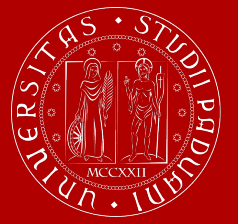
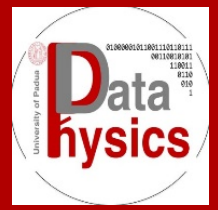
Study of the energy resolution and uncertainties of germanium detectors using Bayesian methods

Advanced Statistics 20/21

Avella Michele
Giorgetti Sabrina
Ziliotto Filippo



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Index

- Germanium Detector: contributions to uncertainty
- Goals of the spectra analysis
- ^{241}Am , ^{137}Cs, ^{60}Co *spectra*
- ^{228} Th *spectra*

Germanium detector: gamma ray spectrometry

Germanium detectors are wildly used in the gamma-ray spectrometry field thanks to their excellent energy resolution.

Energy resolution

$$FWHM = \sqrt{w_d^2 + w_e^2}$$

Depends on:

- The statistics of the charge creation process
- The properties of the detector
- The electronics noise

$$w_d^2 = (2.355)^2 \cdot w \cdot F \cdot E_\gamma$$

Fano factor

$0 < F < 1$

Electron-hole production energy threshold in germanium

$w \sim 3\ eV$

Deposited energy of the photon

$$w_e^2$$

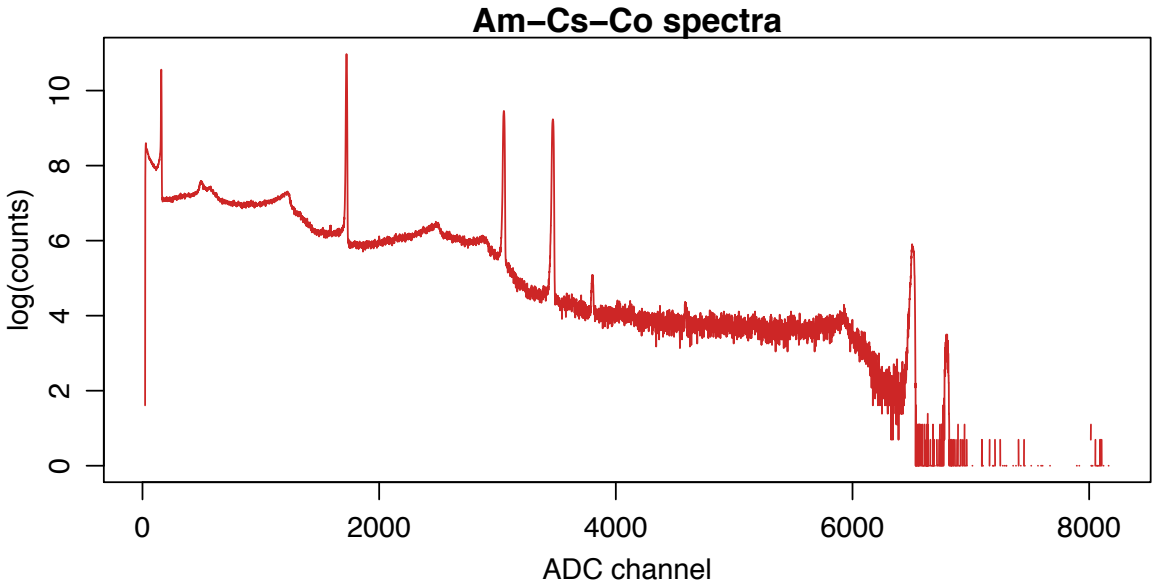
It is connected with the readout electronics and depends on the detector capacitance, the size of the detector and the bias voltage.

Uncalibrated energy spectra analysis

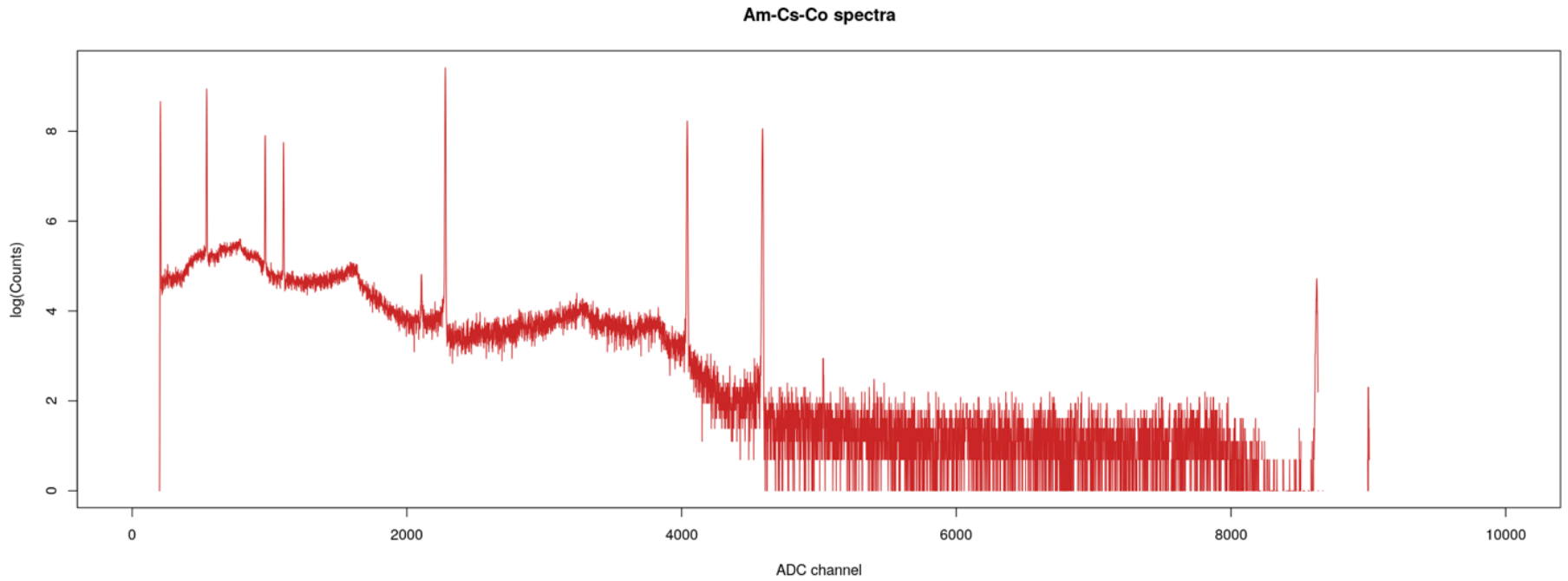
1. Infer the centroid of each γ peak for all available γ sources

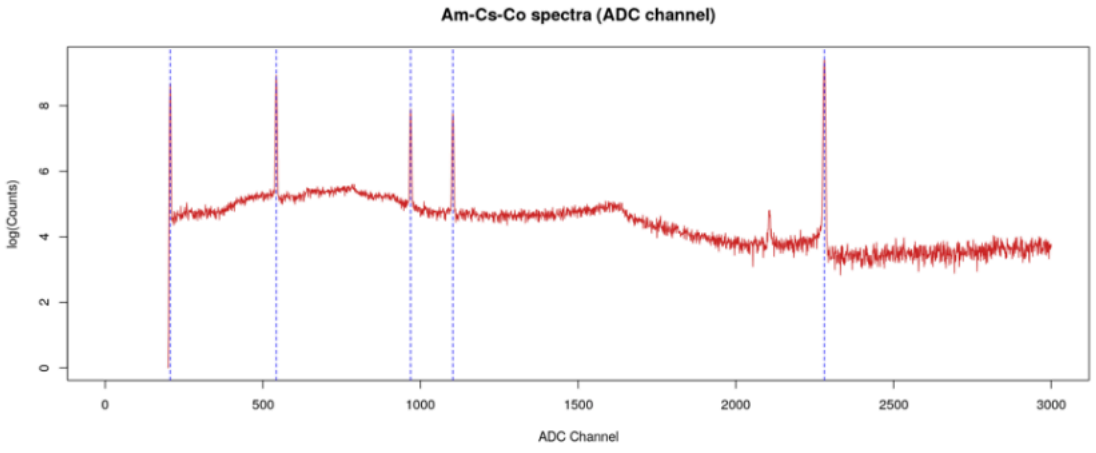
2. Perform a calibration of the detector: associating the centroid of each peak to the nominal value of the detected γ full energy peak

3. Study the behaviors of the energy resolution as a function of the photon energy and infer the other parameters



^{241}Am , ^{137}Cs, ^{60}Co spectra





Nuclide	Photon energy (keV)
^{241}Am	59.5409 keV
^{137}Cs	661.657 keV
^{60}Co (1)	1173.228 keV
^{60}Co (2)	1332.492 keV
^{60}Co (3)	2505.69 keV

1. Inference of the γ peaks

Our goal is to interfere the centroid of the peaks, that are distributed as Gaussian, in particular we know:

Counts N_i follow a Poisson distribution

$$P(N_i|S_i) = \frac{S_i^{N_i}}{N_i!} e^{S_i}$$

Signal S_i distributed as a Gaussian:

$$S_i = A\, e^{-\frac{(\mu - ADC_i)^2}{2\,\sigma^2}} + B$$

To infer μ and σ we can use Bayes theorem:

$$\boldsymbol{P(\mu, log(\sigma), A, B\,|\{N_i\}, I)}$$

$$Posterior \propto Prior \cdot Likelihood$$

$$\boldsymbol{P(\mu, \log\sigma,\ A, B\,|I) = U_{[\dots]}}$$

$$\boldsymbol{P(\{N_i\}|\mu, log\,\sigma, A, B, I) = \prod_{i=1}^{N} P(N_i|S_i)}$$

Inference of the γ peaks: JAGS model

JAGS Model

```
cat("
model {
    # data likelihood
    for (i in 1:length(N)) {
        N[i] ~ dpois(S[i]);
        S[i] <- B+A*exp(-((X[i]-mu)^2)/(2*sigma^2));
    }
    # prior
    mu ~ dunif(150, 2500);
    logsigma ~ dunif(-10,10);
    sigma <- exp(logsigma)
    A ~ dunif(1000,15000);
    B ~ dunif(0,1000);

}
",file = 'project.bug')
```

mu ~ center of the peak

sigma ~ width of the peak

A ~ signal amplitude

B ~ background amplitude

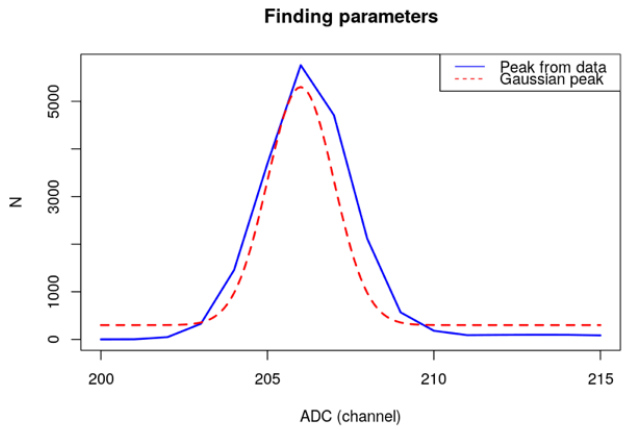
JAGS stands for *Just Another Gibbs Sampler.*

It is a program for analysis of Bayesian hierarchical models using Markov Chain Monte Carlo (MCMC) simulation.

Inference of the γ peaks: R implementation

1. Finding the initial parameters

```
inf <- 200
sup <- 215

x <- adc[inf:sup]
N <- counts[inf:sup]

p <- c(5000,300,206,1)

options(repr.plot.width = 7, repr.plot.height = 5)
plot(x,N,'l',lwd=2,col='blue',xlab='ADC (channel)',ylab='N',main='Finding parameters')
xx <- seq(min(x),max(x),0.1)
lines(xx,signal(xx,p[1],p[2],p[3],p[4]),lwd =2,col='red',lty=2)
legend("topright", legend=c("Peak from data", "Gaussian peak"),
       col=c("blue", "red"), lty=1:2, cex=1)
```



2. Running the JAGS model

```
JAGS.fit <- function(data,inits,nit,thin,burnin,title){
    jm <- jags.model(file = 'project.bug', inits = inits, data=data,quiet=TRUE,n.adapt=burnin)
    chain <- coda.samples(jm, c("A", "B", "mu",'sigma'), n.iter=nit,thin=thin)
    print(summary(chain)$statistics)

    x <- data$X
    N <- data$N
    p <- summary(chain)$statistics[1:4]

    options(repr.plot.width = 8, repr.plot.height = 6)
    plot(x,N,lwd=2,col='blue',main=title,xlab='ADC (channel)',ylab='N')
    xx <- seq(min(x),max(x),0.1)
    lines(xx,signal(xx,p[1],p[2],p[3],p[4]),lwd =2,col='red',lty=2)
    legend("topright", legend=c("JAGS", "Gaussian peak"),
       col=c("blue", "red"), lty=1:2, cex=1)

    return (chain)
}
```

^{241}Am

```
inits = list(   "mu" = 206,
                "logsigma" = log(1),
                "A" = 5000,
                "B" = 300)
data1 = list(N = N,X = x)

chain.am <- JAGS.fit(data1,inits,nit = 100000,thin = 100,burnin = 10000,title='Am')
sigma.fit <- append(sigma.fit, summary(chain.am)$statistics[4,1])
mu.fit <- append(mu.fit, summary(chain.am)$statistics[3,1])
```

^{137}Cs

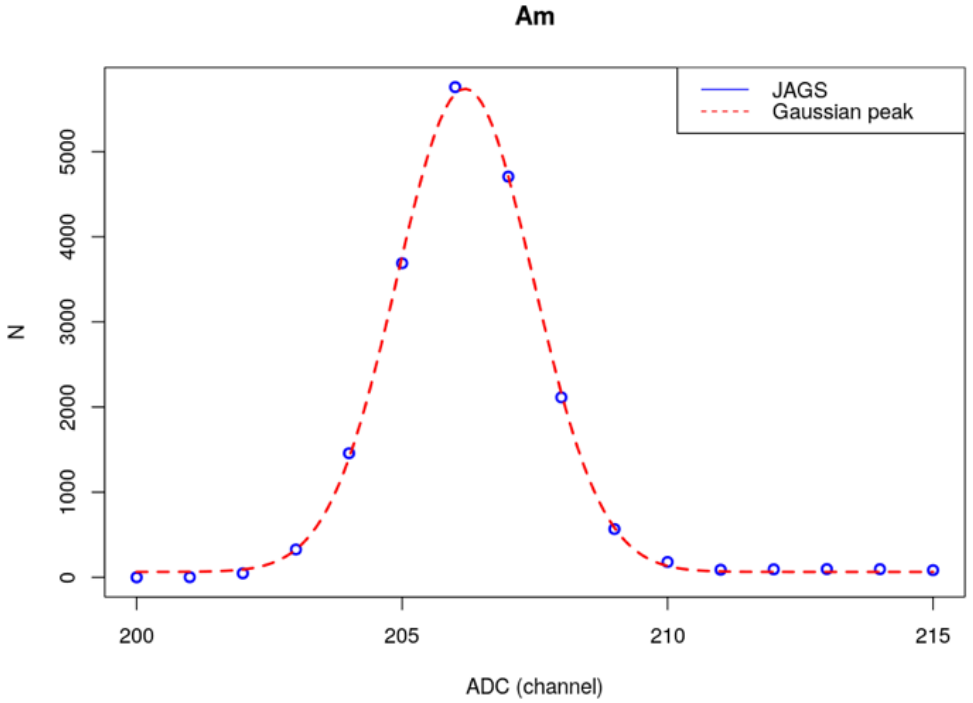
```
inits = list(   "mu" = 543,
                "logsigma" = log(2),
                "A" = 5000,
                "B" = 300)
data1 = list(N = N,X = x)

chain.cs <- JAGS.fit(data1,inits,nit = 100000,thin = 100,burnin = 10000,title='Cs')
sigma.fit <- append(sigma.fit, summary(chain.cs)$statistics[4,1])
mu.fit <- append(mu.fit, summary(chain.cs)$statistics[3,1])
```

^{60}Co

```
inits = list(   "mu" = 969,
                "logsigma" = log(2),
                "A" = 2000,
                "B" = 300)
data1 = list(N = N,X = x)

chain.co1 <- JAGS.fit(data1,inits,nit = 100000,thin = 100,burnin = 10000,title='Co[1]')
sigma.fit <- append(sigma.fit, summary(chain.co1)$statistics[4,1])
mu.fit <- append(mu.fit, summary(chain.co1)$statistics[3,1])
```

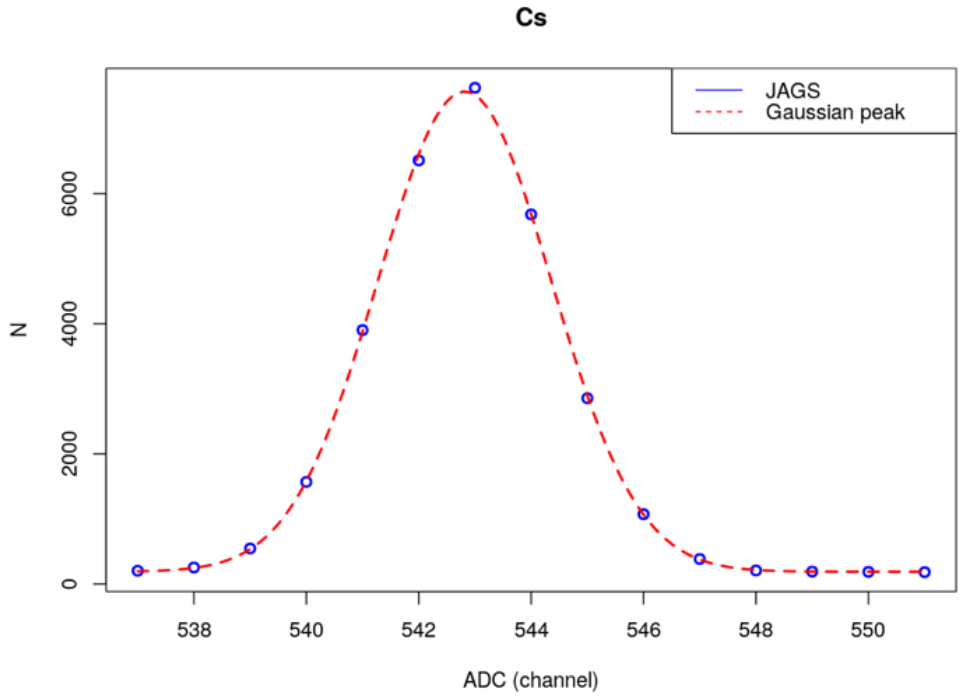
Inference of the γ peaks : results

^{241}Am



	Mean	SD
A	5674.203489	55.083911197
B	64.586364	2.925648066
mu	206.186165	0.009455878
sigma	1.285789	0.008361780

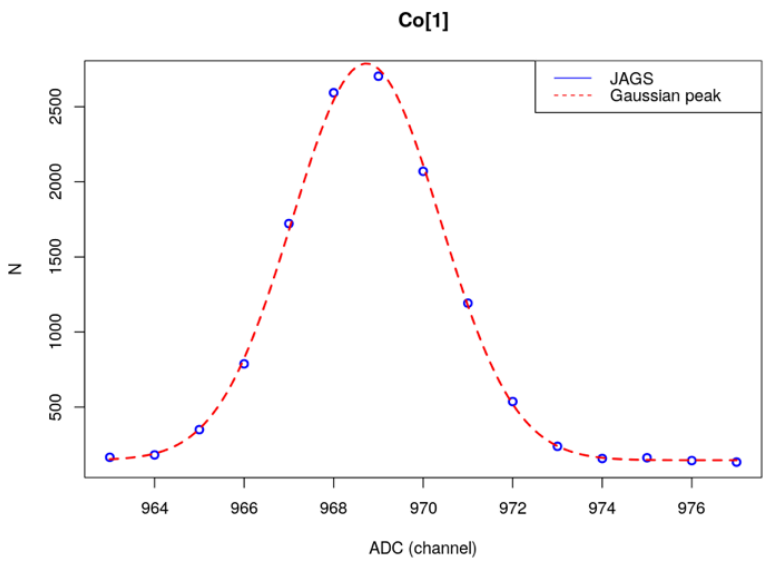
^{137}Cs



	Mean	SD
A	7375.008880	53.334286713
B	192.006147	6.110021013
mu	542.815813	0.009694614
sigma	1.542058	0.008785177

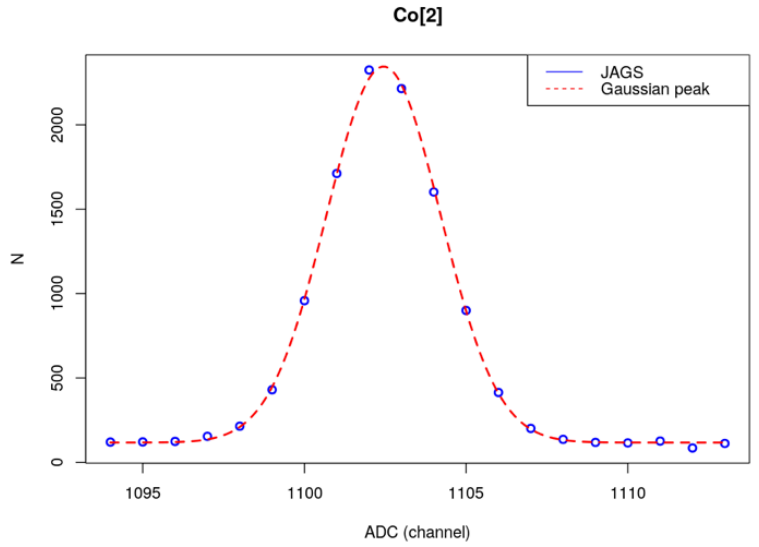
Inference of the γ peaks : results

$^{60}Co[1]$



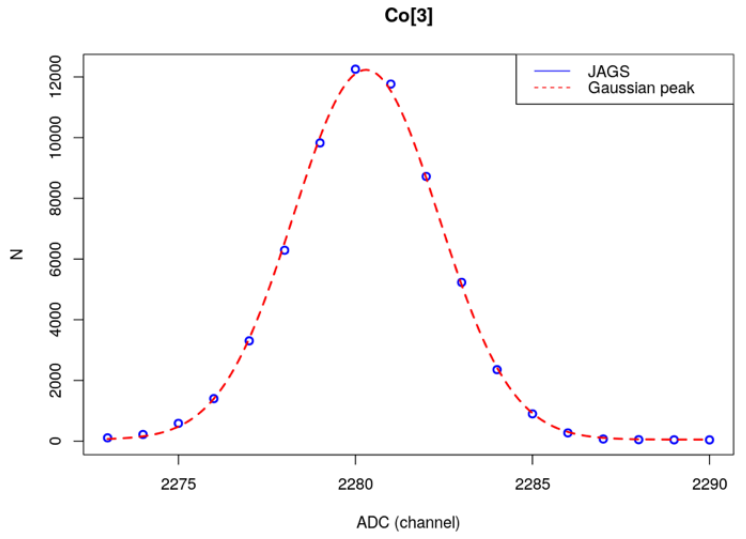
	Mean	SD
A	7375.008880	53.334286713
B	192.006147	6.110021013
mu	542.815813	0.009694614
sigma	1.542058	0.008785177

$^{60}Co[2]$



	Mean	SD
A	2228.850692	29.38554080
B	117.069507	3.43365043
mu	1102.445287	0.02050727
sigma	1.762973	0.01803294

$^{60}Co[3]$



	Mean	SD
A	12183.909366	58.985361618
B	50.792141	3.853141328
mu	2280.297488	0.008166550
sigma	2.047298	0.006920091

2. Calibration

Supposing a linear response of the detector, we can proceed to the conversion of ADC in energy as follows:

$$y = ax + b$$

$$ADC = m \cdot E_\gamma + q$$

$$E_\gamma = a \cdot ADC + b$$

Inferring the angular coefficient a and the intercept b we can obtain the corresponding energy values.

Data

	Energy	x0
	<dbl>	<dbl>
Am-241	59.5409	206.1862
Cs-137	661.6570	542.8158
C0-60.1	1173.2280	968.7286
C0-60.2	1332.4920	1102.4453
C0-60.3	2505.6900	2280.2975

1st Method - Linear regression

```
# The relation between ADC and Energy(keV) is considered to be linear of the form y=mx+c.
# lm() in R is a good tool to model them.
regression <- lm(formula =   df$x0 ~ df$Energy)
regression

# Extracting coefficients
m <- regression$coefficients[2]
q <- regression$coefficients[1]
a <- (1/m)
b <- (-q/m)

# changing adc channel axis to -->> to kev
# Energy is now simply obtained using the relation y=mx+c
energy <- a*data$x + b
cat('a:',a,'\n')
cat('b:',b)
```

Calibration: JAGS

2nd Method - JAGS

```
cat("model {

    # data likelihood
    for (i in 1:5){
        Y[i] ~ dnorm(mu[i],1/(eps[i]));
        mu[i] <- q + m*X[i];
    }

    #priors
    alpha ~ dunif(0,2);
    m <- tan(alpha)
    q ~ dunif(-100,100);

    a <- 1/m;
    b <- -q/m;

    #predictions
    x_in <- 2000 #in adc
    x_out <- 5000
    y_1 <- q + m*x_in;
    y_2 <- q + m*x_out;

}", file='method2.bug')


init <- NULL
init$alpha <- pi/4
init$q <- -4
jm <- jags.model(file='method2.bug',data, init=init)
update(jm,4000)
chain <- coda.samples(jm, c('a','b','y_1','y_2','m','q'), n.iter=2e5, thin=200)
print(summary(chain))
```

Using JAGS, the model chosen for the linear regression is:

$$y_i = ax_i + b + \epsilon_i$$

where ϵ_i is $\sigma_i = \sigma_{peak}$. To infer a and b we can use, once again, the Bayes theorem.

In this case the likelihood is:

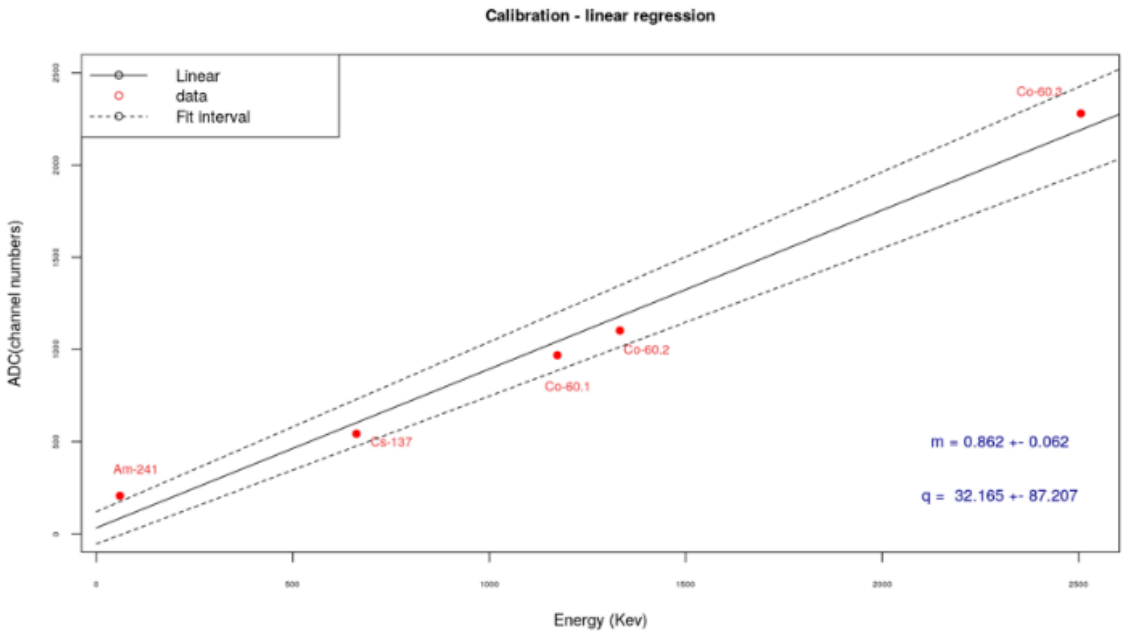
$$P(y_i|\sigma, a, b, x_i, I) = \prod_i \mathcal{N}\ (x = (y_i - x_i \cdot a\ -\ b), \mu = 0, \sigma = \sigma)$$

and the uniform prior is :

$$P(\arctan(a), b, \log(\sigma)|I) = U_{[\dots]}$$

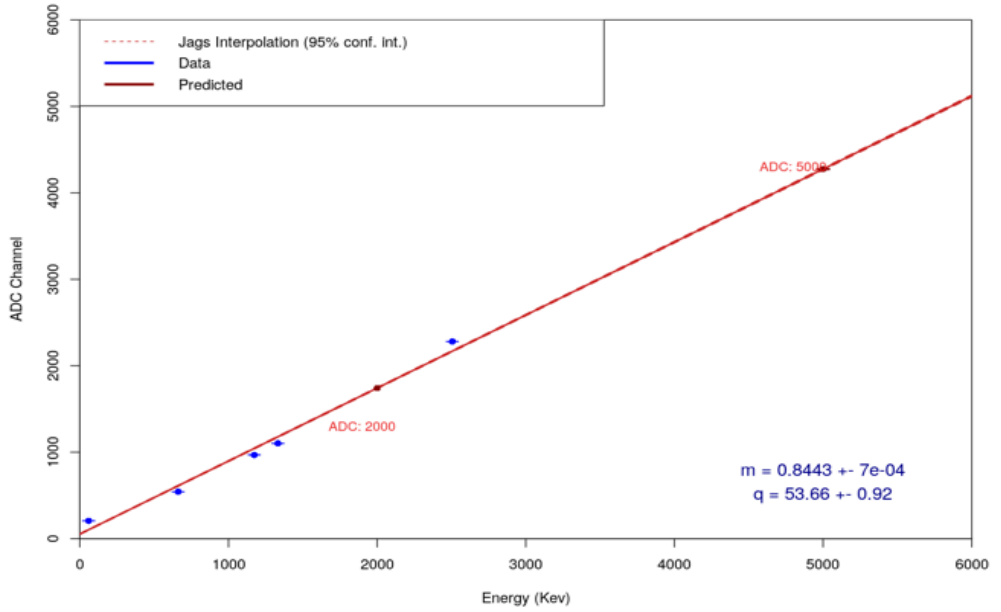
Calibration results

1st Method - Linear regression



$$a = 1.16 \pm 0.08$$
$$b = -37.33 \pm 117.48$$

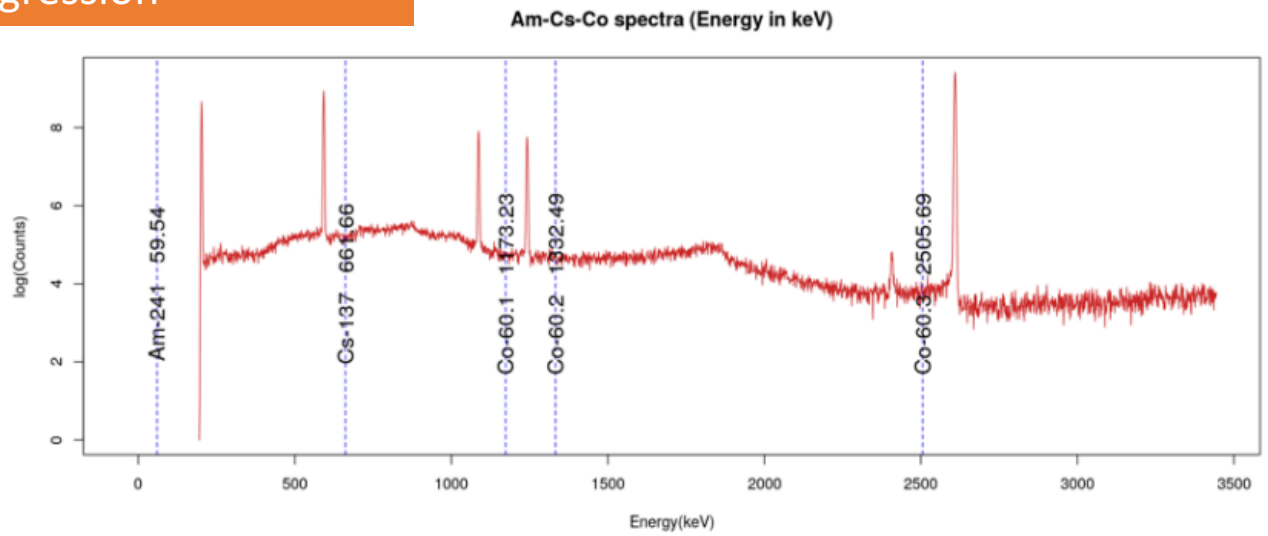
2nd Method - JAGS



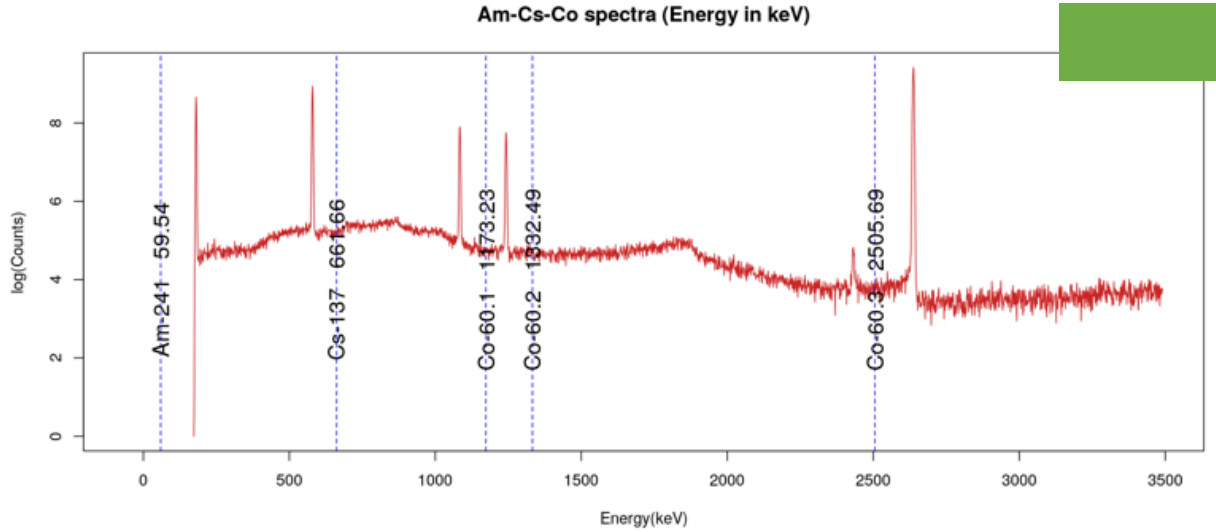
$$a = 1.184 \pm 0.001$$
$$b = -63.55 \pm 1.13$$

Calibration results

1st Method - Linear regression



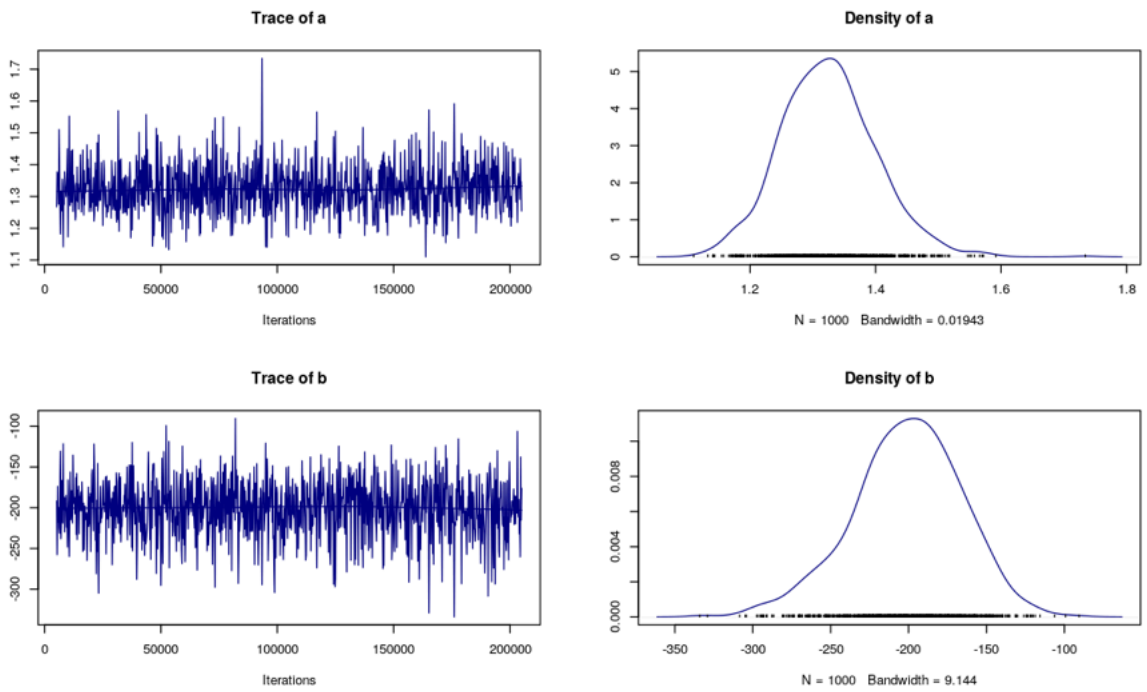
2nd Method - JAGS

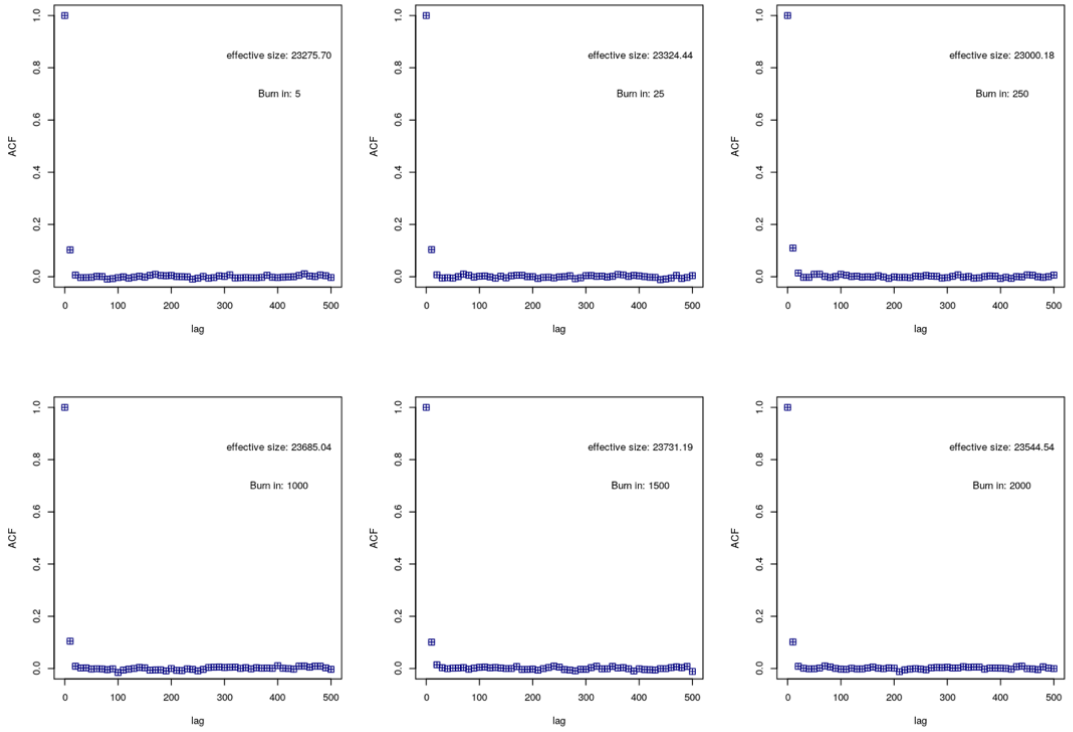


JAGS : chain and auto-correlation plot

- **Posterior** densities and trace of the chains
- As we see from the traces we sample mainly from **high-density regions**

- **Thinning** always > 100 for all the computations
- **Autocorrelation** in the chain goes to zero quickly for every burn-in value used

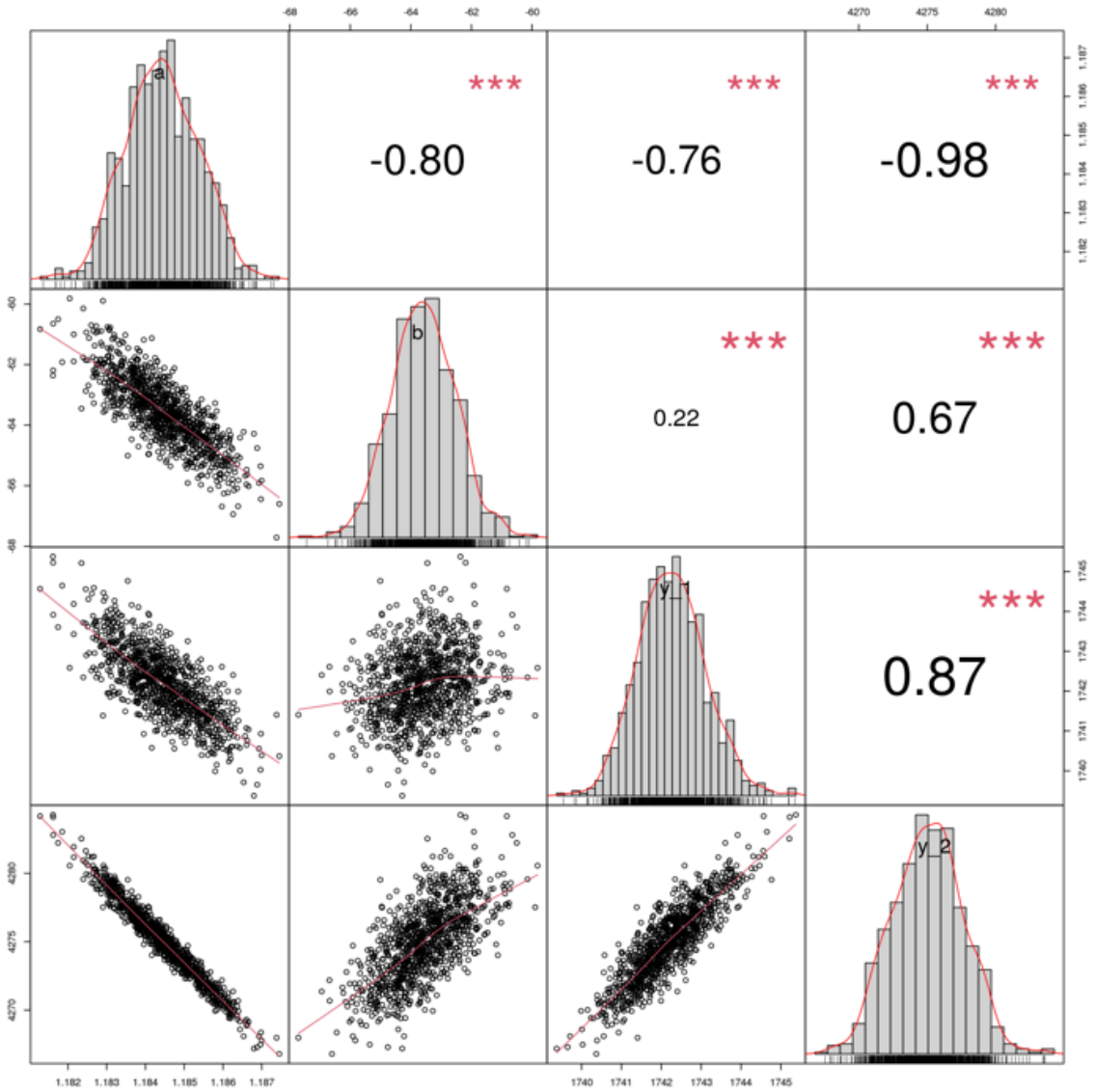




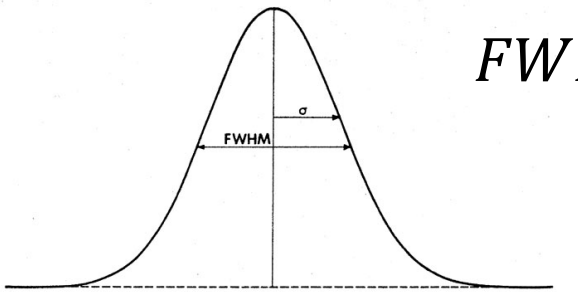
JAGS: correlation plot

```
library(PerformanceAnalytics))
options(repr.plot.width=12, repr.plot.height=12)
chain.df <- as.data.frame(as.mcmc(chain))
chain.df <- chain.df[,c('a','b','y_1','y_2')]
chart.Correlation(chain.df, histogram = T, pch= 19)
```

- **Correlation** between variables and posteriors histograms
- **a** and **b** from the fit are correlated as we expect
- Predicted **high energy** values have a correlation with the fit variables of -1



3. Study of the behaviour of FWHM as a function of E



$$FWHM \simeq 2.355\,\sigma$$

```
m <- 2.355
a <- a.jags
#sigma.fit
fwhm2 <- round((sigma.fit*a*m), digits=7)
#energy
AmCsCo <- c(59.5409,661.657,1173.228,1332.492,2505.69)

df <- data.frame(E=AmCsCo,FWHM=fwhm2)
rownames(df) <- c("Am-241", "Cs-137",  "C0-60.1", "C0-60.2", "C0-60.3")

options(digits=5)
df
```

Values of FWHM

	E	FWHM
	<dbl>	<dbl>
Am-241	59.541	3.5866
Cs-137	661.657	4.3014
C0-60.1	1173.228	4.6108
C0-60.2	1332.492	4.9177
C0-60.3	2505.690	5.7108

Inference of the others parameters: F and w_e

$$FWHM = \sqrt{w_d^2 + w_e^2}$$

$$w_d^2 = 2.355^2 \cdot w \cdot F \cdot E_\gamma$$

F and w_e

1st Method - Linear regression

```
m <- 2.355; a <- a.jags
fwhm <- round((sigma.fit*a*m)^2, digits=7)
AmCsCo <- c(59.5409,661.657,1173.228,1332.492,2505.69)
df <- data.frame(E=AmCsCo,FWHM=fwhm)

data <- NULL
data$X <- df$E
data$Y <- df$FWHM

# lm() in R for linear regression.
regression <- lm(formula =   df$FWHM~ df$E )
regression

# Extracting coefficients
a <- regression$coefficients[2]
b <- regression$coefficients[1]
fwhm <- a * data$x+ b
```

$$FWHM^2 = (2.355^2 \cdot w \cdot F) \cdot E_\gamma + w_e^2$$

$$y = k \cdot E_\gamma + q$$

$$F = \frac{k}{(2.355)^2 \cdot w}$$

$$w_e = \sqrt{q}$$

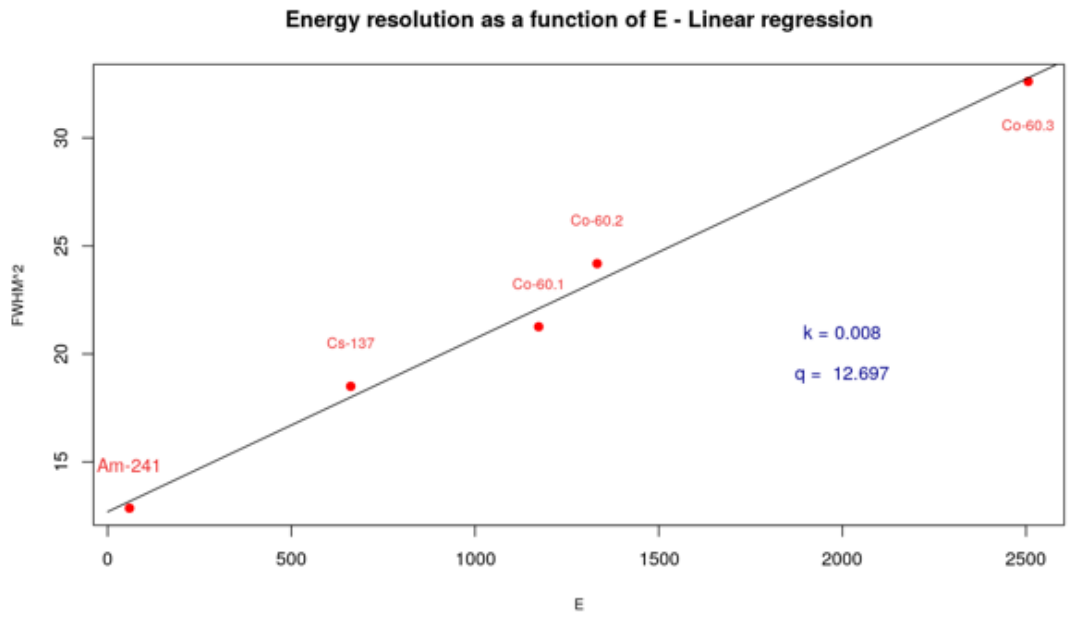
2nd Method - JAGS

$$FWHM_i = \sqrt{w_e^2 + (2.355^2 \cdot w \cdot F \cdot E_{\{\gamma,i\}})} + \epsilon_i$$

```
cat("model {

    # data likelihood
    for (i in 1:length(Y)){
        Y[i] ~ dnorm(mu[i],1/sigma);
        mu[i] <- (w_e^2+(2.355^2)*0.003*X[i]*f)^0.5;
    }

    #priors
    w_e ~ dunif(0,8);
    f ~ dunif(0,1);
    sigma <- exp(logsigma)
    logsigma ~ dunif(-20,1)

}", file='model3.bug')

init <- NULL

data <- NULL
data$X <- AmCsCo
data$Y <- fwhm2

jm <- jags.model(file='model3.bug',data, init=init)
update(jm,4000)
chain <- coda.samples(jm, c('f','w_e','sigma'), n.iter=2e5, thin=200)
print(summary(chain))
```

Inference of F and w_e: results

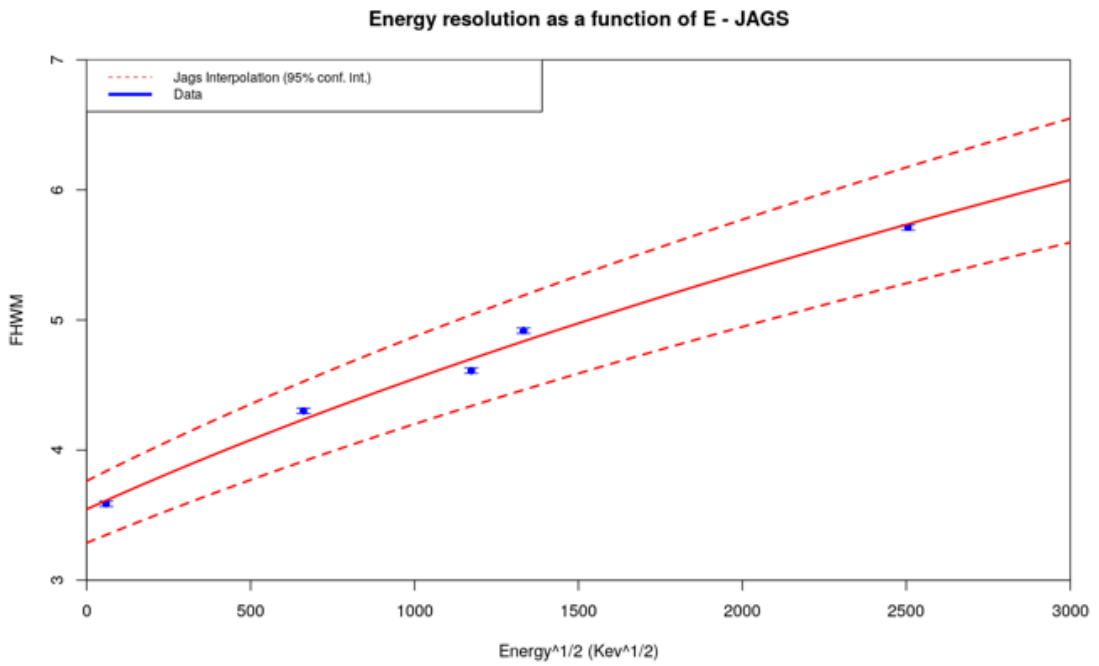
1st Method - Linear regression



$F \simeq 0.48$

$w_e \simeq 3.56$

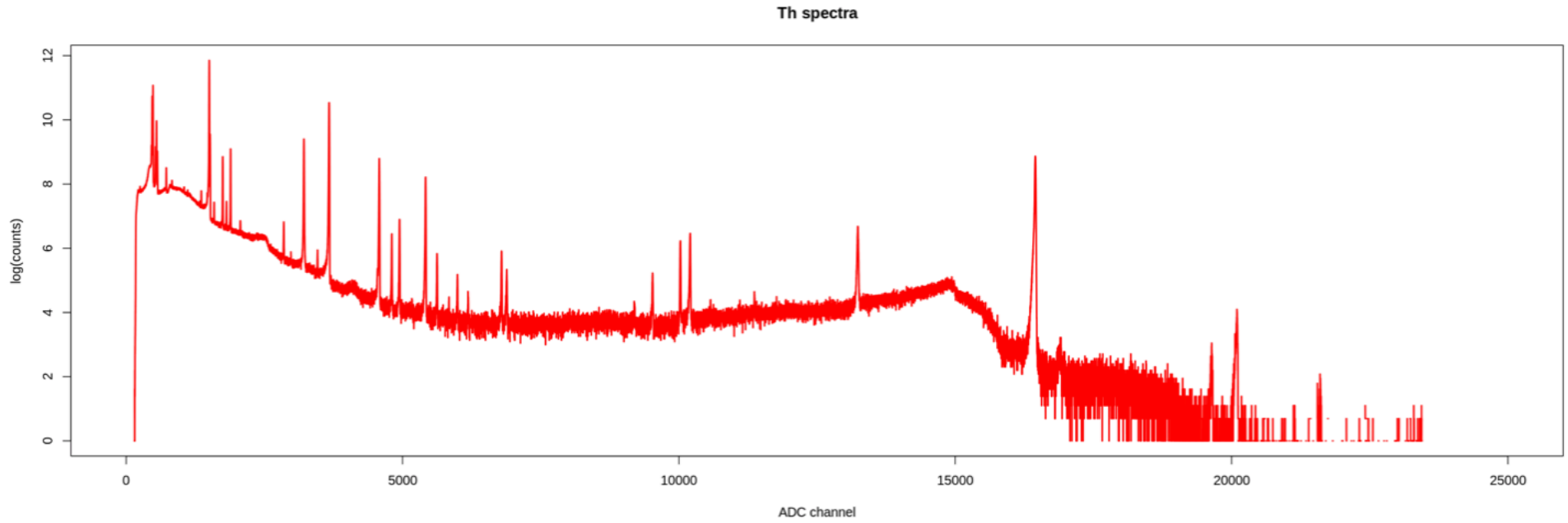
2nd Method - JAGS

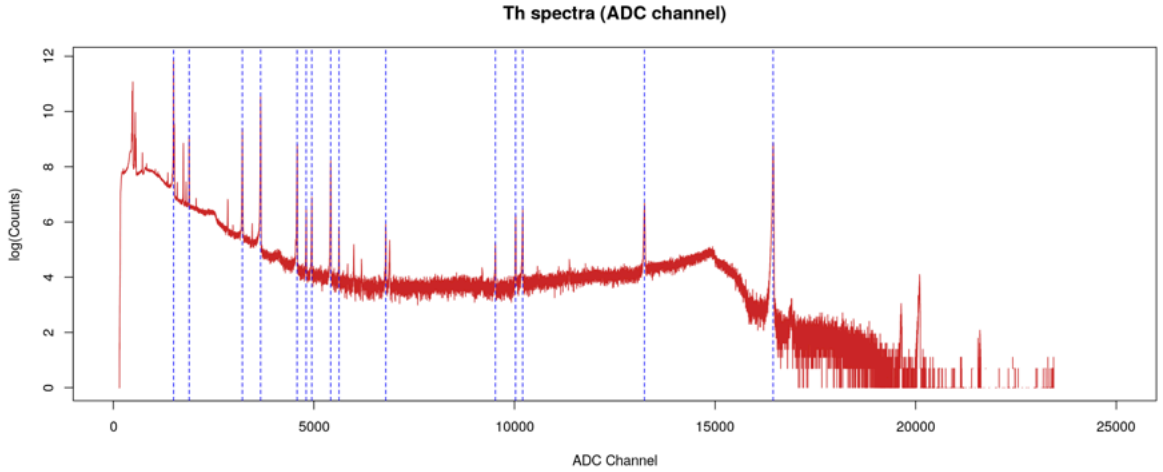


$F \simeq 0.48 \pm 0.04$

$w_e \simeq 3.54 \pm 0.12$

^{228}Th spectra

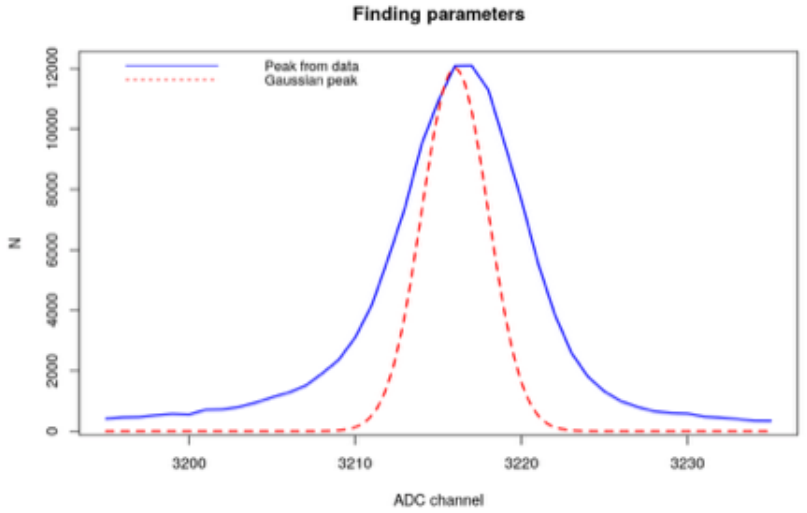




Photon energy (keV)		
238.632 keV	763.45 keV	1512.70 keV
300.089 keV	785.37 keV	1592.511 keV
510.74 keV	860.53 keV	1620.738 keV
583.187 keV	893.408 keV	2103.511 keV
727.330 keV	1078.63 keV	2614.511 keV

Inference of the γ peaks: R implementation

1. Finding the initial parameters



JAGS Model

```
cat("
model {
    # data likelihood
    for (i in 1:length(N)) {
        N[i] ~ dpois(S[i]);
        S[i] <- B+A*exp(-((X[i]-mu)^2)/(2*sigma^2));
    }
    # prior
    mu ~ dunif(1000, 18000);
    logsigma ~ dunif(-10,10);
    sigma <- exp(logsigma)
    A ~ dunif(0,145000);
    B ~ dunif(0,10000); #check B

}
",file = 'projectth.bug')
```

mu ~ center of the peak
sigma ~ width of the peak
A ~ signal amplitude
B ~ background amplitude

Inference of the γ peaks: R implementation

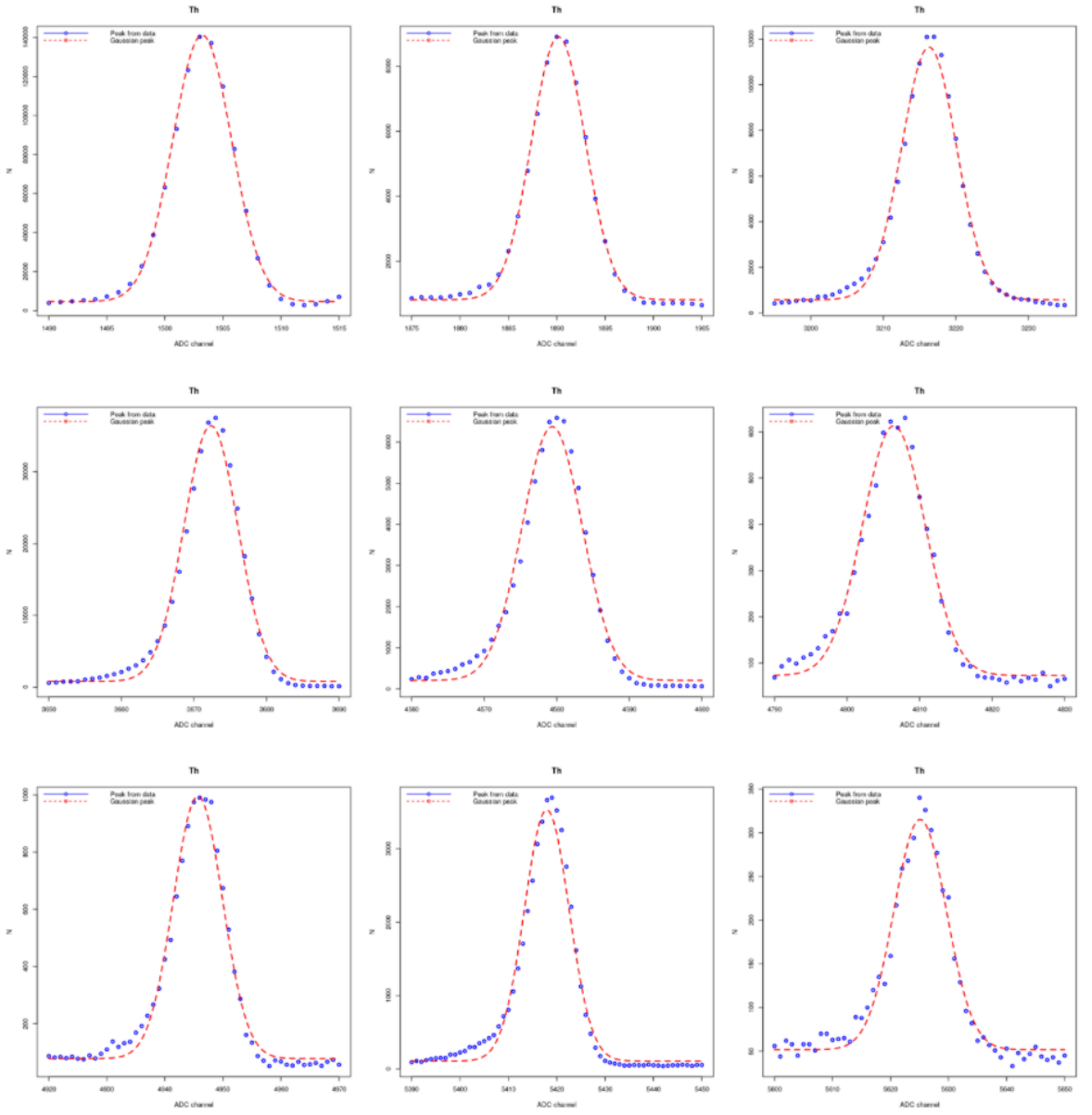
Running the JAGS model

```
JAGS.fit <- function(data,inits,nit,thin,burnin,title){
    jm <- jags.model(file = 'projectth.bug', inits = inits, data=data,quiet=TRUE,n.adapt=burnin)
    chain <- coda.samples(jm, c("A", "B", "mu",'sigma'), n.iter=nit,thin=thin)
    #print(summary(chain)$statistics)

    x <- data$X
    N <- data$N
    p <- summary(chain)$statistics[1:4]
    val <- summary(chain)$statistics[,1]

    #options(repr.plot.width = 8, repr.plot.height = 6)
    plot(x,N,lwd=2,col='blue',main="Th",xlab='adc',ylab='N')
    xx <- seq(min(x),max(x),0.1)
    lines(xx,signal(xx,p[1],p[2],p[3],p[4]),lwd =2,col='red',lty=2)

    return(chain)
}
```

```
for (i in 1:length(peaks)){
    inf <- peaks_low[i]
    sup <- peaks_high[i]

    x <- adc[inf:sup]
    N <- counts[inf:sup]


    p <- c(amplitude[i],offset[i],peaks[i],2)

    inits = list(   "mu" = peaks[i],
                    "logsigma" = log(1),
                    "A" = amplitude[i],
                    "B" = offset[i])
    data = list(N = N,X = x)

    chain <- JAGS.fit(data,inits,nit = 1e4,thin = 20,burnin = 1e3,title='Th')
    print(summary(chain)[[1]])
    sigma.fit <- append(sigma.fit, summary(chain)$statistics[4,1])
    mu.fit <- append(mu.fit, summary(chain)$statistics[3,1])


}
```

Inference of the γ peaks: results

	Mean	SD		Mean	SD		Mean	SD
	<dbl>	<dbl>		<dbl>	<dbl>		<dbl>	<dbl>
A	136611.346	188.100	**A5**	539.859	10.118	**A10**	133.415	4.758
B	4631.845	21.084	**B5**	73.422	2.143	**B10**	40.525	0.896
mu	1503.212	0.003	**mu5**	4806.453	0.076	**mu10**	9523.099	0.208
sigma	2.535	0.003	**sigma5**	4.293	0.078	**sigma10**	6.080	0.217
A1	8072.544	45.695	**A6**	916.802	12.876	**A11**	453.284	8.200
B1	824.512	7.490	**B6**	78.186	1.773	**B11**	46.437	1.538
mu1	1890.178	0.015	**mu6**	4945.702	0.053	**mu11**	10025.993	0.082
sigma1	2.777	0.015	**sigma6**	4.240	0.050	**sigma11**	5.244	0.079
A2	11060.130	46.016	**A7**	3419.896	21.667	**A12**	542.090	8.415
B2	580.019	6.217	**B7**	108.415	1.954	**B12**	66.977	1.678
mu2	3216.315	0.014	**mu7**	5417.938	0.028	**mu12**	10201.904	0.089
sigma2	3.779	0.014	**sigma7**	4.619	0.024	**sigma12**	5.942	0.088
A3	35607.249	77.695	**A8**	263.711	7.229	**A13**	629.332	7.650
B3	809.655	7.269	**B8**	51.699	1.489	**B13**	94.260	2.120
mu3	3672.347	0.007	**mu8**	5625.110	0.113	**mu13**	13236.987	0.105
sigma3	3.721	0.006	**sigma8**	4.509	0.118	**sigma13**	9.741	0.122
A4	6168.613	32.111	**A9**	309.553	7.434	**A14**	6362.683	24.134
B4	206.464	3.755	**B9**	52.732	1.496	**B14**	202.330	1.772
mu4	4579.372	0.019	**mu9**	6791.550	0.106	**mu14**	16447.612	0.027
sigma4	4.155	0.018	**sigma9**	4.700	0.102	**sigma14**	8.348	0.025



2. Calibration

1st Method - Linear regression

```
# The relation between ADC and Energy(keV) is considered to be linear, form y=mx+c.
# lm() in R is a good tool to model them.
regression <- lm(formula =   df$x0 ~ df$Energy)
regression

# Extracting coefficients
m <- regression$coefficients[2]
q <- regression$coefficients[1]

summary(regression)

# changing adc channel axis to -->> to kev
# Energy is now simply obtained using the relation y=mx+c
a <- 1/m
b <- -q/m
cat(paste('a:',round(a,2),'\n'))
cat(paste('b:',round(b,2),'\n'))
```

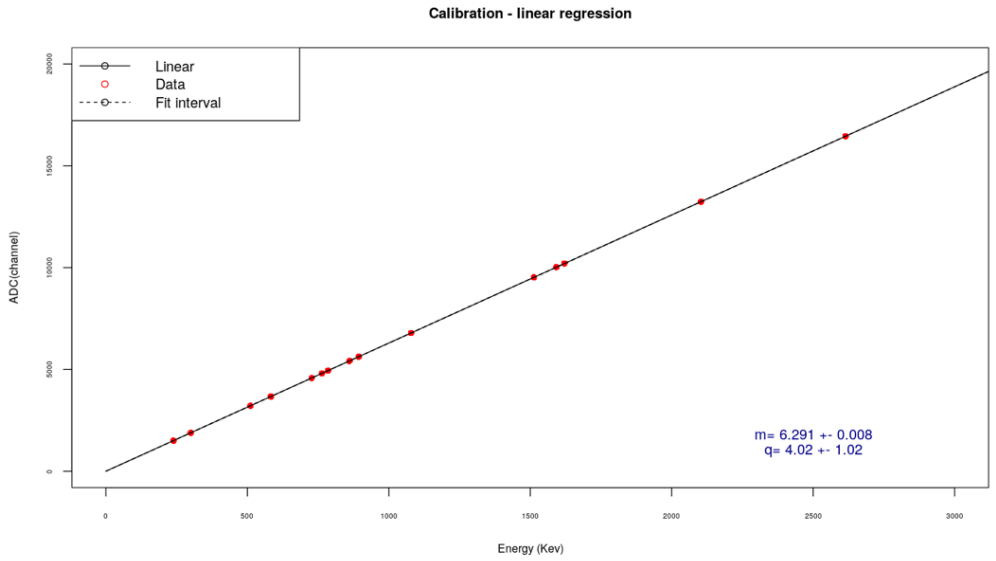
	Energy	x0
	<dbl>	<dbl>
Th1	238.6	1503
Th2	300.1	1890
Th3	510.7	3216
Th4	583.2	3672
Th5	727.3	4579
Th6	763.5	4806
Th7	785.4	4946
Th8	860.5	5418
Th9	893.4	5625
Th10	1078.6	6792
Th11	1512.7	9523
Th12	1592.5	10026
Th13	1620.7	10202
Th14	2103.5	13237
Th15	2614.5	16448

2nd Method - JAGS

```
cat("model {

    # data likelihood
    for (i in 1:length(X)){
        Y[i] ~ dnorm(mu[i],1/(sigma[i]));
        mu[i] <- q + m*X[i];
    }

    #priors
    m <- tan(alpha);
    alpha ~ dunif(0,2)

    q ~ dunif(-100,100);
    a <- (1/m);
    b <- (-q/m);


    #predictions
    x_in <- 1200
    x_out <- 3200
    y_1 <- q + m*x_in;
    y_2 <- q + m*x_out;

}", file='Th_method2.bug')


init <- NULL
init$alpha <- pi/4
```

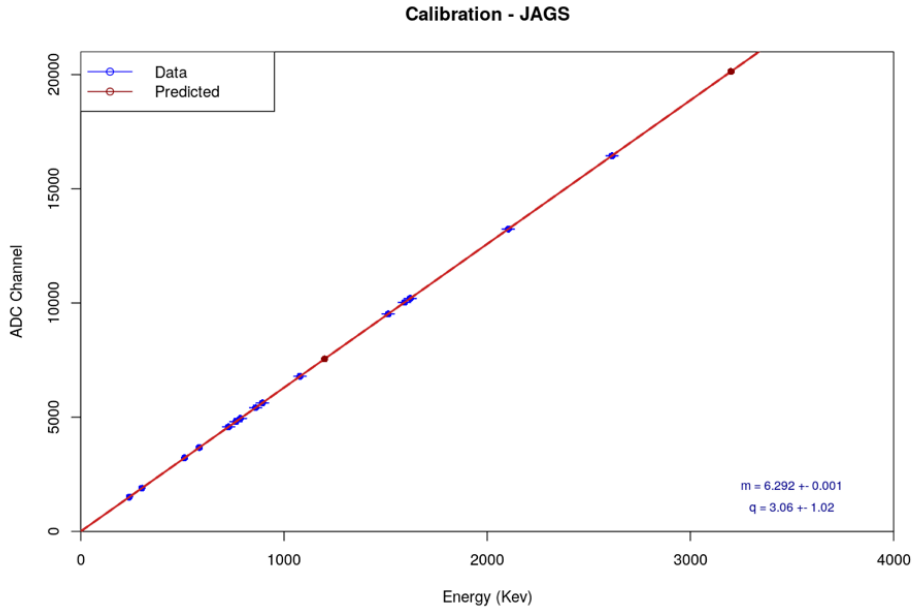
2. Calibration results

1st Method - Linear regression



$$a \simeq 0.16$$
$$b \simeq -0.64 \pm 0.25$$

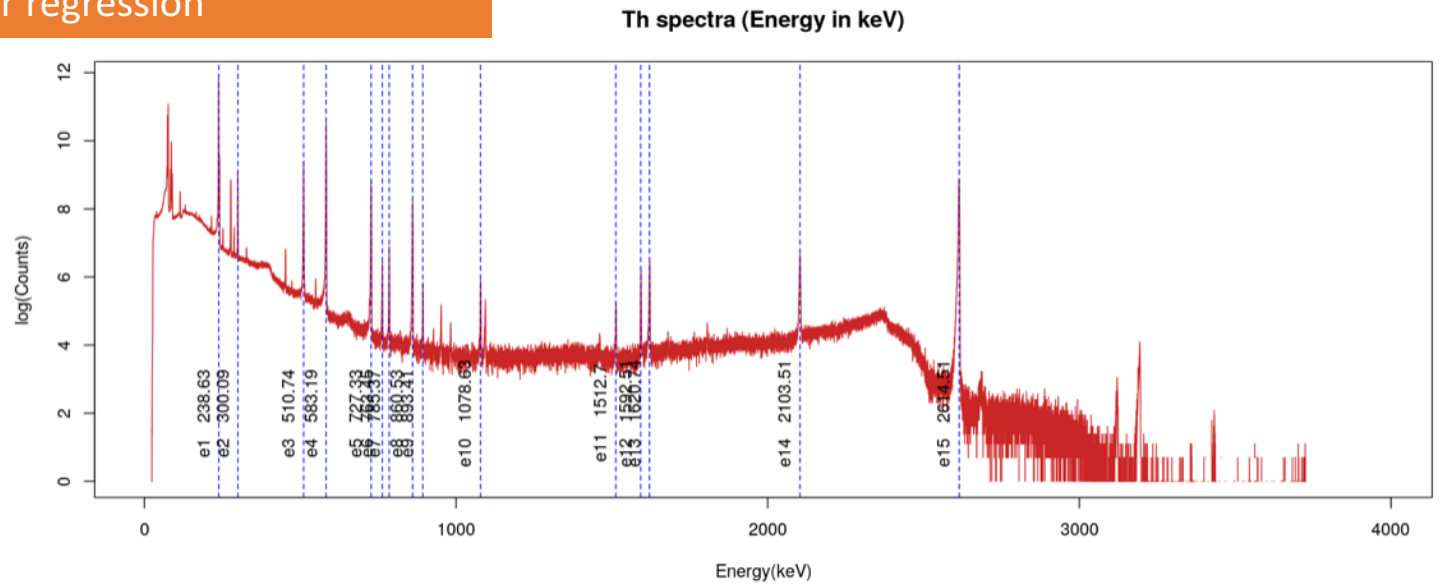
2nd Method - JAGS



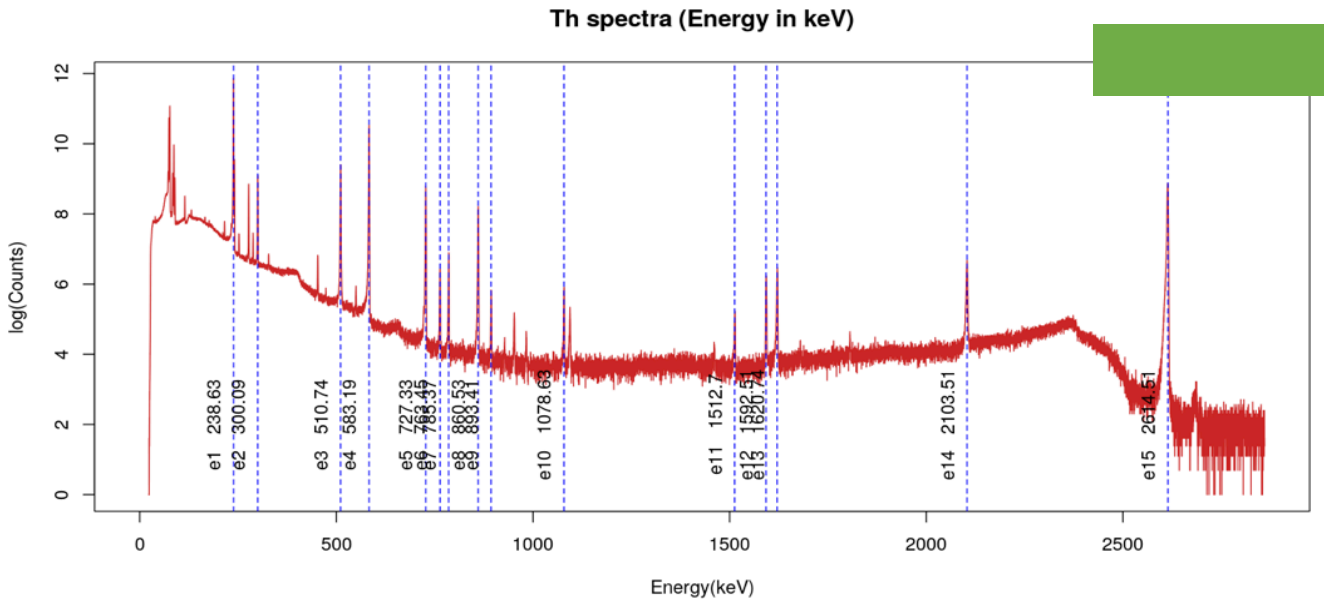
$$a \simeq 0.15$$
$$b \simeq -0.49 \pm 0.16$$

Calibration results

1st Method - Linear regression



2nd Method - JAGS



FWHM and inference of F and w_e

E	FWHM
<dbl>	<dbl>
238.6	0.9489
300.1	1.0399
510.7	1.4138
583.2	1.3928
727.3	1.5549
763.5	1.6096
785.4	1.5881
860.5	1.7291
893.4	1.6891
1078.6	1.7560
1512.7	2.2746
1592.5	1.9627
1620.7	2.2268
2103.5	3.6461
2614.5	3.1244

1st Method - Linear regression

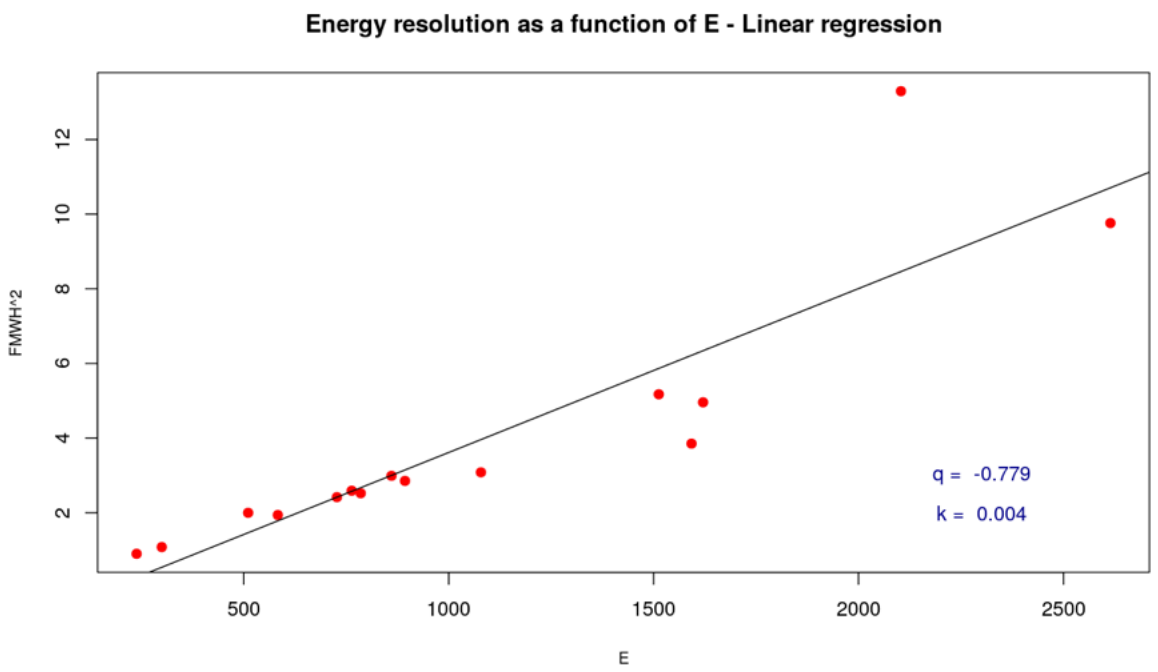
```
# lm() in R for linear regression
regression <- lm(formula =   df$FWHM~ df$E )
regression

# Extracting coefficients
a <- regression$coefficients[2]
b <- regression$coefficients[1]

fwhm <- a * data$x+ b
summary(regression)
```

2nd Method - JAGS

```
cat("model {

    # data likelihood
    for (i in 1:length(Y)){
        Y[i] ~ dnorm(mu[i],1/sigma);
        mu[i] <- (w_e^2+(2.355^2)*0.003*X[i]*f)^0.5;
    }

    #priors
    w_e ~ dunif(-3,3);
    f ~ dunif(0,1);
    sigma <- exp(logsigma)
    logsigma ~ dunif(-10,10)

}", file='model3fh.bug')

data <- NULL
data$X <- df$E
data$Y <- df$FWHM**0.5
data

init <- NULL
```

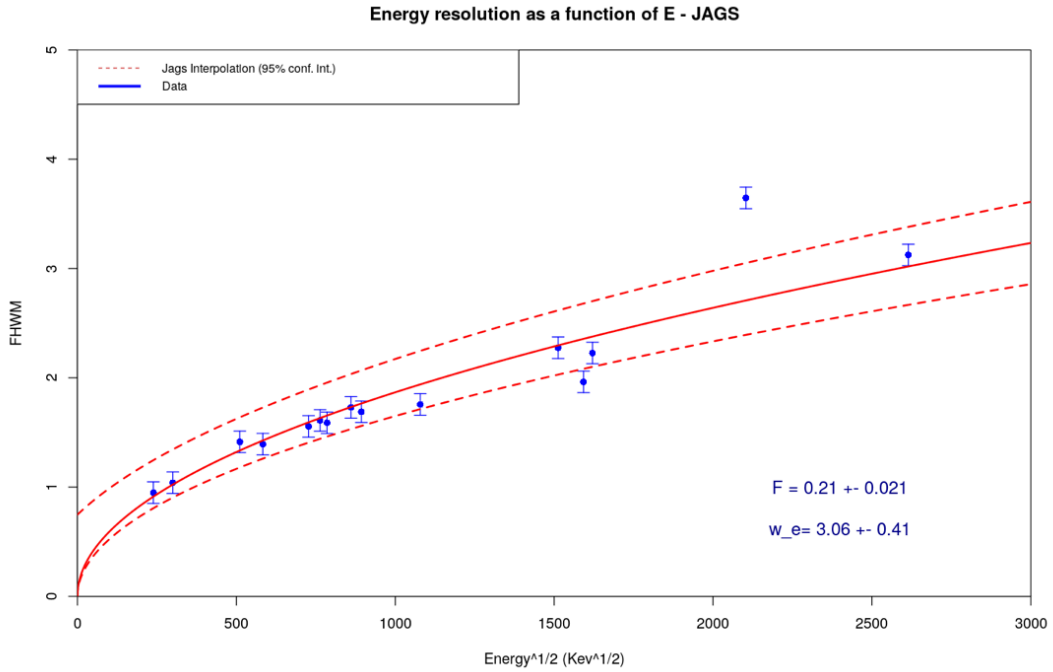
Inference of F and w_e: results

1st Method - Linear regression



$$F \simeq 0.24$$

2nd Method - JAGS



$$F \simeq 0.21 \pm 0.02$$
$$w_e \simeq 3.06 \pm 0.41$$

Thanks for the attention!