nCal - Some Examples

Youyi Fong
Fred Hutchinson Cancer Research Center

December 2, 2017

1 A basic example

The function *ncal* carries out nonlinear calibration, which entails fitting concentration-response curves to sets of samples of known concentrations, also known as standard samples, and using the fitted curves to obtain point estimates and confidence intervals for the analyte concentrations in the samples of interest, also known as the unknown samples. Take Luminex (Defawe et al., 2012) as an example. The assay is conducted on 96-well or 384-well plates. Each plate usually has a set of wells containing standard samples and a set of wells containing unknown samples. Within each well, multiple analytes can be assayed simultaneously. In the following, we will use *plate* and *assay* exchangeably.

ncal has 2 required arguments, *formula* and *data*, and 24 optional arguments. *data* is a data frame object, where each row is a sample, either standard or unknown. The *formula* object specifies the names of the response and concentration columns. Besides these two columns, *data* is also expected to have two columns that help identify standard curves: *analyte* and *assay_id*. If *data* contains both standard and unknown samples, two more columns are also required: *well_role* and *sample_id*. *data* may also have additional columns. For example, it is sometimes convenient to include dilution factor for unknown samples. We now simulate a dataset with one assay, one analyte and four unknown samples with varying dilutions.

```
set.seed(1)
log.conc=log(1e4)-log(3)*9:0
n.replicate=2
fi=simulate1curve (p.eotaxin[1,], rep(log.conc,each=n.replicate), sd.e=0.2)
dat.std=data.frame(fi, expected_conc=exp(rep(log.conc,each=n.replicate)),

 analyte="Test", assay_id="Run 1", sample_id=NA, well_role="Standard",
 dilution=rep(3**(9:0), each=n.replicate))
# add unknown
dat.unk=rbind(
data.frame(fi=exp(6.75), expected_conc=NA, analyte="Test", assay_id="Run 1",
sample_id=1, well_role="Unknown", dilution=1)
```

```
, data.frame(fi=exp(6.70), expected_conc=NA, analyte="Test", assay_id="Run
1", sample_id=2, well_role="Unknown", dilution=1)
, data.frame(fi=exp(3), expected_conc=NA, analyte="Test", assay_id="Run 1",
sample_id=3, well_role="Unknown", dilution=1)
, data.frame(fi=exp(4.4), expected_conc=NA, analyte="Test", assay_id="Run
1", sample_id=4, well_role="Unknown", dilution=10)
)
dat=rbind(dat.std, dat.unk)
```

ncal provides access to two curve-fitting engines, *drm* and *bcrm*. To use *drm* to fit curves separately for each assay, simply call *ncal* with mostly default parameters.

```
res.drm = ncal(log(fi)~expected_conc, dat, return.fits = TRUE)
```

ncal returns a data frame, each row of which corresponds to one unknown sample.

```
res.drm
## fi expected_conc analyte assay_id sample_id well_role dilution est.log.conc
se
## 1 854.06 NA Test Run 1 1 Unknown 1 3.940 0.1623
## 2 812.41 NA Test Run 1 2 Unknown 1 3.902 0.1628
## 3 20.09 NA Test Run 1 3 Unknown 1 -1.370 Inf
## 4 81.45 NA Test Run 1 4 Unknown 10 1.815 9.7521
## est.conc lb.conc ub.conc
## 1 51.419 3.639e+01 7.266e+01
## 2 49.494 3.498e+01 7.002e+01
## 3 0.254 0.000e+00 Inf
## 4 6.142 5.767e-09 6.541e+09
```

The curve fit object is not returned by default, but can be returned as an attribute of the return data frame when return.fits is set to TRUE.

```
fit.drm=attr(res.drm, "fits")[[1]]
```

ncal also creates a four-panel plot (Figure 1) for each assay by default. The upper left panel shows the standard samples data and the fitted curve. The upper right panel is similar to the upper left panel, but adds a set of points representing the samples of interest. The lower left panel shows the estimated variance of the estimated concentrations as functions of the estimated concentration (Fong et al., 2013; Cumberland et al., 2015). The lower right panel shows the coefficient of variation (CV) as a function of the estimated concentration. The limits of quantification (LOQ) are defined as the estimated concentrations at which the percent CV equals 20.

To use *bcrm* to fit a robust Bayesian hierarchical model (Fong et al., 2012), simply pass *bcrm.fit=TRUE* to *ncal*. (Proper installation of JAGS and the R package rjags are required.)

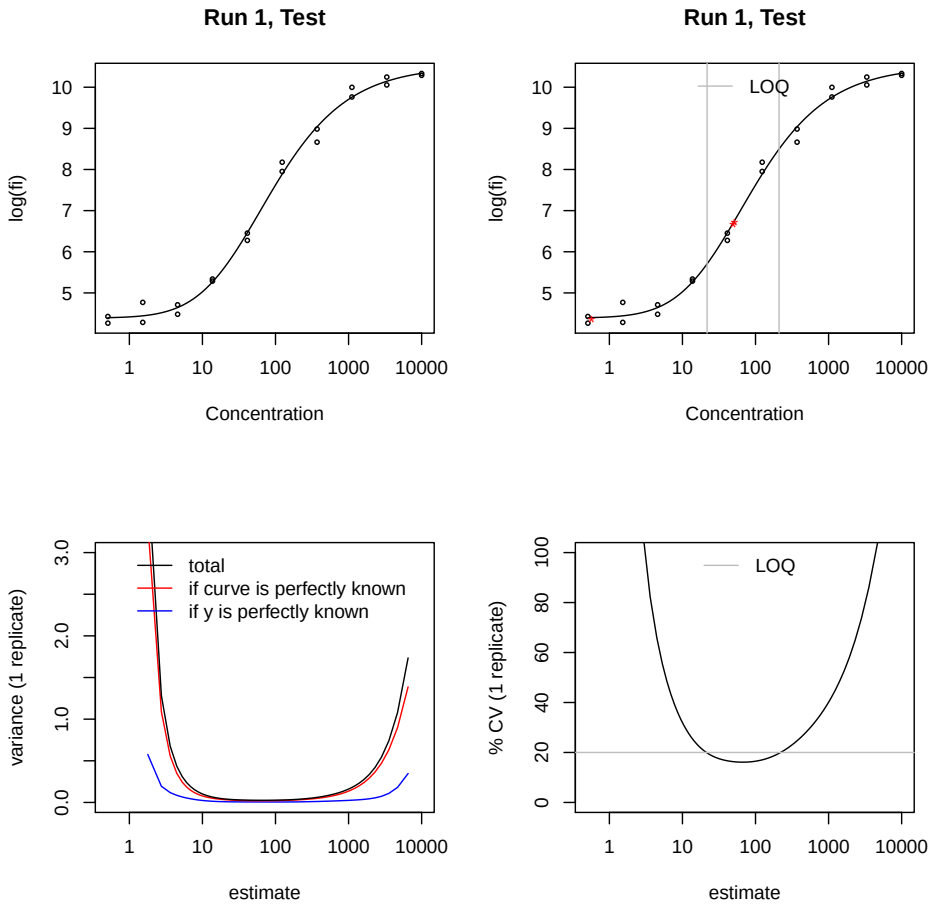

Figure 1: ncal graphical output, drm fit.

```
res.bcrm = ncal(log(fi)~expected_conc, dat, bcrm.fit=T, return.fits = TRUE,
bcrm.model="norm", control.jags=list(n.iter=5e3))
fit.bcrm=attr(res.bcrm, "fits")
```

The behavior of *ncal* is the same as when *drm* is used as the curve-fitting engine. For this example, the two curve fitting methods produce similar results. This can seen by comparing the two graphical outputs in Figure 1 and Figure 2, or by comparing the estimate concentrations and the associated uncertainty of the unknown samples, or by comparing the estimated parameters of the five-parameter logistic curves and the associated uncertainty.

```
rbind(cla2gh(coef(fit.drm)), coef(fit.bcrm))
rbind(sqrt(diag(vcov(fit.drm))), sqrt(diag(vcov(fit.bcrm, type="classical"))))
```

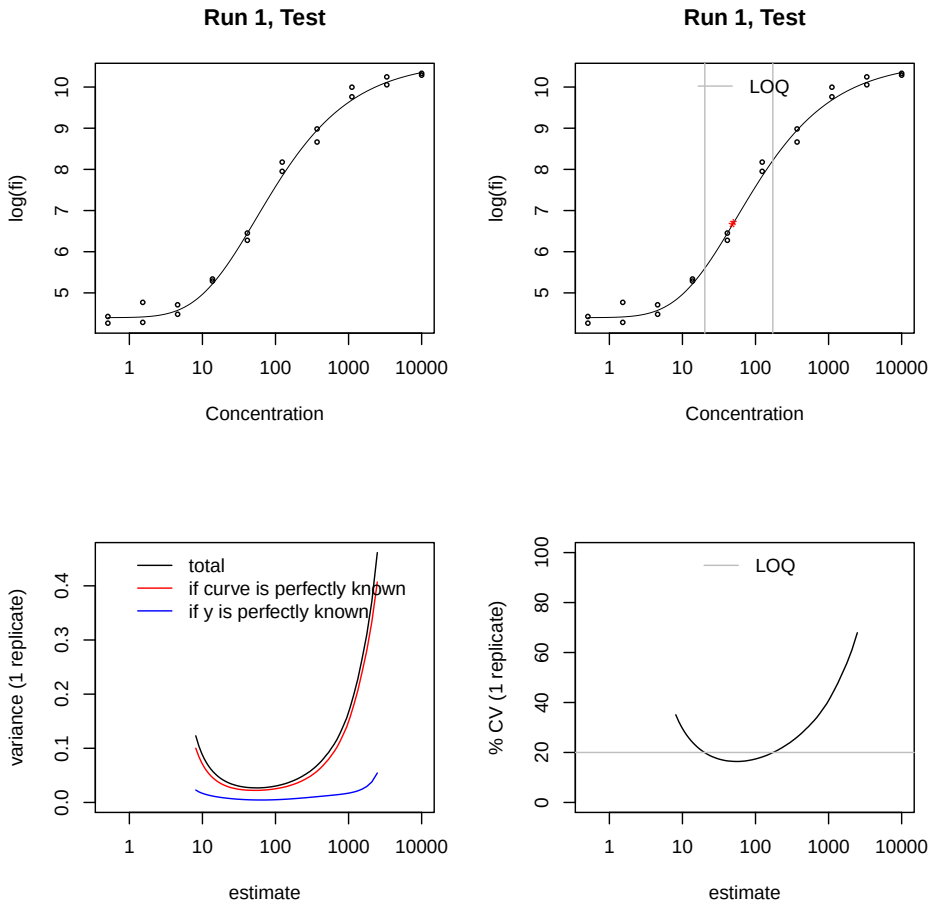

Figure 2: ncal graphical output, bcrm fit.

Given a fitted curve, the analyte concentrations of new unknown samples can be estimated via *getConc* by passing it the fit object and the observed responses.

```
getConc(fit.bcrm, c(5.7,6.3))
```

2 Borrowing information across multiple assays

In a real application, multiple plates often need to be run in a stretch of days or weeks because all unknown samples can not fit in a single assay. In some cases, it is worth borrowing information across standard curves for the same analyte. We now use a real dataset to illustrate this. The dataset, hier.model.ex.2, is part of the *ncal* package. In Fong et al. (2013), the first four assays

are used as illustration due to space limit. To be consistent, we use four assays here as well. We first fit a Bayesian robust random effects model (Fong et al., 2012) for the four assays. We choose the *error.model* to be *gh_t4*, which means the g-h parameterization is used (Richards, 1959; Fong et al., 2012) and the Student's t distribution with 4 degrees of freedom is assumed as the error distribution. Ideally we would like to collect 10^5posterior samples, but to reduce the time it takes to compile this vignette, we only run 10^4iterations. The printed message about residuals hints at an outliers problem.

```
dat=subset(hier.model.ex.2, assay_id %in% paste("Run",1:4))
fit.bcrm=bcrm(log(fi)~expected_conc, dat, error.model="t4", prior="cytokine",
n.iter=1e4)
```

We will extract the curve fit for each assay from *fit.bcrm* using *get.single.fit* and plot it together with the robust *drm* fit. In addition, we will plot curves fitted using one of the popular commercial softwares, Graphpad Prism, with robust option.

```
# parameters from Prism fits
prism.1 = c("c"=1.596,"d"=10.28,"f"=0.7202,"b"=-0.8815,"e"=10^((1.597+
      1/0.8815*log10(2**(1/0.7202)-1))) )
prism.2 = c("c"=1.350,"d"=11.32,"f"=8.640e+010,"b"=-0.3452,"e"=10^((1.485+
      1/0.3452*log10(2**(1/8.640e+010)-1))) )
prism.3 = c("c"=1.333,"d"=10.23,"f"=0.7366,"b"=-0.8502,"e"=10^((1.526+
      1/0.8502*log10(2**(1/0.7366)-1))) )
prism.4 = c("c"=1.580,"d"=10.37,"f"=1.694,"b"=-0.6639,"e"=10^((1.530+
      1/0.6639*log10(2**(1/1.694)-1))) )
prism=rbind(prism.1,prism.2,prism.3,prism.4)
# start plotting
par(mfrow=c(2,2))
for (i in 1:4) {
 assay.id=paste("Run", i)
 # fit drm model
 fit.drm = drm.fit(log(fi)~expected_conc, data=dat[dat$assay_id==assay.id,],
robust="median")
 plot(fit.drm, type="all", col="black", main=assay.id, lty=2)
 plot(get.single.fit(fit.bcrm, assay.id), add=T, log="x", col=1)
 # plot Prism fit
 xx=exp(seq(log(0.51),log(1e4),length=100))
 lines(xx, FivePL.x(xx,prism[i,]), type="l", lty=1, col="darkgray")
 legend(x="bottomright",legend=c("Prism, robust","drm, median","bcrm, t4"),lty=c(1,2,1),
```

```
        col=c("darkgray",1,1),bty="n")
}
```

Figure 2 shows that in Run 2, which has multiple outliers, *bcrm* produces a different fit from *drm* and Prism. This suggests by borrowing information judiciously we can overcome the challenging issue of masking in the outliers problem.

3 Robust Bayesian hierarchical model priors

The priors of the robust Bayesian hierarchical model used in *bcrm* are specified as follows. Denote $\boldsymbol{\theta} = \{c, d, g, \log(h), \log(f)\}$(g-h parameterization Richards, 1959; Fong et al., 2012). Let $\bar{\boldsymbol{\theta}}$be the average of the least square fit parameters across assay runs.

$$\begin{aligned}\boldsymbol{\theta}_0 &\sim N\left(\bar{\boldsymbol{\theta}},\ \text{precision}=diag\left(0.40, 6.09, 1.75, 16.96, 1.08\right)\right)\\ \Omega &\sim Wishart_5\left(r=8, S=diag\left(1.83, 9.37, 23, 172, 2.01\right)\right)\end{aligned}$$

For $\varepsilon_{ik} \sim t_4\left(0, \sigma^2\right)$, we assume prior $\sigma^{-2} \sim Gamma\left(\text{shape}=2, \text{rate}=0.02\right)$. For $\varepsilon_{ik} \sim N\left(0, \sigma^2\right)$, we assume prior $\sigma^{-2} \sim Gamma\left(\text{shape}=2, \text{rate}=0.06\right)$.

References

Cumberland, W., Fong, Y., Yu, X., Defawe, O., Frahm, N. and De Rosa, S. (2015), "Nonlinear Calibration Model Choice between the Four and Five Parameter Logistic Models," *Journal of Biopharmaceutical Statistics*, 25, 972–983.

Defawe, O., Fong, Y., Vasilyeva, E., Pickett, M., Carter, D., Gabriel, E. et al (2012), "Optimization and qualification of a multiplex bead array to assess cytokine and chemokine production by vaccine-specific cells," *Journal of Immunological Methods*, 382, 117–128.

Fong, Y., Wakefield, J., De Rosa, S. and Frahm, N. (2012), "A Robust Bayesian Random Effects Model for Nonlinear Calibration Problems," *Biometrics*, 68, 1103–1112.

Fong, Y., Sebestyen, K., Yu, X., Gilbert, P. and Self, S. (2013), "nCal: a R package for nonlinear calibration," *Bioinformatics*, 29, 2653–2654.

Richards, F. (1959), "A flexible growth function for empirical use," *Journal of Experimental Botany*, 10, 290–300.

4 Appendix A: Frequently asked questions

```
Q: How do I install nCal?
A: Please follow the following steps. Step 4 is necessary for running nCal GUI.
1.  Install R. R can be downloaded from http://cran.r-project.org/
```

2. Run R. On Windows, this can be done by double-clicking on the R icon.
3. Install the nCal package. At the R console, type
> install.packages ("nCal")
and press enter.
4. Install the gWidgetstcltk package. At the R console, type
> install.packages ("gWidgetstcltk")
and press enter.

Q: How do I launch the nCal GUI?
A: The nCal GUI is launched from within R. Please follow the following steps.
1. Run R. On Windows, this can be done by by double-clicking on the R icon.
2. At the R console, load the nCal library by
> library (nCal)
3. run nCal GUI by
> ncalGUI()

Q: What other programs does nCal need to be able to import Luminex raw output?
A: To import data from raw Luminex files, PERL and the R package gdata are needed.
PERL can be downloaded from http://www.activestate.com/activeperl/downloads
To install the gdata package, enter the following command in R.
> install.packages("gdata")

Q: What other programs does nCal need to be able to fit the robust Bayesian hierarchial model?
A: To fit Bayesian robust hierarchical model, JAGS and the R package rjags are needed.
First, download and install JAGS from http://sourceforge.net/projects/mcmc-jags/files/JAGS/
Second, in R enter the following command to install the rjags package.
> install.packages("rjags")

Q: Help on GUI
A: The GUI (screenshot: http://research.fhcrc.org/content/dam/stripe/youyiFong/files/gui.pdf)
is divided into four sections.
Input
* You can select a file containing the input data using the browse button.
The nCal package comes with an example Luminex raw output xls file, 02-14A22-IgA-Biotin-tiny,
and it is located at the R_installation_directory/library/nCal/misc folder.
You can use this file to test the GUI.
* The input file can be in one of three different formats.
* The Response variable field allows users to enter the name of the response variable,
and it defaults to log(fi).
* The Concentration/dose variable field allows users to enter the name of the
concentration or dose variable, and it defaults to expected_conc.
Output
* The Estimated concentrations field allows users to enter the name of the file
to save the estimated concentrations, and it defaults to estimated_concentrations.csv.
The browse button next to it allows users to choose a folder to save the file in.
* The Calibration plots field allows users to enter the name of the file to

save the calibration plots, and it defaults to Calibration_plots.pdf.
The browse button next to it allows users to choose a folder to save the file in.
Curve fitting method
* Users can choose between two curve fitting methods.
The last section allows users to use the last fitted curve to estimate concentrations for any responses entered by users.
* More than one response can be separated by space or comma, e.g. 5.1, 6.3
* The Apply button becomes activated when there has been a fitted curve.
When the Apply button is clicked, a new window pops up. In the window is a table, each row of which corresponds to one entered response.

Q: What is the relationship between nCal and Ruminex?
A: Ruminex is the precursor of nCal. To switch to nCal, the only necessary change is to replace Ruminex with nCal in the library() command because rumi is also provided in nCal and can be called in the same way as it was in Ruminex.