PersonAlytics© User’s Guide

Stephen Tueller, Ty Ridenour, Derek Ramirez, Jessica Cance

2019-12-31

Table of Contents

# Introduction

The purpose of this user’s guide is to illustrate how to use PersonAlytics to analyze

1. Single case time series data, also known as single subject or N-of-1 studies. Interrupted time series design designs can be used to introduce one (or more) interventions which results in two (or more) phases (e.g., pre-intervention and post-intervention).
2. Small sample intensive longitudinal design data.
3. Idiographic clinical trial data (ICT), which uses a single-case or small sample longitudinal data set where all participants experience two or more treatment phases (e.g., baseline and follow-up).

These three types of data are analyzed using longitudinal mixed effects models, also known as hierarchical linear models, multilevel models, latent growth curve models, or mixed method trajectory analysis.

While there are already several R packages such as nlme and gamlss that can be used to implement these models, the purpose of the PersonAlytics package is to automate the following aspects of analyzing these types of data:

1. Model selection for detecting the shape of the trajectory of the dependent variable over time. Using a good fitting trajectory shape yields parameters that correspond to the shape and improve interpretability. Current options include polynomial growth (e.g., linear, quadratic, cubic, etc.) and piecewise growth (e.g., linear growth within each phase).
2. Model selection for detecting an appropriate time series model for the residuals. Getting a good fitting residual correlation model improves standard error estimation.
3. Analysis of mixed effects models in situations where the combination of predictors, outcomes, and/or individuals (i.e., in the case where individual level models are desired such as in personalized medicine) yields a number of analyses to unwieldy implement manually. This is especially true if the model selection process for the trajectory shape and the residual correlation model is conducted for each combination of predictors, outcomes, and/or individuals. High throughput analyses are achieved in PersonAlytics through parallelization. Jobs are split among two or more processors on a computer and run in parallel. Results are recombined at the end of the process. Included in PersonAlytics are options for the user to specify Type I error rate or False Discovery Rate (FDR) corrections. See the section title “High Throughput Examples” for details on implementing high throughput analyses.

**High Throughput**. The PersonAlytics package automates the task of conducting large numbers of analyses using high throughput computation. A large number of analyses may be required when the combination of predictors, outcomes, and/or individuals yields a number of analyses to unwieldy implement manually.

**Parallelization**. High throughput analyses are achieved through parallelization. Jobs are split among two or more processors on a computer and are run in parallel to each other. Results are recombined at the end of the process.

**High Throughput Example 1: Migraine Triggers**. 346 migraine patients were followed for 90 days. They recorded information on 71 potential migraine and non-migraine headache triggers such as alcohol, weather, and exercise. Individual models were required to determine the five person-specific triggers with the largest effect size. Even though 90 time points is large, it was deemed to small to estimate all 71 trigger effects simultaneously, so trigger effects were estimated one at a time. The analysis required 346 patients X 2 outcomes X 71 triggers = 49,132 PersonAlytic runs.

**High Throughput Example 2: THC Metabolomics**. 17 patients participated in a two-phase design. The baseline phase had 2 hours of observation. The intervention was 25mg of THC and the intervention phase had 6 hours of observation. A total of 20 time points generated blood metabolomic data and outcomes including sleepiness, reaction time, memory, and behavior. The analysis required 18,023 chemical compounds (the metabolites) X 8 outcomes = 144,184 PersonAlytic runs.

# The PersonAlytic framework

Ty to write content here

# Installing PersonAlytics

*Note. It is assumed that the user has basic familiarity with R. If needed, there are numerous online tutorials to help the user become familiar with R.* That said, we will define a few key terms here that are used throughout this document:

* Console: This is the part of R that can be used interactively. Code can be typed in and results will be displayed in the console.
* Object: Anything saved to R’s memory. For example, if you type x <- 1in the console, you have created an object x which has a value of 1. If you then type x into the console and press enter, x’s value of 1 will be printed to the console.
* Environment: This is R’s current “active memory” that stores all of the objects you have created or loaded into R. Type ls() to view them all.
* Function: An R object that takes in other R objects (called parameters), performs some computation on them, and may (or may not) return a new object. A simple example is the mean() function. The parameter is some numeric data called a vector such as x <- c(1,2,3). Passing x to the mean function mean(x) returns a value of 2.
* Parameter: Named R objects that are passed to an R function.
* R documentation: R functions, including the functions of PersonAlytics are documented. This includes a descriptions of their parameters and how they should be used. The documentation can be accessed by typing a questions mark followed by the name of the function (e.g., ?mean).
* Working directory: The directory on your computer where R is “working”. An R function may return objects to the environment or they may save results to the working directory. To see the current working directory type ‘getwd()’.

First the user must install R, available at <https://cran.r-project.org/>. It is also suggested that the user install a modern code editor such as Rstudio available at [https://www.rstudio.com/](%60https://www.rstudio.com/%60). It is assumed that the reader is familiar with basic R use. If needed, an internet search will provide tutorials to help the user become familiar with R. The RTools software (which is not an R package) should be also installed from <https://cran.r-project.org/bin/windows/Rtools/> prior to installing Personalytics.

Installing PersonAlytics is done using the install\_github function of the devtools package. First install devtools using

install.packages('devtools')

After re-starting R, type the following into the console:

devtools::install\_github("ICTatRTI/PersonAlytics", build\_opts = c("--no-resave-data", "--no-manual"), build\_vignettes = TRUE)

The install\_github function may give the user the option to update other R packages needed by PersonAlytics that are already installed by the user. Unless the user is experienced with R package versions and updates, it is recommended that the user ignore this step and press enter without selecting an option to update packages. If the user does decide to update packages and runs into an error, they may need to restart R, manually update the package in the error message, and then rerun the install\_github command. If a newer version of a package is required by PersonAlytics the install\_github will attempt to install it.

Once install\_github starts running, it will install all the other R packages required by PersonAlytics if the user does not already have them. Building of the vignettes (such as this one) can take several minutes.

# Starting PersonAlytics

Now we can load PersonAlytics using

library(PersonAlytics)

After loading all of the R packages that PersonAlytics requires, you’ll see instructions to bring up this document by typing the following in the console (assuming you left build\_vignettes = TRUE when installing PersonAlytics):

vignette("PersonAlytics\_Users\_Guide")

*Note: If you are using Rstudio the R documentation, such as accessed by typing ?PersonAlytic in the console, will appear under the “Help” tab. Included in most documentation pages is an example section. If you highlight code in the example and press ctrl+enter, than code will run in the R console. Typing vignette("PersonAlytics\_Users\_Guide") will bring up this user’s guide in the “Viewer” tab. Although highlighted code cannot be run from the viewer, you can copy/paste from the guide into the console to run the examples herein.*

# Basic PersonAlytics Use

The PersonAlytic() function is the primary user interface for the PersonAlytics package and the user can access the documentation for PersonAlytic() by typing

?PersonAlytic

## The OvaryICT Data

We will illustrate the basic parameters of the PersonAlytic() function using a data set called OvaryICT, which contains data modified from the Ovary data in the nlme package. A description is available by typing ?Ovary in the R console. The OvaryICT data set was modified to include a phase variable to represent the structure of an ICT. A description of the modifications are available by typing ?OvaryICT in the R console. The OvaryICT data come with the PersonAlytics package and are used in most of the documentation examples. A phase variable is an essential component of an ICT. A typical phase variable represents pre- and post-treatment phases of a study, and more that two phases are an option in PersonAlytics.

The first six rows of the OvaryICT data are shown in Table x. Note that the data are in ‘long format’. In the example below, Mare 1’s first six time points are represented in separate rows. The time points are days prior to ovulation (for negative values), at ovulation (for values of 0 or 1) or between ovulations (for positive values other than 0 and 1). In addition to the phase variables (phase 1 and phase 2), the OvaryICT data also includes six randomly generated predictors (or independent variables) named Target1 to Target6 which will be used to illustrate other PersonAlytics features.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Mare | Time | follicles | Phase | Phase2 | Target1 | Target2 | Target3 | Target4 | Target5 | Target6 |
| 83 | 1 | -0.14 | 9 | 0 | 1 | 3 | 3 | 4 | -0.53 | 0.18 | 0.74 |
| 84 | 1 | -0.09 | 9 | 0 | 1 | 3 | 1 | 4 | -1.37 | 1.34 | 0.63 |
| 85 | 1 | -0.05 | 7 | 0 | 1 | 1 | 4 | 4 | -2.21 | 1.20 | 1.24 |
| 86 | 1 | 0.00 | 6 | 0 | 1 | 4 | 2 | 4 | 1.82 | 0.87 | 0.23 |
| 87 | 1 | 0.05 | 7 | 0 | 1 | 1 | 1 | 2 | -0.65 | -0.12 | -0.31 |
| 88 | 1 | 0.09 | 6 | 0 | 1 | 4 | 3 | 4 | -0.28 | 0.34 | 1.50 |

## Required PersonAlytics Parameters

The four required parameters for PersonAlytic() are

1. data: the name of the user’s data set. This data needs to have been read into R prior to using PersonAlytic(). A web search can be used to learn how to read in multiple types of data into R, including (but not limited to) csv, xlsx, sas, stata, spss, and most database formats. The data must be structured in ‘long format’ where time points are repeated within individual as was illustrated above for the OvaryICT data. The examples in this user’s guide are based on the OvaryICT data.
2. ids: the name of the identification variable for individuals. This must be a quoted variable name matching the user’s ID variable in the data set that the user provided to the parameter data.
3. dvs: the name of the dependent variable. This must be a quoted variable name matching the user’s dependent variable in the data set that the user provided to the parameter data. If the user has multiple dependent variables, PersonAlytics will iterate over them one at a time. Multiple dependent variables are specified as a character list, e.g., dvs=list('dv1', 'dv2', etc.). This is a high throughput option that will be described in more detail in the section titled “High Throughput Options”.
4. time: the name of the time variable. This must be a quoted variable name matching the user’s dependent variable in the data set that the user provided to the parameter data.

**list()**. In R, a list is a collection of variables or R objects that may or may not be of the same type. For example, a list might all be character strings, as is the case with dvs=list('dv1', 'dv2', etc.). In other situations, a list may mix data frames, character vectors, or other data types.

We now illustrate using the four basic parameters with the OvaryICT data. This example also includes setting autoSelect=NULL, which turns off automated model comparisons for selecting the trajectory shape and residual correlation structure, options that are discussed in the “Autoselection” section.

eg\_required <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 time="Time",  
 autoSelect=NULL)

It is important to assign the results of a call to PersonAlytic to an R object. In the example above, the object is named eg\_required. If the user neglects to assign the results to an R object, minimal output will be printed to the screen and the remaining results will be lost. If high throughput is initialized by including more than one dependent variable, including more than one target independent variables (see parameter target\_ivs below), or if requesting individual models (see parameter individual\_mods below), the resulting object is a data.frame concatenating results across the parameters submitted to the high throughput run. Reading the output is detailed in the section titled “Reading PersonAlytics Output”.

## Commonly Used Optional PersonAlytics Parameters

In this section we introduce commonly used optional PersonAlytics parameters.

### The phase variable

The phase parameter specifies the phase variable. While a phase variable is required for an ICT, the PersonAlytic() function does not require a phase variable to allow for other analyses such as time series analyses and small sample intensive longitudinal data. The phase parameter must be a quoted variable name matching phase variable in the data set that the user provided to the parameter data.

eg\_phase <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 autoSelect=NULL)

By default, the phase by time interaction will be included in the model. However, if the parameter alignPhase=piecewise, a piecewise growth model is fit instead and the time by phase interaction will be dropped if it model fails to converge with the interaction term. The ‘alignPhase’ parameter is described in the section on other optional parameters.

### The Shape of the Trajectory

By default, a linear growth model is fit to the data (time\_power=1). Any integer may be supplied to time\_power to specify the corresponding trajectory shape (e.g., linear, quadratic, cubic, etc.). Alternatively, PersonAlytics can automatically detect the order for time using the TO parameter of the autoSelect parameter as described in the section on autoselection.

Here is an example of a quadratic growth trajectory specified using time\_power=2:

eg\_time\_power <- PersonAlytic(output="MyResults",  
 data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 time\_power=2,  
 autoSelect=NULL)

### Residual Correlation Structure

The residual autocorrelation structure can be specified by the user using the correlation parameter. The correlation parameter default is NULL, corresponding to no within-group correlations.

Any option listed in the ?corStruct documentation of the nlme package can be specified (see also the correlation parameter in ?lme). When specifying a correlation structure, the parameters P and Q must be specified explicitly and the entire correlation structure must be in quotes. Alternatively, PersonAlytics can automatically detect the values for P and Q for an ARMA(P,Q) model using the AR option of the autoSelect parameter as described below.

Here is an example specifying an ARMA(3,4) residual correlation structure:

eg\_correlation <- PersonAlytic(output="MyResults",  
 data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 correlation="corARMA(p=3,q=4)",  
 autoSelect=NULL)

Note that the parameters p and q must be lower case and the entire correlation structure must be in quotes. Alternatively, PersonAlytics can perform automatic model selection for the values for p and q in an ARMA(p,q) model using the AR parameter of the autoSelect parameter as described in the section on autoselection.

*Note: For users unfamiliar with residual correlation structures (or other features of the nlme package), a helpful reference is the book “Mixed-Effects Models in S and S-PLUS” by Pinheiro and Bates (2000).*

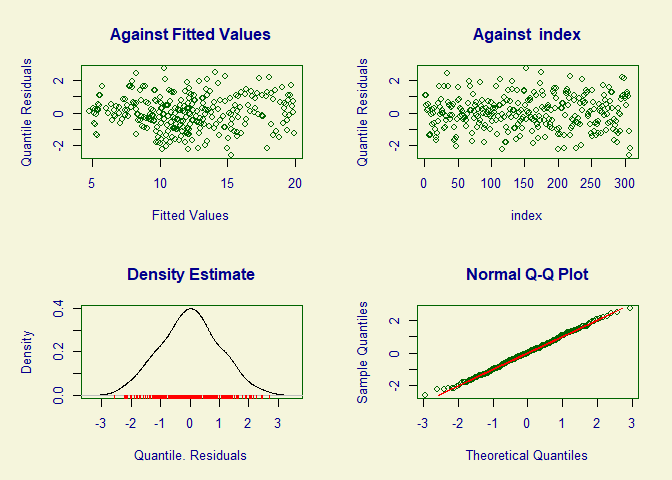
### Distributional Assumptions

By default, a linear mixed effects model is fit assuming a normal distribution for the outcome (or, more precisely, for the residuals). If the package parameter is set to 'gamlss', the family parameter can be used to specify any of the distributions available in the ?gamlss.family package. Once a model is fit, the plot function can be used to examine the residuals if a single model is fit (i.e., dvs has only one variable, target\_ivs is not used, and individual\_mods is FALSE).

Here is an example of a normal model fit using gamlss.

eg\_normal <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 family=NO(),  
 package="gamlss",  
 autoSelect=NULL)

plot(eg\_normal)

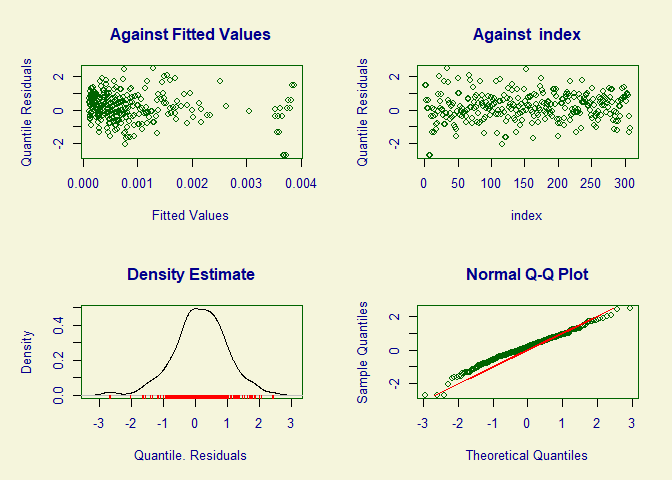


## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
## Summary of the Quantile Residuals  
## mean = 4.314702e-17   
## variance = 1.003257   
## coef. of skewness = 0.02660023   
## coef. of kurtosis = 2.684892   
## Filliben correlation coefficient = 0.998804   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here, the above example is repeated, this time using a Weibull Type 2 distribution:

eg\_weibull2 <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 family=WEI2(),  
 package="gamlss",  
 autoSelect=NULL)

plot(eg\_weibull2)



## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
## Summary of the Quantile Residuals  
## mean = 0.1554844   
## variance = 0.680279   
## coef. of skewness = -0.2875389   
## coef. of kurtosis = 3.794029   
## Filliben correlation coefficient = 0.9942695   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Alternatively, the PersonAlytic() function can automatically select the best fitting distribution using the DIST parameter described in the autoselection section below.

# High Throughput Options

High throughput automation is invoked by any combination of the following:

1. Multiple dependent variables, which are analyzed one at a time.
2. Multiple target independent variables, which are included one at a time. This is often necessary when there are too many independent variables to be analyzed simultaneously as illustrated in the two high throughput examples (migraine triggers and THC metabolomics).
3. Individual level models, in which separate models are fit for each participant in the data set. This is necessary when it is unrealistic to expect each individual to have the same residual correlation structure or if it is expected that each participant will have a unique set of target independent variables that have the biggest effects on their outcomes as is the case in the migraine triggers example.

## Multiple Dependent Variables

Here we illustrate a high throughput example by first creating two additional outcomes and fitting a basic growth model to each outcome. Note that creating variables that are the square and the root of the outcome is not advised unless the user has substantive reasons to do so, this is simply for illustration purposes:

OvaryICT$follicles2 <- OvaryICT$follicles^2  
OvaryICT$folliclesr <- sqrt(OvaryICT$follicles2)  
eg\_htp <- PersonAlytic(output="htp\_example",  
 data=OvaryICT,  
 ids="Mare",  
 dvs=list("follicles", "follicles2", "folliclesr"),  
 phase="Phase",  
 time="Time",  
 autoSelect=NULL)

##   
##   
## Model fitting starting...

##   
 |   
 |========================== | 33%  
 |   
 |==================================================== | 67%  
 |   
 |==============================================================================| 100%

##   
##   
## Model fitting took:  
## Time difference of 4.714972 secs.

## Independent Variables and Target Independent Variables

Independent, or predictor, variables can be added using the ivs parameter. This might be a grouping variables (e.g., treatment/control) or demographic variable (e.g., age or sex). If there are more than one dependent (or outcome) variables in dvs, the variables in ivs will be included as predictors for each dependent variable.

eg\_ivs <- PersonAlytic(output="MyResults",  
 data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 ivs=list("Target1", "Target2", "Target3"),  
 autoSelect=NULL)

If the user wishes to iterate over multiple independent variables one at a time, use target\_ivs. This is a high throughput option.

eg\_target\_ivs <- PersonAlytic(output="MyResults",  
 data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 target\_ivs=list("Target4", "Target5", "Target6"),  
 autoSelect=NULL)

##   
##   
## Fitting models of the dependent variable `follicles` for 11 cases in `Mare`  
## and for 3 target independent variables in `target\_ivs`.

##   
 |   
 |========================== | 33%  
 |   
 |==================================================== | 67%  
 |   
 |==============================================================================| 100%

##   
##   
## Model fitting for the dependent variable `follicles` took:  
## Time difference of 4.517 secs.

If the user has some independent variables that should be in every model and some that should be added one at a time, both ivs and target\_ivs can be used. Here, Target1-3 will be included in every model, but Target4-6 will be added one at a time.

eg\_target\_ivs2 <- PersonAlytic(output="MyResults",  
 data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 ivs=list("Target1", "Target2", "Target3"),  
 target\_ivs=list("Target4", "Target5", "Target6"),  
 autoSelect=NULL)

##   
##   
## Fitting models of the dependent variable `follicles` for 11 cases in `Mare`  
## and for 3 target independent variables in `target\_ivs`.

##   
 |   
 |========================== | 33%  
 |   
 |==================================================== | 67%  
 |   
 |==============================================================================| 100%

##   
##   
## Model fitting for the dependent variable `follicles` took:  
## Time difference of 4.496002 secs.

If the user wants variables to be dummy coded, the user should first specify this in the data set. For example, this code specifies that the Target1 variable should be treated as a categorical variable and dummy coded in any future analyses:

OvaryICT$Target1 <- factor(OvaryICT$Target1)

For more information about factors in R, users can read the documentation by typing ?factor in the console or conducting an internet search for tutorials in factors in R.

The interactions parameter can be used to specify pairs of independent variables to be included in interaction terms. Main effects for all variables in the interaction terms will be estimated even if they are not listed in the ivs or target\_ivs parameter as shown in this example.

eg\_interactions <- PersonAlytic(output="MyResults",  
 data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 interactions=list(c("Target1", "Target2"), c("Target1", "Target3"),   
 c("Target2", "Target3")),  
 autoSelect=NULL)

## Individual Level Models

PersonAlytics can automate the task of fitting individual models to each case in the data set. This is useful for applications such as individualized medicine such as the migraine data example in the introduction. In that example, the investigators wanted to find the most potent migraine triggers for each patient (an individual level question), not the triggers that were most common across all patients (a sample level question). The option of fitting individual models is enabled by setting individual\_mods=TRUE. The resulting output will be the data.frame saved to the assigned object in your R session. This will also be saved to a csv file named using the output parameter. In each of these two equivalent forms of output, each row corresponds to a separate case.

## Combinations of dvs, target\_ivs and individuals\_mods

Any combination of dvs, target\_ivs and individuals\_mods can be specified. The migraine triggers example uses all three types of variables, and the THC metabolomics example uses both target independent variables (the metabolites) and dependent variables (the 8 outcomes). The PersonAlytic function sets up the parallelization to minimize redundant operations depending on which combination of high throughput parameters the user requests.

# Reading PersonAlytics Output

The parameter output is a character string that is used to name a file for saving output. If left NULL, the default is ‘PersonAlytic\_Output’. Do not give a file extension, these will be added automatically depending on whether a single analysis was run (yielding a .txt file) or high throughput parameters were invoked (yielding a .csv file).

## Single Analysis Output

Here we fit the same model using nlme and gamlss and illustrate options for viewing and manipulating the output. Documentation for these R packages can be obtained by typing

library(nlme)  
?lme  
library(gamlss)  
?gamlss

The nlme package implements the linear (normal) mixed effects model using the lme() function. The gamlss package generalizes the mixed effects model for over 80 non-normal distributions. Typing ?gamlss.family in the console will bring up the full list of distributions.

In PersonAlytics, using the package parameter can be used to specify the gamlss package instead of nlme, which is the default. When package="gamlss", the family parameter is used to specify the distribution using the “R names” column in the documentation for ?gamlss.family (e.g., family=BE() for beta regression). The family parameter is described in more detail in the section titled “Distributional Assumptions” above. The default distribution assumed when using package="gamlss" is the normal distribution (i.e., family=NO()).

In the current example we first run the model using package="nlme". Since this is the default, we do not need to explicitly specify the package parameter though we do so in this example. The output will be saved the the R object eg\_nlme for further use inside the R console. In addition, output="nlme\_example" will create a file in the working directory named ‘nlme\_example.txt’. If the user are unsure what the user’s current working directory is, type getwd() into the R console. A full path using forward slashes ‘/’ instead of backslashes ’\' can also be used. For example, output='C:/MyResults'. For convenience, we will also turn off the autoSelect parameter by setting it to NULL. The autoSelect parameter is discussed in detail below.

eg\_nlme <- PersonAlytic(output="nlme\_example",  
 data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 package="nlme",  
 autoSelect=NULL)

Then we run the same model using package="gamlss". The output will be saved the the R object eg\_gamlss for further use inside the R console. In addition, output="gamlss\_example" will create a file in the working directory named ‘gamlss\_example.txt’.

eg\_gamlss <- PersonAlytic(output="gamlss\_example",  
 data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 package="gamlss",  
 autoSelect=NULL)

In these examples, the R object eg\_nlme is a class lme object, and the R object eg\_gamlss is a class gamlss object.

class(eg\_nlme)

## [1] "lme"

class(eg\_gamlss)

## [1] "gamlss" "gam" "glm" "lm"

Both lme and gamlss objects have a summary() method which prints detailed results to the R console. For the nlme example we get the following.

summary(eg\_nlme)

Linear mixed-effects model fit by REML  
 Data: tempData   
 AIC BIC logLik  
 1670.449 1700.185 -827.2244  
  
Random effects:  
 Formula: ~Time | Mare  
 Structure: General positive-definite, Log-Cholesky parametrization  
 StdDev Corr   
(Intercept) 2.751709 (Intr)  
Time 3.871676 -0.202  
Residual 3.317972   
  
Fixed effects: follicles ~ Time \* Phase   
 Value Std.Error DF t-value p-value  
(Intercept) 10.66163 0.9016267 294 11.824883 0.0000  
Time -0.86898 1.7667760 294 -0.491845 0.6232  
Phase 10.97209 1.3126329 294 8.358845 0.0000  
Time:Phase -8.64385 1.9742483 294 -4.378299 0.0000  
 Correlation:   
 (Intr) Time Phase   
Time -0.318   
Phase -0.105 0.135   
Time:Phase 0.175 -0.504 -0.817  
  
Standardized Within-Group Residuals:  
 Min Q1 Med Q3 Max   
-2.49764683 -0.65159342 -0.01330142 0.61976543 2.61092061   
  
Number of Observations: 308  
Number of Groups: 11

The fixed effects results in the section titled ‘Fixed Effects’ is what gets saved to the file ‘nlme\_example.txt’. For the gamlss example we get the following:

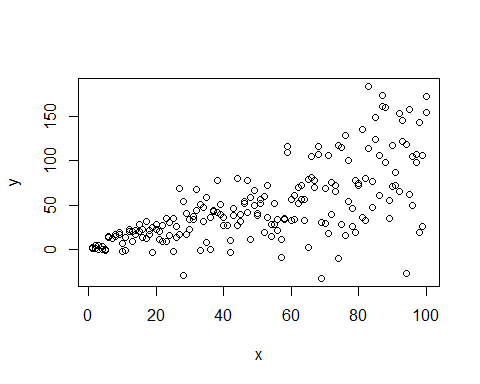
summary(eg\_gamlss)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Family: c("NO", "Normal")   
  
Call: gamlss::gamlss(formula = self$formula, sigma.formula = sigma.formula,   
 family = currentFamily, data = tempData, control = ctrl)   
  
Fitting method: RS()   
  
------------------------------------------------------------------  
Mu link function: identity  
Mu Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 10.6898 0.3404 31.402 < 2e-16 \*\*\*  
Time -0.8637 1.2787 -0.675 0.5   
Phase 10.9725 1.2654 8.671 3.28e-16 \*\*\*  
Time:Phase -8.6439 1.9033 -4.541 8.25e-06 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
------------------------------------------------------------------  
Sigma link function: log  
Sigma Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 1.16333 0.04029 28.87 <2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
------------------------------------------------------------------  
NOTE: Additive smoothing terms exist in the formulas:   
 i) Std. Error for smoothers are for the linear effect only.   
ii) Std. Error for the linear terms may not be reliable.   
------------------------------------------------------------------  
No. of observations in the fit: 308   
Degrees of Freedom for the fit: 23.39381  
 Residual Deg. of Freedom: 284.6062   
 at cycle: 2   
   
Global Deviance: 1590.679   
 AIC: 1637.467   
 SBC: 1724.728   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The fixed effects results in the section titled ‘Mu Coefficients’ and the variance coefficients in the section titled ‘Sigma Coefficients’ is what gets saved to the file ‘gamlss\_example.txt’.

It is beyond the scope of this document to detail the differences between the nlme and gamlss approaches, but we will highlight one important difference. Notice that the parameter estimates in the ‘Value’ column of the ‘Fixed Effects’ section of the eg\_nlme output and the ‘Estimate’ column of the ‘Mu Coefficients’ section of the eg\_gamlss output are very similar. However, the standard errors are of the gamlss model are lower. This is because the gamlss approach models not only the mean, but also the variance. If there is any heteroscedasticity of variance, the gamlss results will consequently have lower standard errors than the nlme results. In the gamlss output, the variance parameter(s) are in the ‘Sigma Coefficients’ section.

**Heteroscedasticity**. heteroscedasticity occurs in data when the variability of the dependent variable is unequal across the range of values of one (or more) predictors. In this example, the the variance of y gets larger as values of the predictor x get larger.



## Multiple Analysis/High Throughput Analysis Output

For this section we will reuse the “Multiple Dependent Variables” example of the “High Throughput Options” section:

OvaryICT$follicles2 <- OvaryICT$follicles^2  
OvaryICT$folliclesr <- sqrt(OvaryICT$follicles2)  
eg\_htp <- PersonAlytic(output="htp\_example",  
 data=OvaryICT,  
 ids="Mare",  
 dvs=list("follicles", "follicles2", "folliclesr"),  
 phase="Phase",  
 time="Time",  
 autoSelect=NULL)

The output will be saved in the R object eg\_htp for further use inside the R console and output="htp\_example" will create a file in the working directory named ‘htp\_example.csv’ that is a saved version of the R object eg\_htp. This csv file can be opened in any spreadsheet program. The rows are all possible combinations of dvs, target\_ivs, and ids (if individual\_mods=TRUE). The column variables are listed below and come in five sets:

1. Variables identify the combination of user inputs that lead to a given row’s analysis. Most column names correspond to their respective parameter name in PersonAlytic. Other variables include information on the version of PersonAlytics, date, time, and the directory in which the models were run.
   * ‘Mare’: This is is the name of the id variable passed to the ids parameter. In the current example, the value is ‘All Cases’. If individual models are requested by setting individual\_mods=TRUE, this column will give the id for each case in the data set.
   * ‘ids’: This is the name of the ids variable which is the column name of column 1.
   * ‘dv’: The name of the dependent variable.
   * ‘time’: The name of the time variable.
   * ‘ivs’: The names of the indepedent variables (describe below).
   * ‘target\_iv’: The name of a target independent variable (described below).
   * ‘interactions’: The names of the interaction terms (described below).
   * ‘time\_power’: The shape of the trajectories over time (described below).
   * ‘alignPhase’: How time was realigned by phase, if any (described below).
   * ‘correlation’: The residual correlation structure (described below). If the value is NULL, this corresponds to no within-group correlations (see ?lme).
   * ‘family’: The distribution for the dependent variable (described below).
   * ‘standardize’: Which variables were standardized (described below).
   * ‘method’: The estimation method.
   * ‘package’: Which R package was used to fit the models.
   * ‘PersonAlytics’: The version of the PersonAlytics used for the analysis.
   * ‘Date\_Time’: The date and time the model was run.
2. Variables for checking whether the data were read in correctly.
   * ‘N\_participants’: The number of participants or cases in the data set.
   * ‘N\_time\_points’: The number of time points for the participant with the longest time series.
   * ‘N\_time\_points\_complete’: The number of time points with complete data.
   * ‘dvVariance’: The variance of the depenent variable.
   * ‘timeVariance’: The variance of the time variable.
   * ‘ivVariance’: The variance of the independent variables.
   * ‘target\_ivVariance’: The variance of the target independent variable.
   * ‘converge’: Model convergence status.
   * ‘estimator’: What model estimator was used (described below).
   * ‘analyzed\_N’: The number of observations analyzed. This is the sum of time points, or the sum of the time points across all participants if multiple cases were analyzed.
3. Information about the analysis
   * ‘call’: The model formula created from the user inputs.
   * ‘wasLRTrun’: If ‘target\_ivs’ were provided, a likelihood ratio test (LRT) for models with and without the target independent variable will be attempted, and if succesfull, ‘wasLRTrun’ will be ‘TRUE’.
   * ‘targ\_ivs\_lrt\_pvalue’: If the LRT was run, the p-value is recorded here.
   * ‘fixed’: The fixed effects portion of the ‘call’.
   * ‘random’: The random effects portion of the ‘call’.
   * ‘formula’: This is similar to the ‘call’ variable which gives the intended formula. If the model will not converge using the intended formula, simplifications of the formula are attempted in the following order:
     + No correlation structure
     + No correlation structure and no random slopes
     + If the model is piecewise, drop the phase by time interaction
   * ‘correlation0’: The correlation portion of the ‘call’.
   * ‘directory’: The directory where model output is saved.
   * ‘date’: The date the output was saved (which may be different from the ‘Date\_Time’ model was run for long runs).
4. Descriptive statistics in pairs with the first column describing the statistic with the prefix statName and the second column in each pair with the prefix statValue giving the statistic’s value.
5. Model results with a parameter estimates, standard error, t-value, degrees of freedom, and p-value.
6. If the finite population correction (FPC) is specified (see below for details), the model results repeated with FPCs for the standard errors (and consequently, the p-values). The example below doesn’t include the FPC, but if it did the parameter estimates, standard errors, t-values, degrees of freedom, and p-values would be repeated with the suffix fpc.

Here are all the column names for the current example:

[1] "Mare" "ids" "dv"   
 [4] "time" "phase" "ivs"   
 [7] "target\_iv" "interactions" "time\_power"   
[10] "alignPhase" "correlation" "family"   
[13] "gamlss.family" "standardize" "method"   
[16] "package" "Personalytics" "Date\_Time"   
[19] "N\_participants" "N\_time\_points" "N\_time\_points\_complete"  
[22] "dvVariance" "timeVariance" "ivVariance"   
[25] "target\_ivVariance" "converge" "estimator"   
[28] "analyzed\_N" "call" "wasLRTrun"   
[31] "targ\_ivs\_lrt\_pvalue" "fixed" "random"   
[34] "formula" "correlation0" "directory"   
[37] "date" "statName1" "statValue1"   
[40] "statName2" "statValue2" "statName3"   
[43] "statValue3" "X.Intercept..Value" "X.Intercept..Std.Error"  
[46] "X.Intercept..DF" "X.Intercept..t.value" "X.Intercept..p.value"   
[49] "Time.Value" "Time.Std.Error" "Time.DF"   
[52] "Time.t.value" "Time.p.value" "Phase.Value"   
[55] "Phase.Std.Error" "Phase.DF" "Phase.t.value"   
[58] "Phase.p.value" "Time.Phase.Value" "Time.Phase.Std.Error"   
[61] "Time.Phase.DF" "Time.Phase.t.value" "Time.Phase.p.value"

# Autoselection of the Residual Autocorrelation Structure, Time Order, and Dependent Variable Distribution

The Autoselection parameters automate the tedious process of conducting ML model comparisons to determine any of three parameters described in this section. The value NULL (autoSelect=NULL), or an empty list (autoSelect=list()) turns all options off. Leaving any parameter out of the list will turn that parameter off (e.g., autoSelect=list(TO=list(polyMax=3)) will only implement Autoselection of the time order). The default is autoSelect=list(AR = list(P = 3, Q = 3), TO = list(polyMax = 3), DIST = list()). As noted above, a list in R is a collection of objects (or variables) that need not be of the same type. In this example,

* autoSelect is a list that has length 3.
* The first object in autoSelect is another list, AR = list(P = 3, Q = 3).
  + AR has length 2
  + The first value is named P and has a value of 3.
  + The second value is named Q and also have a value of 3.
* The second object in autoSelect is also a list, TO = list(polyMax = 3). In this example TO has length 1 containing a value named polyMax with a value of 3.
* The final object in autoSelect is a list DIST = list(). DIST has length 0, i.e., it is an empty list.

This preceding discussion is meant to familiarize new R users with lists, and AR, TO, and DIST are explained in more detail in the remainder of this section.

## Residual Correlation Structure AR

AR is the autoregressive moving-average (ARMA) order of the residual correlation structure. Determining a good fitting correlation structure will provide more accurate standard errors. The default, AR=list(P=3, Q=3) will search all combinations of p=c(0,1,2,3) and q=c(0,1,2,3), as well as the default with no within-group correlations (correlation=NULL), for the best fitting ARMA correlation structure. This is done with a fit index instead of ML likelihood ratio tests (LRT) because not all ARMA models are nested (a requirement of the LRT). The fit index that will be used is set using the whichIC parameter with options BIC or AIC. It is beyond the scope of this document to discuss the LRT or choosing between BIC and AIC, and users should leave the default as BIC until they have familiarized themselves with the differences between them. If or individual\_mods=TRUE, correlation model selection is implemented using the auto.arima function of the forecast package. See ?auto.arima.

Here is an example of using the AR parameter of the autoSelect parameter without the TO or DIST parameters:

eg\_autoSelect\_AR <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 method="ML",  
 package="gamlss",  
 autoSelect = list(AR=list(P=3, Q=2)))

##   
##   
## PersonAlytics: Automatic detection of the  
## residual correlation structure starting...

##   
 |   
 |========================== | 33%  
 |   
 |==================================================== | 67%  
 |   
 |==============================================================================| 100%

##   
## Automatic detection of the residual  
## correlation structure took: Time difference of 16.179 secs.

##   
## The best correlation structure among those tested is nlme::corARMA(p=0,q=2)

The result is that the best correlation structure is nlme::corARMA(p=0,q=2), an ARMA model with an autoregressive (AR) order of 0 and a moving average (MA) order of 2.

## The Trajectory Shape TO

This parameter sets the shape of the trajectory using the polynomial order of the time/outcome relationship. Determining a good shape for the trajectory over time helps with model interpretation. Automatic detection is implemented using likelihood ratio test of the model with vs. where is the polynomial order of the time variable. The default is TO=list(polyMax=3), where polyMax is the largest values of to test. As noted above, time\_power is used when the user wants to specify a value for , and doing so requires setting TO=list() or leaving it out of the autoSelect parameter.

Here is an example of using the TO parameter of the autoSelect parameter, setting the maximum polynomial order of the time variable to 4 without the AR or DIST parameters:

eg\_autoSelect\_TO <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 method="ML",  
 package="gamlss",  
 autoSelect = list(TO=list(polyMax=4)))

##   
##   
## PersonAlytics: Automatic detection of the  
## time/outcome relationship starting...

##   
## Automatic detection of the time/outcome  
## relationship took: Time difference of 35.42201 secs.  
##   
## The best polynomial order for time from 1 to 4 was 3

The result is that the best polynomial order is 3, which is a cubic growth model.

An alternative to using a polynomial order for time is to approximate the trajectory shape using a separate linear model within each of the phases. This can be set using the piecewise parameter of the alignPhase parameter as described below.

## Dependent Variable Distributional Assumption DIST

The DIST parameter autoselect the best fitting distribution of the dependent variable using the fitDist function of the gamlss package. To see all of the available options, type

?gamlss.family

into the console. The default is DIST=list(). To turn off Autoselection of the best fitting dependent variable distribution, remove DIST from the autoSelect list.

Here is an example of using the DIST parameter of the autoSelect parameter without the TO or AR parameters:

eg\_autoSelect\_DIST <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 package="gamlss",  
 autoSelect = list(DIST=list()))

##   
## The variable follicles has the following characteristics:   
## Integer : TRUE   
## Binary : FALSE   
## Proportion : FALSE   
## Positive : TRUE   
## Multinomial : FALSE (integer with >2 & <= 5 categories)   
## Count : TRUE (positive integer)   
## Continuous : FALSE (non-integer)   
##   
## Error in solve.default(oout$hessian) :   
## Lapack routine dgesv: system is exactly singular: U[4,4] = 0  
## Error in solve.default(oout$hessian) :   
## Lapack routine dgesv: system is exactly singular: U[3,3] = 0  
## Error in solve.default(oout$hessian) :   
## Lapack routine dgesv: system is exactly singular: U[4,4] = 0  
## Error in solve.default(oout$hessian) :   
## Lapack routine dgesv: system is exactly singular: U[4,4] = 0  
##   
## Family: c("WEI2", "Weibull type 2")   
## Fitting method: "nlminb"   
##   
## Call: gamlssML(formula = y, family = DIST[i], data = sys.parent())   
##   
## Mu Coefficients:  
## [1] -6.686  
## Sigma Coefficients:  
## [1] 0.9418  
##   
## Degrees of Freedom for the fit: 2 Residual Deg. of Freedom 306   
## Global Deviance: 1856.73   
## AIC: 1860.73   
## SBC: 1868.19

##   
## To explore this distribution install `gamlss.demo` and type  
##   
## dev.new()  
## gamlss.demo::demoDist()  
##   
## into the console, find your distribution, and use the  
## slider bars to select the parameters printed above  
## (mu, sigma, nu, and tau).

##   
## Descriptive statistics by phase:  
## mean median sd skewness kurtosis  
## 0 10.52 10 4.07 0.23 2.92  
## 1 13.65 13 5.47 0.12 2.00  
## OverAll 12.04 12 5.04 0.38 2.54

The output above is printed to the console, and users should focus on the following parts:

* The first section titled “The variable follicles has the following characteristics” describes the variable. If any of these appear incorrect, check your data.
* The next section may contain errors. This occur when distributions are fit to the dependent variable but do not converge. These distributions will be removed from consideration.
* The section starting with “Family:” returns the best fitting distribution. In this example, the best fitting distribution is the “Weibull type 2” distribution. The resulting object eg\_autoSelect\_DIST is a gamlss object fit using the Weibull type 2 distribution. The remaining output gives parameter estimates and descriptive statistics for the best fitting distribution.

# Other Optional PersonAlytic() Parameters

## Subgroup Analysis

The subgroup parameter can be used to fit the model to a subset of the data. A logical (TRUE/FALSE) or binary (0/1) vector can be used to specify which cases to use.

Here is an example restricting the analysis to the first 5 mares in the OvaryICT data set. The code OvaryICT$Mare<6 creates a logical vector which is true if the mare id is less than 6:

eg\_subgroup <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 subgroup=OvaryICT$Mare<6,  
 autoSelect=NULL)

## Variable Standardization

Variable standardization (mean zero and unit variance) is facilitated using the standardize parameter which is a list with three parameters that can be set to TRUE or FALSE (the default for all three is FALSE):

* dv: if set to TRUE, the dependent variable is standardized. If there are multiple dependent variables in dvs, each dependent variable will be standardized.
* ivs: if set to TRUE, the the independent variables in ivs and target\_ivs are standardized. This is useful for determining which target independent variable has the largest effect size since standardization put their respective effect sizes on the same scale. All continuous variables should be standardized.
* byids: if set to TRUE, standardization is done within each individual. This parameter should be set to TRUE if individual\_mods=TRUE (individual\_mods parameter is discussed in the section on high throughput options).

Here is an example where the target independent variables are standardized:

eg\_standardize <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 target\_ivs=list("Target1", "Target2", "Target3"),  
 standardize=list(dv=FALSE, ivs=TRUE, byids=FALSE),  
 autoSelect=NULL)

##   
## PersonAlytics is standardizing the variables in `targe\_ivs`.

##   
##   
## Fitting models of the dependent variable `follicles` for 11 cases in `Mare`  
## and for 3 target independent variables in `target\_ivs`.

##   
 |   
 |========================== | 33%  
 |   
 |==================================================== | 67%  
 |   
 |==============================================================================| 100%

##   
##   
## Model fitting for the dependent variable `follicles` took:  
## Time difference of 4.051973 secs.

## Estimation Methods

A challenge of single subject and small sample mixed effect modeling is a lack of statistical power. One result of research in this area recommends that model comparisons be made using maximum likelihood (ML) estimation, while final model results should be estimated using restricted maximum likelihood (REML). The Autoselection parameters detailed below use ML model comparisons by default, and final results reported in the output are estimated using REML by default. This cannot be changed, but for a single model without Autoselection, the method parameter can be used to select ML instead of REML:

eg\_method <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 method="ML",  
 autoSelect=NULL)

## Dealing with Invalid Variable Names

If the names of the target predictors in target\_ivs had to be edited to make valid variable names (see ?make.names), the charSub parameter allows the user to put the illegal characters back in for the row variable names in high throughput output. For example, if the original variable name was “17.00\_832.2375m/z”, a letter would need to prefix the variable name and the “/” would need to be replaced with another character, (“X17.00\_832.2375m.z”). To get the row names of the output back to original variable name, use charSub=list(c("X", ""), c("m.z", "m/z")).

Note that inputs to charSub must be in double quotes and are case sensitive. All duplicates will be substituted. For example, if the variable name was “X1X23.x” and charSub=list(c(“X”, “”)), the resulting row label for this variable would be “123.x”.

## Type I Error or False Discovery Rate Adjustment

For high throughput analyses, we recommend selecting either a Type I error rate adjustment (such as Bonferonni) or a False Discovery Rate (FDR) adjustments (such as the method of Benjamini, Hochberg, and Yekutieli). This is implemented using the p.method parameter which takes on any value available in the ?p.adjust function. In conjunction with the p.method is the Type I error rate alpha, which has a default of .05. If there are multiple dependent variables in dvs, adjustments are made across target independent variables within each dependent variable. If individual\_mods=TRUE, adjustments are made across target independent variables within each dependent variable within each case.

Here is an example using the method of Benjamini, Hochberg, and Yekutieli (p.method=BY) and a Type I error rate of .1, adjusting for the FDR across the three predictors in target\_ivs:

eg\_p.method <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 target\_ivs=list("Target1", "Target2", "Target3"),  
 p.method='BY',  
 alpha=.1,  
 autoSelect=NULL  
 )

##   
##   
## Fitting models of the dependent variable `follicles` for 11 cases in `Mare`  
## and for 3 target independent variables in `target\_ivs`.

##   
 |   
 |========================== | 33%  
 |   
 |==================================================== | 67%  
 |   
 |==============================================================================| 100%

##   
##   
## Model fitting for the dependent variable `follicles` took:  
## Time difference of 5.390997 secs.

## Phase Alignment

The alignPhase parameter provides options for aligning the time variable with the phase variable. The default is alignPhase='none' and the time variable is left as-is.

1. alignPhase='align' aligns the time variable at the transition from the first and second phase within each participant. This alignment makes it so the effect at time=0 is the start of the second phase. For example, consider two participants who each have 15 time points of data, but the second phase starts at time 6 for participant A, and at time 8 for participant B:
   * Participant A’s resulting time variable will be -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
   * Participant B’s resulting time variable will be -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7.
2. alignPhase='piecewise' creates a piecewise straight-line growth model within each phase. This parameter can be used to simplify complex trajectories when the within-phase trajectory is approximately linear for each phase. Although this may make results easier to interpret than a curvelinear model (e.g., when time\_power=3), the piecewise model may have many more random effects and therefore be less parsimonious than a model with polynomial time. If alignPhase='piecewise', the TO parameter of the autodDetect parameter and the time\_power parameter are ignored.

## Finite Population Correction

A finite population correction (FPC) can be made if the size of the population from which the user’s sample originated is known. As an example, population sizes may be known for rare diseases. Preliminary simulation studies by the authors of this guide have found that on average, the FPC generally reduces standard errors, but only by a small amount. The parameter fpc has a default of 0, which turns off the FPC. If the user’s population size is known, set fpc to the population size. In this example, the population size is set to 6,000.

Here is an example where the user’s finite population size is 6,000.

eg\_fpc <- PersonAlytic(data=OvaryICT,  
 ids="Mare",  
 dvs="follicles",  
 phase="Phase",  
 time="Time",  
 ivs=list("Target1", "Target2", "Target3"),  
 fpc=6000,  
 autoSelect=NULL  
 )

The argument fpc can be set to the user’s finite population size and an FPC will be included in the analyses and output.

A finite population correction (FPC) can be used when the sample is more than 5% of the population. In this situation the central limit theorem doesn’t hold and the standard errors of the user’s parameter estimates will be too large. It is important to understand the conditions under which a finite population correction may make a difference in a power analysis conducting via simulation in the companion package PersonAlyticsPower. In the simulation, B data sets are simulated, a model is fit to the simulated data, and the proportion of data sets for which is the statistical power. The FPC will only make a difference in situation where a large number of the B replications have p-values just larger than alpha and the FPC results in these p-values being less than alpha. In our experience, this situation is rare but possible. The p-value distributions are usually smoothly positively skewed or near uniform. Neither of this distributions puts enough p-values near alpha for the FPC to have a large effect on power. Users can apply the FPC to real data analyses using the PersonAlytics package but they should not expect the FPC to yield large improvements in power.

## Processors for Paralellization

The cores parameter allows the user to specify how many processors (or cores) on their computer can be devoted to a high throughput PersonAlytics run. By default, the PersonAlytic() function detects the number of cores and uses one fewer than are on the machine, which allows the user to use other programs while PersonAlytics runs. If the user has a machine dedicated to analyses, setting the the number of cores to the maximum available will reduce computation time. Do not set this value to a number greater than the number of processors on the user’s machine or it may cause R or the user’s computer to crash. To determine the number of cores the user has, type the following into the R console:

parallel::detectCores()

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE