ggPMX with nlmixr

```r
library(ggPMX)
library(nlmixr)
```

It is simple to create a ggPMX controller for a nlmixr object.

Using the theophylline example with a nlmixr model we have:

```r
one.compartment <- function() {
    ini({
        tka <- 0.45 # Log Ka
        tcl <- 1 # Log Cl
        tv <- 3.45    # Log V
        eta.ka ~ 0.6
        eta.cl ~ 0.3
        eta.v ~ 0.1
        add.sd <- 0.7
    })
    model({
        ka <- exp(tka + eta.ka)
        cl <- exp(tcl + eta.cl)
        v <- exp(tv + eta.v)
        d/dt(depot) = -ka * depot
        d/dt(center) = ka * depot - cl / v * center
        cp = center / v
        cp ~ add(add.sd)
    })
}
nlmixr(one.compartment)
#> __ RxODE-based ODE model _______________________________________________________
#> -- Initialization: -------------------------------------------------------------
#> Fixed Effects ($theta):
#>  tka  tcl   tv
#> 0.45 1.00 3.45
#>
#> Omega ($omega):
#>        eta.ka eta.cl eta.v
#> eta.ka    0.6    0.0   0.0
#> eta.cl    0.0    0.3   0.0
#> eta.v     0.0    0.0   0.1
#> -- mu-referencing ($muRefTable): -----------------------------------------------
#>   theta    eta
#> 1   tka eta.ka
#> 2   tcl eta.cl
#> 3    tv  eta.v
#>
#> -- Model: ----------------------------------------------------------------------
#>         ka <- exp(tka + eta.ka)
#>         cl <- exp(tcl + eta.cl)
#>         v <- exp(tv + eta.v)
#>         d/dt(depot) = -ka * depot
#>         d/dt(center) = ka * depot - cl / v * center
#>         cp = center / v
#>         cp ~ add(add.sd)
#> ────────────────────────────────────────────────────────────────────────────────
```

At the time of this writing, nlmixr requires python; Since cran does not have python installed, we run this model locally by:

```
fit <- nlmixr(one.compartment, theo_sd, est="saem", control=list(print=0))
saveRDS(fit,"fit.rds")
```

Then we can read this into the system

```
fit <- readRDS("fit.rds")
capture.output(print(fit)) %>%
sapply(crayon::strip_style) %>%
unname %>%
print
#>  [1] "-- nlmixr SAEM(ODE); OBJF not calculated fit ----------------------------------- "
#>  [2] " Gaussian/Laplacian Likelihoods: AIC() or $objf etc. "
#>  [3] " FOCEi CWRES & Likelihoods: addCwres() "
#>  [4] ""
#>  [5] "-- Time (sec; $time): ---------------------------------------------------------- "
#>  [6] "          saem    setup table covariance    other"
#>  [7] "elapsed 20.803 3.921085 0.009      0.007 0.150915"
#>  [8] ""
#>  [9] "-- Population Parameters ($parFixed or $parFixedDf): --------------------------- "
#> [10] "       Parameter  Est.     SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%"
#> [11] "tka       Log Ka 0.451  0.196 43.5       1.57 (1.07, 2.31)     71.9     0.411% "
#> [12] "tcl       Log Cl  1.02 0.0836 8.22       2.77 (2.35, 3.26)     27.0      3.36% "
#> [13] "tv         Log V  3.45 0.0469 1.36       31.5 (28.7, 34.5)     14.0      10.0% "
#> [14] "add.sd           0.692                               0.692                     "
#> [15] " "
#> [16] ""
#> [17] "  Covariance Type ($covMethod): linFim"
#> [18] "  No correlations in between subject variability (BSV) matrix"
#> [19] "  Full BSV covariance ($omega) or correlation ($omegaR; diagonals=SDs) "
#> [20] "  Distribution stats (mean/skewness/kurtosis/p-value) available in $shrink "
#> [21] ""
#> [22] "-- Fit Data (object is a modified tibble): ------------------------------------- "
#> [23] "# A tibble: 132 x 18"
#> [24] "  ID     TIME    DV  EVID  PRED    RES IPRED   IRES  IWRES eta.ka eta.cl   eta.v"
#> [25] "  <fct> <dbl> <dbl> <int> <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>   <dbl>"
#> [26] "1 1     0      0.74     0  0     0.74   0     0.74   1.07   0.105 -0.487 -0.0800"
#> [27] "2 1     0.25   2.84     0  3.26 -0.423  3.86 -1.02  -1.48   0.105 -0.487 -0.0800"
#> [28] "3 1     0.570  6.57     0  5.84  0.725  6.81 -0.235 -0.340  0.105 -0.487 -0.0800"
#> [29] "# ... with 129 more rows, and 6 more variables: ka <dbl>, cl <dbl>, v <dbl>,"
#> [30] "#   cp <dbl>, depot <dbl>, center <dbl>"
```

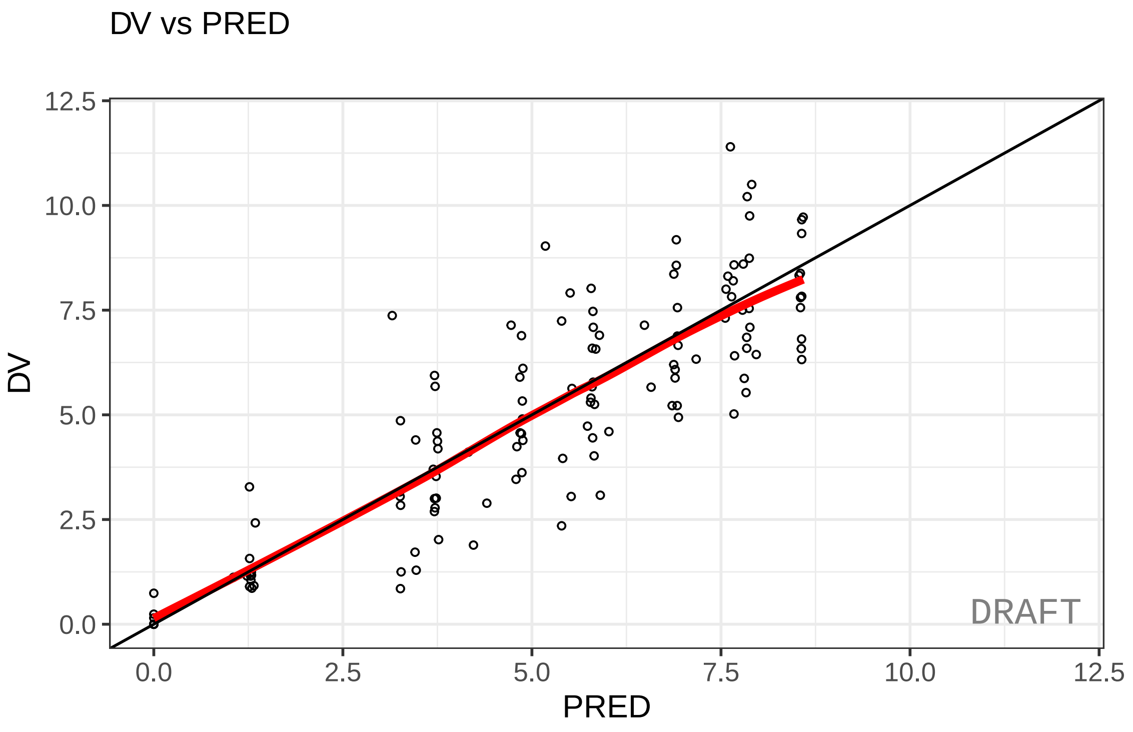
The `fit` object is a nlmixr fit; You can read it into the nlmixr controller by:

```
fit %>%
    pmx_nlmixr(vpc = FALSE) -> ## VPC is turned on by default, can turn off.
    ctr ## Assigned to controller
```

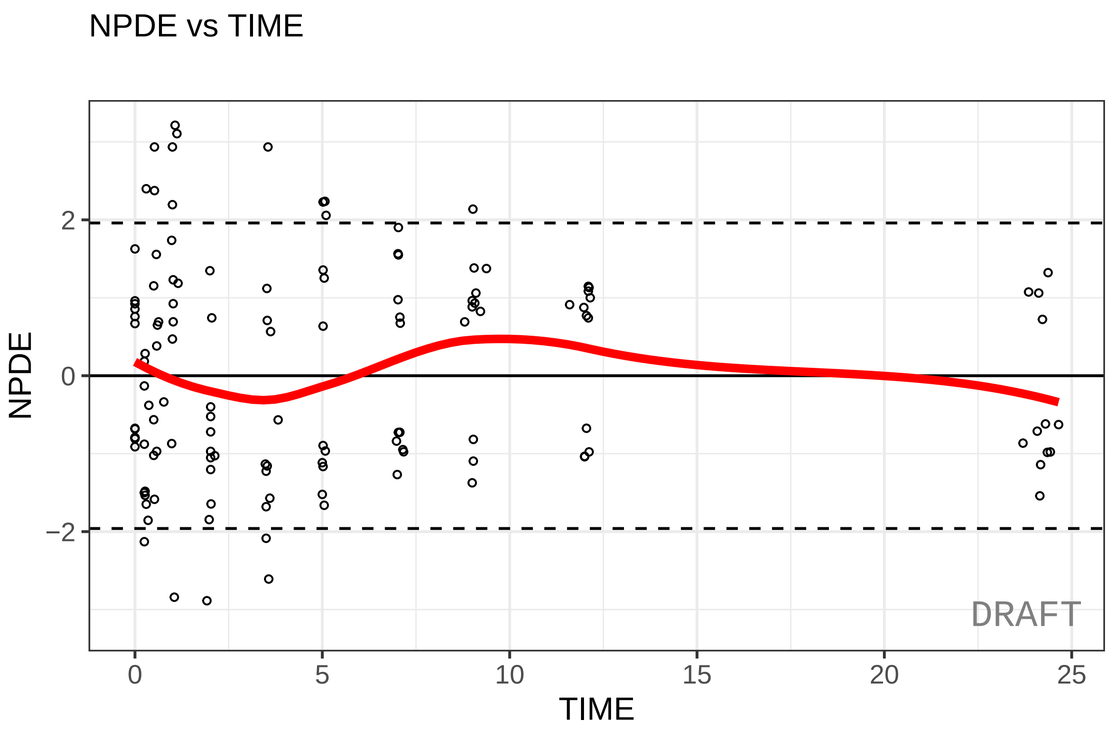
```
#> Compiling NPDE model...done
```

Once the controller is created, you can of course have the same types of diagnostic plots as the standard ggPMX package; Using the same examples as the user manual:

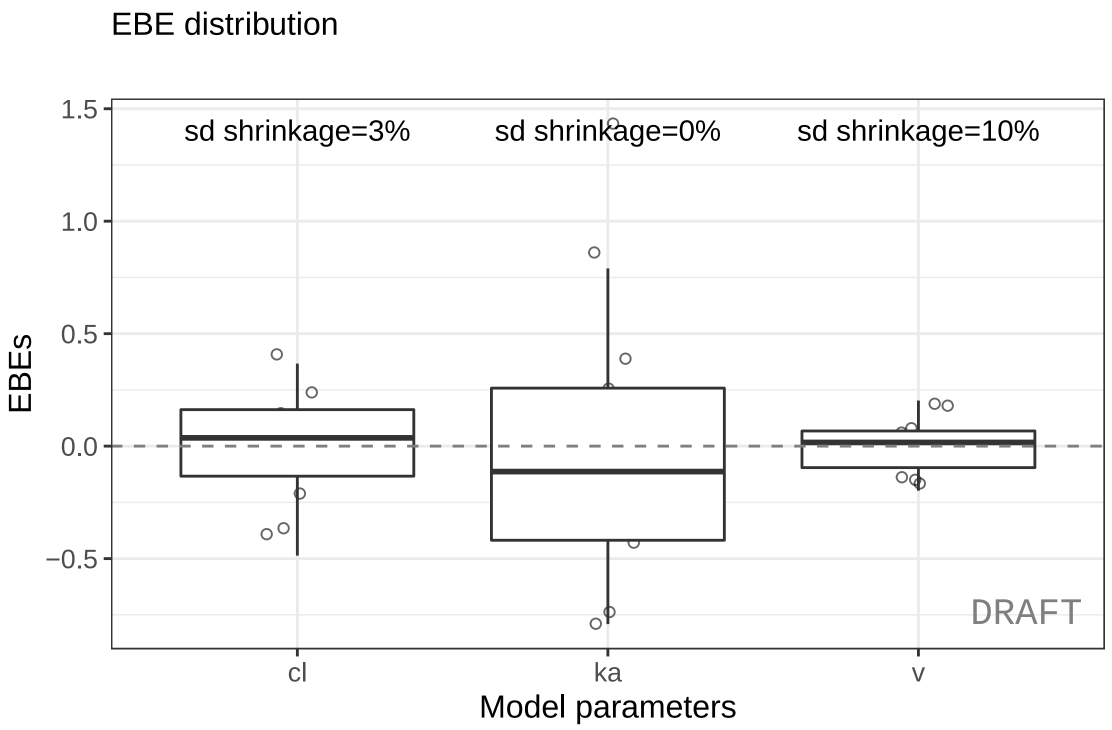
```
ctr %>% pmx_plot_dv_pred
#> `geom_smooth()` using formula 'y ~ x'
```



```
ctr %>% pmx_plot_npde_time
#> `geom_smooth()` using formula 'y ~ x'
```

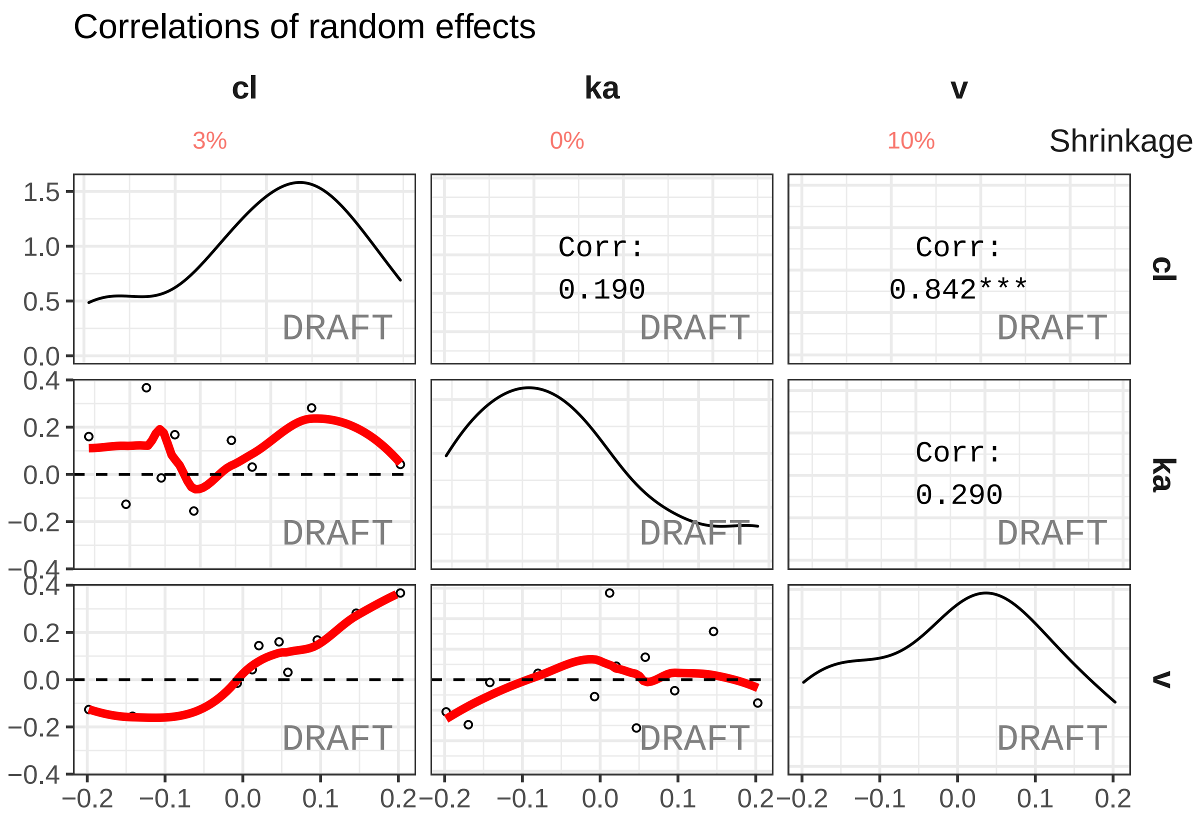


```
ctr %>% pmx_plot_vpc
#> NULL
ctr %>% pmx_plot_eta_box
```



```
ctr %>% pmx_plot_eta_matrix(
  shrink=list(size=3,hjust=1.5))
#> `geom_smooth()` using formula 'y ~ x'
#> Warning: Removed 2 rows containing non-finite values (stat_smooth).
#> Warning: Removed 2 rows containing missing values (geom_point).
#> `geom_smooth()` using formula 'y ~ x'
#> Warning: Removed 2 rows containing non-finite values (stat_smooth).

#> Warning: Removed 2 rows containing missing values (geom_point).
#> Warning: Removed 1 rows containing missing values (geom_smooth).
#> `geom_smooth()` using formula 'y ~ x'
```



This shows the standard plots for theophylline