nCal - Some Examples

Youyi Fong
Fred Hutchinson Cancer Research Center

July 12, 2013

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(10000) - 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(6.7), 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). 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., 2013), simply pass *bcrm.fit=TRUE* to *ncal*. (Proper installation of JAGS and the R package rjags are required.)

```
res.bcrm <- ncal(log(fi) ~ expected_conc, dat, bcrm.fit = T,
    return.fits = TRUE, bcrm.model = "norm", control.jags = list(n.iter = 5000))
```

```
## Loading required package:  rjags
## Loading required package:  coda
## Linked to JAGS 3.3.0
## Loaded modules:  basemod,bugs
```

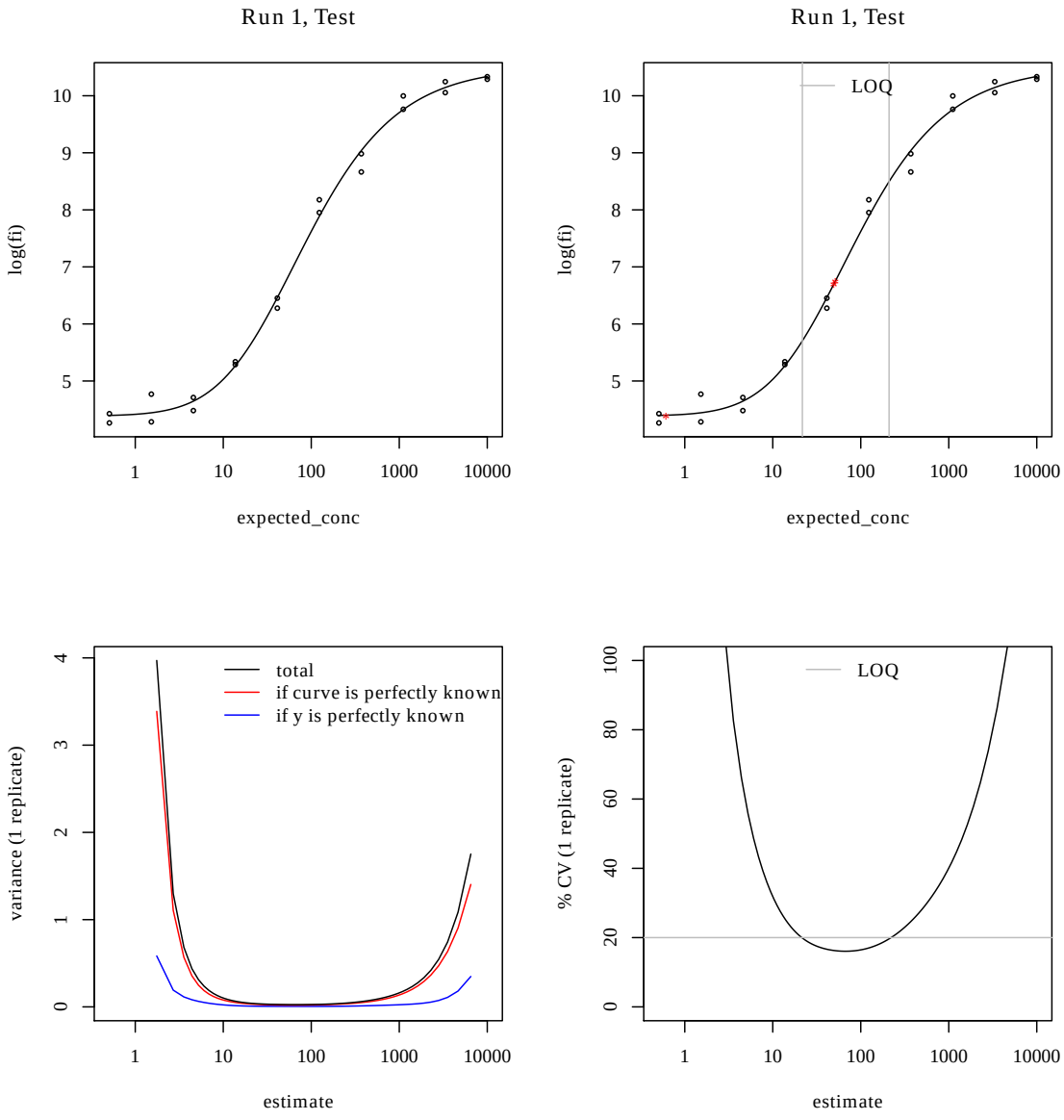

Figure 1: ncal graphical output, drm fit.

```
##   Running jags

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,

```
res.bcrm

##       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        3.935 0.1634
## 2 812.41            NA    Test    Run 1         2   Unknown        1        3.896 0.1642
## 3  20.09            NA    Test    Run 1         3   Unknown        1       -1.370    Inf
## 4  81.45            NA    Test    Run 1         4   Unknown       10        2.223    NaN
##   est.conc lb.conc ub.conc
## 1   51.137   36.10   72.44
## 2   49.213   34.68   69.84
## 3    0.254    0.00     Inf
## 4    9.232     NaN     NaN
```

or by comparing the estimated parameters of the five-parameter logistic curves and the associated uncertainty.

```
rbind(cla2gh(coef(fit.drm)), coef(fit.bcrm))

##          c     d     g     h     f
## [1,] 4.386 10.51 4.146 1.322 2.547
## [2,] 4.345 10.50 4.176 1.305 2.045

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

##           b      c      d     e     f
## [1,] 0.1368 0.1261 0.1882 33.02 2.707
## [2,] 0.1314 0.1582 0.1446 39.00 2.767
```

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))

##      log.conc   s.e. concentration lower.bound upper.bound      s1       s2  se.x
## [1,]    3.052 0.2046         21.15       17.11       25.58 0.03194 0.009911 4.327
## [2,]    3.579 0.1741         35.83       30.14       42.41 0.02284 0.007479 6.240
```

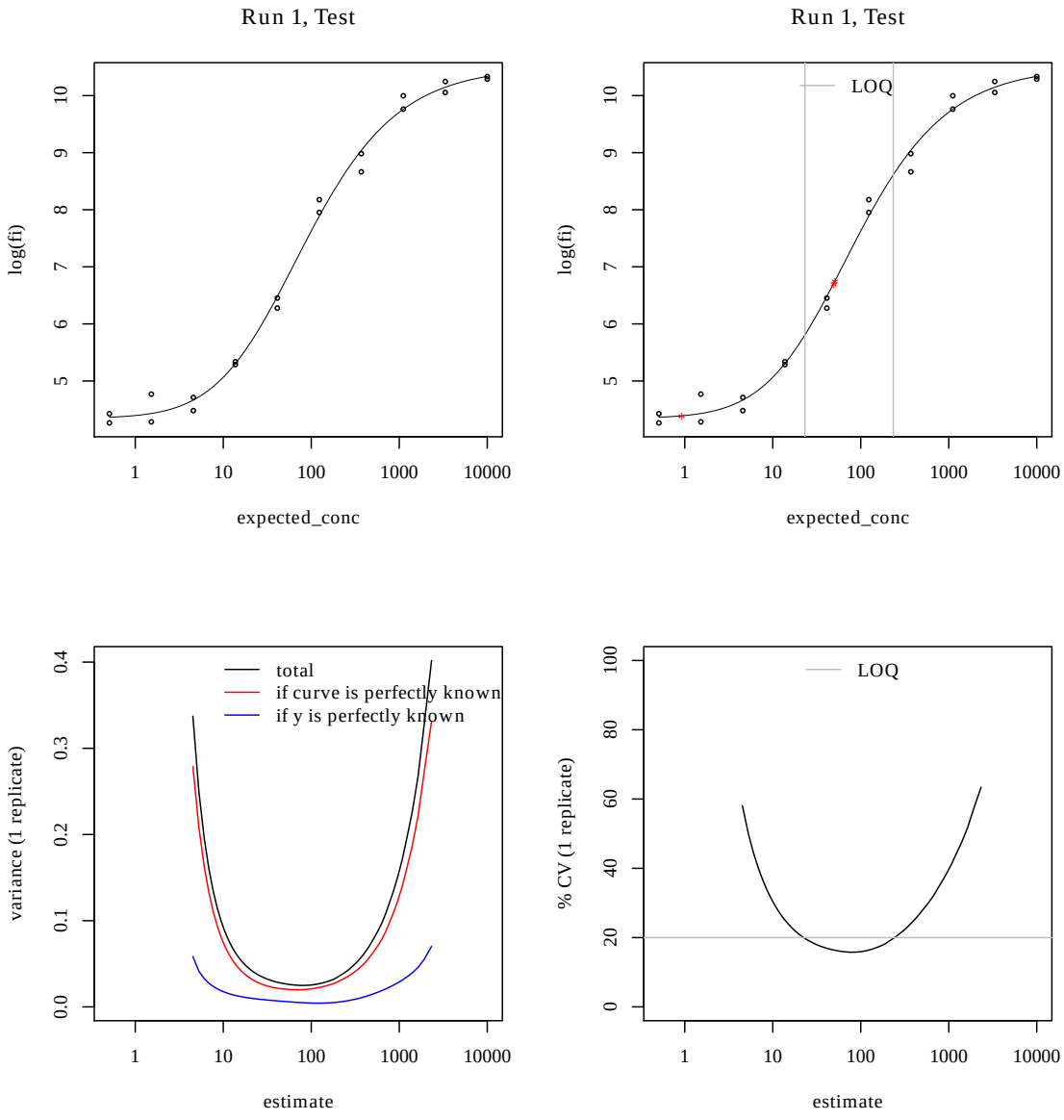

Figure 2: ncal graphical output, bcrm fit.

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 10^5 posterior samples, but to reduce the time it takes to compile this vignette, we only run 10^4 iterations. 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 = "gh_t4",
    informative.prior = T, n.iter = 10000)

## [1] drm.fit: residuals larger than ususal in Run 2, MCP-1
##   Running jags
```

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 = 1.35, d = 11.32, f = 8.64e+10, b = -0.3452,
    e = 10^((1.485 + 1/0.3452 * log10(2^(1/8.64e+10) - 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 = 1.58, d = 10.37, f = 1.694, b = -0.6639, e = 10^((1.53 +
    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(10000), 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")
}

## [1] drm.fit: residuals larger than ususal in Run 2, MCP-1
## [1] drm.fit: residuals larger than ususal in Run 3, MCP-1
```

Figure 3 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 5PL 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

Defawe, O., Fong, Y., Vasilyeva, E., Pickett, M., Carter, D., Gabriel, E., Rerks-Ngarm, S., Nitayaphan, S., Frahm, N., McElrath, M., 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(1):117–128.

Fong, Y., Sebestyen, K., Yu, X., Gilbert, P., and Self, S. (2013). ncal: a r package for nonlinear calibration. *Bioinformatics*, submitted.

Fong, Y., Wakefield, J., De Rosa, S., and Frahm, N. (2012). A Robust Bayesian Random Effects Model for Nonlinear Calibration Problems. *Biometrics*, 68(4):1103–1112.

Richards, F. (1959). A flexible growth function for empirical use. *Journal of Experimental Botany*, 10(2):290–300.

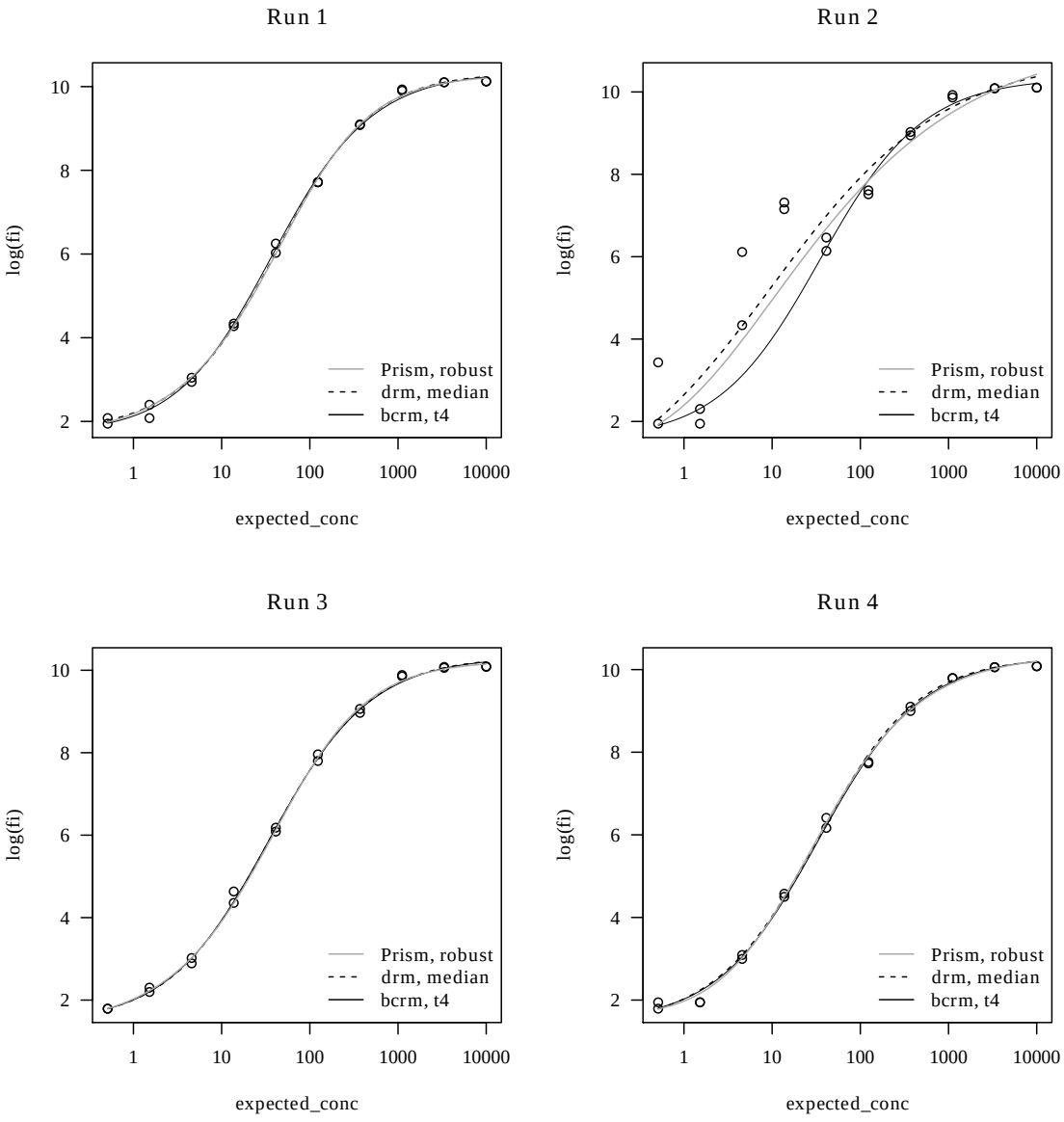

Figure 3: Comparing bcrm fit with drm and Prism fits.