformedMFI[SingletsIndices,MarkersForAssessment]
> OSSINGLETS=data.frame(OriginalMFI[SingletsIndices,MarkersForAssessment])
>
```

The index 1 is now associated to the SSC in the new data TSSINGLETS and OSSINGLETS. Kmeans clustering method to define the actual bead populations using 2D gating, $(1, markerindex)$, for instance the marker B515 has now index 3 in the new data TSSINGLETS and OSSINGLETS. Gaussian distribution fitting is used for peak statistics extraction, the fitting parameters means and standard deviations are the estimations of MFIs and SDs values. The function "gPS" in flowQB has the Gaussian distribution fitting which uses R software "MASS" package [9].

```
> gps=gPS(TSSINGLETS[,c(1,3)],OSSINGLETS[,c(1,3)],7)
>
```

Estimations of MFIs and SDs values using Robust Statistics [6] is implemented in the function "rPS" in flowQB.

```
> rps=rPS(TSSINGLETS[,c(1,3)],OSSINGLETS[,c(1,3)],7)
>
```

So now we are ready to turn the extracted peak statistics into Table 1 for the marker B515, using

```
> library(xtable)
> PSdataB515=data.frame(NE=gps[,1]
+ ,gMeans=gps[,2]
+ ,rMeans=rps[,2]
+ ,gSDs=gps[,3]
+ ,rSDs=rps[,3]
+ ,gRDs=gps[,4]
+ ,rRDs=rps[,4])
> row.names(PSdataB515)=paste("Peak",1:nClusters,sep="")
> print(xtable(PSdataB515, caption =
+ "Extracted Gaussian (g), Robust (r) peak statistics and associated RDs.
+ NE is the number of events.", label = "tab:one"))
```

	NE	gMeans	rMeans	gSDs	rSDs	gRDs	rRDs
Peak1	4043.00	-3.46	0.00	15.65	17.12	0.00	0.00
Peak2	3752.00	43.88	39.90	20.76	18.68	1.27	1.08
Peak3	6367.00	912.38	909.30	119.67	116.75	6.14	6.37
Peak4	6592.00	4062.14	4057.20	266.34	250.63	8.14	8.55
Peak5	6193.00	20720.57	20701.80	757.83	628.92	16.25	18.90
Peak6	5941.00	50591.43	50541.75	1755.22	1223.59	11.88	16.10
Peak7	12543.00	159928.89	135300.89	49638.21	63818.13	2.13	1.30

Table 1: Extracted Gaussian (g), Robust (r) peak statistics and associated RDs. NE is the number of events.

Multi-level Kmeans

Estimations of MFIs and SDs values using multi level Kmeans is implemented in the functions "MultilevelgPS" and "MultilevelrPS" in flowQB. The function "MultilevelrPS" uses robust statsitics and "MultilevelgPS" uses Gaussian distribution fitting R software "MASS" package to extract the statistics for the peak found by multi-Kmeans.

```
> mrps=MultilevelrPS(TSSINGLETS[,2:4],OSSINGLETS[,2:4],7)
> mgps=MultilevelgPS(TSSINGLETS[,2:4],OSSINGLETS[,2:4],7)
> library(xtable)
> print(xtable(mrps, caption = "Multi level Kmeans data with Robust (r) statistics  for  pea
+ NE is the number of events.", label = "tab:mr"))


> print(xtable(mgps, caption =
+ "Multi level Kmeans data with Gaussian (g) statistics  for  peaks.
+ NE is the number of events.", label = "tab:mg"))
```

	NE	rMean1	rMean2	rMean3	rSD1	rSD2	rSD3
5	7790.00	7.35	16.80	4.60	28.02	29.58	17.73
1	6369.00	259.35	909.30	27.60	76.28	116.75	20.46
6	6592.00	822.15	4057.20	105.80	132.32	250.63	30.01
2	6197.00	1690.50	20702.85	264.04	189.92	628.92	45.01
7	5933.00	8896.65	50541.75	1535.48	470.13	1222.03	139.13
4	6283.00	28967.40	110493.59	4846.56	938.71	2152.96	369.64
3	6267.00	73636.50	209414.09	10860.60	2303.96	4162.70	803.39

Table 2: Multi level Kmeans data with Robust (r) statistics for peaks. NE is the number of events.

	NE	gMean1	gMean2	gMean3	gSD1	gSD2	gSD3
3	7790.00	10.98	19.25	4.61	32.38	29.73	17.10
2	6369.00	263.58	911.56	28.36	78.69	121.17	21.05
6	6592.00	827.62	4060.12	107.18	134.08	269.94	30.48
4	6197.00	1695.77	20736.70	265.30	192.95	929.03	46.12
5	5933.00	8904.95	50587.02	1536.88	523.51	2377.19	140.51
7	6283.00	28960.63	110403.44	4851.75	1117.84	3487.02	659.41
1	6267.00	73542.13	209444.50	10849.86	2625.81	4847.67	745.62

Table 3: Multi level Kmeans data with Gaussian (g) statistics for peaks. NE is the number of events.

Model Fitting uses Weighted Regression Method

We implemented the weighted least squares regression by using the function (lm) in the R "utils" package [7] in an iterative fashion to define the most probable weights, for more details about weights derivation see Supplemental Information. For the first iteration, the weights are, $(\omega_i = \frac{n_i - 1}{2(MFI_i^2)})_{1 \leq i \leq n}$, where MFI_i is the mean fluorescent intensity associated to the cluster (Peak) i and n_i its number of events and n is the number of the clusters used in the fitting model. The initial estimates of $(c_j)_{j=0,1,2}$ from the quadratic equation $SD^2 = c_0 + c_1 * MFI + c_2 * MFI^2$ are used to evaluate the weights, $(\omega_i = \frac{n_i - 1}{2(c_0 + c_1 * MFI_i + c_2 * MFI_i^2)^2})_{1 \leq i \leq n}$, and to rerun this computation until it converges.

Peaks for model fitting:

```
> pi=1
> pf=6
```

CVs are:

```
> rPSdataB515=data.frame(NE=rps[,1],rMeans=rps[,2]
+  ,rSDs=rps[,3])
> CVs=rPSdataB515[,3]/rPSdataB515[,2]
```

No calibration is set:

```
> p=1
> Q2D=MFI2MESF(rPSdataB515[,2:3],p,0)
```

Ordinary quadratic regression is used to get the first coefficients and further 5 iterations using weighted quadratic regression are processed:

```
> NE=rPSdataB515[pi:pf,1]
> S=Q2D[pi:pf,2]
> c0=0
> c1=1
> c2=0
> WWs=0.5*(NE-1)/((1+c2*S^2+c1*S+c0)^2)
> for( iter in 1:6)
+ {
+ QQBw=qrWEIGHTEDMESF(Q2D,pi,pf,WWs)
+ COFS=c(c0,as.numeric(QQBw)[5],as.numeric(QQBw)[8],c1,as.numeric(QQBw)[6],as.numeric(QQBw)[
+ if(iter==2) COEFFS=COFS
+ if(iter>2) COEFFS=rbind(COEFFS,COFS)
+ c1=1/as.numeric(QQBw)[1]
+ c2=as.numeric(QQBw)[4]
+ c0=as.numeric(QQBw)[2]/as.numeric(QQBw)[1]
+ WWs=0.5*(NE-1)/((1+c2*S^2+c1*S+c0)^2)
+ }
```

The weights are

```
> print(WWs)
```

[1] 2.078501e+03 1.865192e-04 5.481321e-09 1.596194e-11 1.012754e-14 [6] 4.773365e-17

```
>
```

So now we are ready to turn the $COEFFS$ into Table 4, using

```
> COEFFS=data.frame(COEFFS)
> names(COEFFS)= c( "c0" , "Pc0" ,"StdErrorc0", "c1" , "Pc1","StdErrorc1","c2percent" , "Pc2
> row.names(COEFFS)=paste("Iter",1:5,sep="")
> print(xtable(COEFFS, caption = "Computed coefficients $(c_{0},c_{1},c_{2})$
+ using weighted quadratic regression where $Pc_{i}$ is the pvalue associated to $c_{i}$ and
```

The $Q = \frac{1}{c_1}$, $B = \frac{c_0}{c_1}$ and $CV_{intrinsic} = \sqrt{c_2}$ quantities are,

```
> Q=1/c1
> B=c0/c1
> CVintrinsic=sqrt(c2)
> print(paste("Detection efficiency-Q=",round(Q,4)))
```

[1] "Detection efficiency-Q= 0.1215"

	c0	Pc0	StdErrorc0	c1	Pc1	StdErrorc1	c2percent	Pc2	StdErrorc2
Iter1	-0.00	1.00	0.01	11.32	0.01	1.39	0.04	0.26	0.00
Iter2	-0.00	1.00	0.01	9.31	0.00	0.64	0.16	0.17	0.00
Iter3	-0.00	1.00	0.00	8.39	0.00	0.38	0.30	0.14	0.00
Iter4	-0.00	1.00	0.00	8.25	0.00	0.34	0.34	0.14	0.00
Iter5	-0.00	1.00	0.00	8.23	0.00	0.33	0.35	0.14	0.00

Table 4: Computed coefficients (c_0, c_1, c_2) using weighted quadratic regression where Pc_i is the pvalue associated to c_i and $StdErrorc_i$ is Std. Error of c_i.

```
> print(paste("Background illumination-B=",round(B,4)))
```

[1] "Background illumination-B= 0"

```
> print(paste("Intrinsic CV of the beads-CVintrinsic=",round(CVintrinsic,4)))
```

[1] "Intrinsic CV of the beads-CVintrinsic= 0.0593"
The discriminant Δ and the roots are:

```
> delta=c1^2-c0*c2*4
> R1=-c1-sqrt(delta)
> R1=R1/(2*c2)
> R2=-c1+sqrt(delta)
> R2=R2/(2*c2)
> print(paste("Discriminant Delta=",round(delta,4)))
```

[1] "Discriminant Delta= 67.7394"

```
> print(paste("Root1=",round(R1,4)))
```

[1] "Root1= -2342.2664"

```
> print(paste("Root2=",round(R2,4)))
```

[1] "Root2= 0"

Robust statistics from Multi-level Kmeans and Q , B and $CV_{intrinsic}$ calculation

CVs are:

```
> mrPSdataB515=data.frame(NE=mrps[,1],rMeans=mrps[,3]
+  ,rSDs=mrps[,6])
> CVs=mrPSdataB515[,3]/mrPSdataB515[,2]
```

No calibration is set:

```
> p=1
> Q2D=MFI2MESF(mrPSdataB515[,2:3],p,0)
```

Ordinary quadratic regression is used to get the first coefficients and further 5 iterations using weighted quadratic regression are processed:

```
> NE=mrPSdataB515[pi:pf,1]
> S=Q2D[pi:pf,2]
> c0=0
> c1=1
> c2=0
> WWs=0.5*(NE-1)/((1+c2*S^2+c1*S+c0)^2)
> for( iter in 1:6)
+ {
+ QQBw=qrWEIGHTEDMESF(Q2D,pi,pf,WWs)
+ COFS=c(c0,as.numeric(QQBw)[5],as.numeric(QQBw)[8],c1,as.numeric(QQBw)[6],as.numeric(QQBw)[
+ if(iter==2) COEFFS=COFS
+ if(iter>2) COEFFS=rbind(COEFFS,COFS)
+ c1=1/as.numeric(QQBw)[1]
+ c2=as.numeric(QQBw)[4]
+ c0=as.numeric(QQBw)[2]/as.numeric(QQBw)[1]
+ WWs=0.5*(NE-1)/((1+c2*S^2+c1*S+c0)^2)
+ }
```

The weights are

```
> print(WWs)
```

[1] 2.829531e-05 5.427293e-08 8.781401e-10 1.391224e-12 7.865455e-15 [6] 9.407360e-17

```
>
```

So now we are ready to turn the $COEFFS$ into Table 5, using

```
> COEFFS=data.frame(COEFFS)
> names(COEFFS)= c( "c0" , "Pc0" ,"StdErrorc0", "c1" , "Pc1","StdErrorc1","c2percent" , "Pc2
> row.names(COEFFS)=paste("Iter",1:5,sep="")
> print(xtable(COEFFS, caption = "Multi-level Kmeans-Robust: Computed coefficients $(c_{0},c
+ using weighted quadratic regression where $Pc_{i}$ is the pvalue associated to $c_{i}$ and
```

For multi-level Kmeans-Robust, the $Q = \frac{1}{c_1}$, $B = \frac{c_0}{c_1}$ and $CV_{intrinsic} = \sqrt{c_2}$ quantities are,

```
> Q=1/c1
> B=c0/c1
> CVintrinsic=sqrt(c2)
> print(paste("Detection efficiency-Q=",round(Q,4)))
```

[1] "Detection efficiency-Q= 0.0707"

```
> print(paste("Background illumination-B=",round(B,4)))
```

	c0	Pc0	StdErrorc0	c1	Pc1	StdErrorc1	c2percent	Pc2	StdErrorc2
Iter1	540.62	0.00	2.74	14.19	0.00	0.07	0.03	0.00	0.00
Iter2	541.46	0.00	2.71	14.15	0.00	0.07	0.03	0.00	0.00
Iter3	541.47	0.00	2.71	14.15	0.00	0.07	0.03	0.00	0.00
Iter4	541.47	0.00	2.71	14.15	0.00	0.07	0.03	0.00	0.00
Iter5	541.47	0.00	2.71	14.15	0.00	0.07	0.03	0.00	0.00

Table 5: Multi-level Kmeans-Robust: Computed coefficients (c_0, c_1, c_2) using weighted quadratic regression where Pc_i is the pvalue associated to c_i and $StdErrorc_i$ is Std. Error of c_i.

[1] "Background illumination-B= 38.2639"

```
> print(paste("Intrinsic CV of the beads-CVintrinsic=",round(CVintrinsic,4)))
```

[1] "Intrinsic CV of the beads-CVintrinsic= 0.0163"
And the discriminant Δ and the roots are:

```
> delta=c1^2-c0*c2*4
> R1=-c1-sqrt(delta)
> R1=R1/(2*c2)
> R2=-c1+sqrt(delta)
> R2=R2/(2*c2)
> print(paste("Discriminant Delta=",round(delta,4)))
```

[1] "Discriminant Delta= 199.6707"

```
> print(paste("Root1=",round(R1,4)))
```

[1] "Root1= -53180.4797"

```
> print(paste("Root2=",round(R2,4)))
```

[1] "Root2= -38.2914"

Gaussian statistics from Multi-level Kmeans and Q , B and $CV_{intrinsic}$ calculation

CVs are:

```
> mgPSdataB515=data.frame(NE=mgps[,1],rMeans=mgps[,3]
+  ,rSDs=mgps[,6])
> CVs=mgPSdataB515[,3]/mgPSdataB515[,2]
```

No calibration is set:

```
> p=1
> Q2D=MFI2MESF(mgPSdataB515[,2:3],p,0)
```

Ordinary quadratic regression is used to get the first coefficients and further 5 iterations using weighted quadratic regression are processed:

```
> NE=mgPSdataB515[pi:pf,1]
> S=Q2D[pi:pf,2]
> c0=0
> c1=1
> c2=0
> WWs=0.5*(NE-1)/((1+c2*S^2+c1*S+c0)^2)
> for( iter in 1:6)
+ {
+ QQBw=qrWEIGHTEDMESF(Q2D,pi,pf,WWs)
+ COFS=c(c0,as.numeric(QQBw)[5],as.numeric(QQBw)[8],c1,as.numeric(QQBw)[6],as.numeric(QQBw)[
+ if(iter==2) COEFFS=COFS
+ if(iter>2) COEFFS=rbind(COEFFS,COFS)
+ c1=1/as.numeric(QQBw)[1]
+ c2=as.numeric(QQBw)[4]
+ c0=as.numeric(QQBw)[2]/as.numeric(QQBw)[1]
+ WWs=0.5*(NE-1)/((1+c2*S^2+c1*S+c0)^2)
+ }
```

The weights are

```
> print(WWs)
```

[1] 2.370749e-05 2.157736e-08 1.215902e-10 8.823906e-15 4.768401e-18 [6] 2.364113e-19

```
>
```

So now we are ready to turn the $COEFFS$ into Table 5, using

```
> COEFFS=data.frame(COEFFS)
> names(COEFFS)= c( "c0" , "Pc0" ,"StdErrorc0", "c1" , "Pc1","StdErrorc1","c2percent" , "Pc2
> row.names(COEFFS)=paste("Iter",1:5,sep="")
> print(xtable(COEFFS, caption = "Multi-level Kmeans-Gaussian: Computed coefficients $(c_{0}
+ using weighted quadratic regression where $Pc_{i}$ is the pvalue associated to $c_{i}$ and
```

For multi-level Kmeans-Gaussian, the $Q = \frac{1}{c_1}$, $B = \frac{c_0}{c_1}$ and $CV_{intrinsic} = \sqrt{c_2}$ quantities are,

```
> Q=1/c1
> B=c0/c1
> CVintrinsic=sqrt(c2)
> print(paste("Detection efficiency-Q=",round(Q,4)))
```

[1] "Detection efficiency-Q= 0.0676"

```
> print(paste("Background illumination-B=",round(B,4)))
```

	c0	Pc0	StdErrorc0	c1	Pc1	StdErrorc1	c2percent	Pc2	StdErrorc2
Iter1	516.08	0.00	3.27	14.62	0.00	0.13	0.11	0.00	0.00
Iter2	513.55	0.00	4.01	14.79	0.00	0.15	0.08	0.00	0.00
Iter3	513.71	0.00	3.99	14.78	0.00	0.15	0.08	0.00	0.00
Iter4	513.71	0.00	3.99	14.78	0.00	0.15	0.08	0.00	0.00
Iter5	513.71	0.00	3.99	14.78	0.00	0.15	0.08	0.00	0.00

Table 6: Multi-level Kmeans-Gaussian: Computed coefficients (c_0, c_1, c_2) using weighted quadratic regression where Pc_i is the pvalue associated to c_i and $StdErrorc_i$ is Std. Error of c_i.

[1] "Background illumination-B= 34.747"

```
> print(paste("Intrinsic CV of the beads-CVintrinsic=",round(CVintrinsic,4)))
```

[1] "Intrinsic CV of the beads-CVintrinsic= 0.0279"
And the discriminant Δ and the roots are:

```
> delta=c1^2-c0*c2*4
> R1=-c1-sqrt(delta)
> R1=R1/(2*c2)
> R2=-c1+sqrt(delta)
> R2=R2/(2*c2)
> print(paste("Discriminant Delta=",round(delta,4)))
```

[1] "Discriminant Delta= 216.9754"

```
> print(paste("Root1=",round(R1,4)))
```

[1] "Root1= -18957.6341"

```
> print(paste("Root2=",round(R2,4)))
```

[1] "Root2= -34.8108"

References

[1] J. Wood, *Fundamental Flow Cytometer Properties Governing Sensitivity and Resolution*, Cytometry 33, (1998), p. 260 - 6.

[2] R. Hoffman and J. Wood, *Characterization of Flow Cytometer Instrument Sensitivity*, Current Protocols in Cytometry, Chapter 1: Unit 1.20 (2007).

[3] E. Chase and R. Hoffman, *Resolution of Dimly Fluorescent Particles: a Practical Measure of Fluorescence Sensitivity*, Cytometry 33 (1998), p. 267-279.

[4] D. R. Parks, m. Roederer, W. A. Moore, *A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data.* Cytometry Part (A), (2006), 96(6), p. 541-51.

[5] A. Ortyn, E. Hall, C. George, K. Frost, A. Basiji, J. Perry, A. Zimmerman, D. Coder, P. Morrissey, *Sensitivity measurement and compensation in spectral imaging*, Cytometry 69, (2006), p. 852 - 862.

[6] *Robust Statistics in BD FACSDivaTM 6.0 Software*, BD Tech Note #23 − 9609 − 00. (2007).

[7] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, (2011),

[8] R. Ihaka and R. Gentleman R. *A Language for Data Analysis and Graphics*, Journal of Computational and Graphical Statistics, vol. 5, No. 3, (1996), p. 299-314.

[9] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, Springer, Fourth Edition, New York, (2002).

[10] F. El Khettabi et al. 2013, *Automated Quadratic Characterization of Flow Cytometer Instrument Sensitivit*, to be submitted.