Refit with Saved Parameters

Overview

This vignette walks through using a text file of previously fit model parameters to use in the `bdotsRefit` function. This is convenient if you have already gone through the refitting process and would like to save/load the refitted parameters in a new session.

To demonstate this process, we start with fitting a set of curves to our data

```
library(bdots)

fit <- bdotsFit(data = cohort_unrelated,
                subject = "Subject",
                time = "Time",
                y = "Fixations",
                group = c("Group", "LookType"),
                curveType = doubleGauss(concave = TRUE),
                cor = TRUE,
                numRefits = 2,
                cores = 2,
                verbose = FALSE)

refit <- bdotsRefit(fit, quickRefit = TRUE, fitCode = 5)
```

From this, we can create an appropriate `data.table` that can be used in a later session

```
parDT <- coefWriteout(refit)
head(parDT)
#>    Subject Group LookType       mu        ht      sig1     sig2        base1
#> 1:       1    50   Cohort 429.7595 0.1985978  159.8869 314.6389  0.009709772
#> 2:       1    65   Cohort 634.9292 0.2635044  303.8081 215.3845 -0.020636092
#> 3:       2    50   Cohort 647.0655 0.2543769  518.9632 255.9871 -0.213087597
#> 4:       2    65   Cohort 723.0551 0.2582110  392.9509 252.9381 -0.054827082
#> 5:       3    50   Cohort 501.4843 0.2247729  500.8605 158.4164 -0.331698893
#> 6:       3    65   Cohort 485.5232 0.3111034 1268.8384 115.2930 -4.072126219
#>         base2
#> 1: 0.03376106
#> 2: 0.02892360
#> 3: 0.01368195
#> 4: 0.03197292
#> 5: 0.02522686
#> 6: 0.04518018
```

It's important that columns are included that match the unique identifying columns in our `bdotsObj`, and that the parameters match the coefficients used from `bdotsFit`

```
## Subject, Group, and LookType
head(refit)
#>    Subject Group LookType        fit        R2  AR1 fitCode
#> 1:       1    50   Cohort <gnls[18]> 0.9697202 TRUE       0
#> 2:       1    65   Cohort <gnls[18]> 0.9804901 TRUE       0
#> 3:       2    50   Cohort <gnls[18]> 0.9811708 TRUE       0
#> 4:       2    65   Cohort <gnls[18]> 0.9697466 TRUE       0
#> 5:       3    50   Cohort <gnls[18]> 0.9761906 TRUE       0
#> 6:       3    65   Cohort <gnls[18]> 0.9448814 TRUE       1
```

```
## doubleGauss pars
colnames(coef(refit))
#> [1] "mu"    "ht"    "sig1"  "sig2"  "base1" "base2"
```

We can save our parameter `data.table` for later use, or read in any other appropriately formatted `data.frame`

```
## Save this for later using data.table::fwrite
fwrite(parDT, file = "mypars.csv")
parDT <- fread("mypars.csv")
```

Once we have this, we can pass it as an argument to the `bdotsRefit` function. Doing so will ignore the remaining arguments

```
new_refit <- bdotsRefit(refit, paramDT = parDT)
```

We end up with a `bdotsObj` that matches what we had previously. As seeds have not yet been implemented, the resulting parameters may not be exact. It will, however, assist with not having to go through the entire refitting process again manually (although, there is always the option to save the entire object with `save(refit, file = "refit.RData))`

```
head(new_refit)
#>    Subject Group          LookType        fit        R2  AR1 fitCode
#> 1:       1    50            Cohort <gnls[18]> 0.9697202 TRUE       0
#> 2:       1    50 Unrelated_Cohort <gnls[18]> 0.9789994 TRUE       0
#> 3:       1    65            Cohort <gnls[18]> 0.9804901 TRUE       0
#> 4:       1    65 Unrelated_Cohort <gnls[18]> 0.8716404 TRUE       1
#> 5:       2    50            Cohort <gnls[18]> 0.9811708 TRUE       0
#> 6:       2    50 Unrelated_Cohort <gnls[18]> 0.9561166 TRUE       0
```