# draw\_tte

1. Would rather remove some of the defaults here, such as dist=’exp’ and coef1=1, to avoid any potential oversights when using this function.
   1. removed
2. In general, the variable “hr” is not always a hazard ratio but depends on the selected distribution. For AFT models, this variable represents an acceleration factor. Recommend changing this to a more general label (e.g., beta\_tx, tx\_effect etc.).
   1. Changed to beta\_tx
3. For most of the distributions supported by this function, the treatment effect is modeled in the form of a HR/AFT, whereas no treatment effect is explicitly modeled for beta or Poisson-gamma; how will treatment effects be modeled in these cases (e.g., separate fits)?
   1. They can be directly integrated into the relevant coefficient, the beta\_tx parameter was just to facilitate this integration to the user, but nothing stops the user from using it directly into the relevant parameter. We normally don’t run a treatment effect on utilities, so it was removed. It was unclear how it would affect the poisson-gamma, so I have not included it there for now.
4. For dist %in% (“lnorm”, “weibull”, “llogis”, “gengamma”) why is log(hr) subtracted from coef1/coef2? For Weibull, log-logistic, and generalized gamma, shouldn’t this be coef2 + log(exp(beta\_tx)) if your beta\_tx is coming from flexsurvreg (log-normal doesn’t require log transform)? This can be checked by fitting models in flexsurv to a simply toy example (e.g., lung or colon data from survival package). Jack and I previously looked into this when developing the TTEFit documentation and confirmed that the application of relative effects should be coef2 + beta\_tx if using output directly from a flexsurvreg fit. However, if your inputted AFT is already transformed as exp(-beta\_tx), then the current implementation may be correct.
   1. I believe this is a legacy issue from the original package that I have not modified, and in those cases there were some transformations of the parameters to make them consistent with SAS, so it could have come from there. I have modified it to be consistent based on this feedback (using +).
5. Recommend making the variables names for the model coefficients to be more descriptive, as coef1, coef 2 etc. may be a bit unclear for the end user. These could be called location and ancillary parameters to be consistent with the terminology used by flexsurv. At the moment coef1 is a location parameter for the log-normal and exponential distributions but an ancillary parameter for all others, so it’s a bit inconsistent.
   1. The order of the parameters is set to match the order of the coefficients returned by flexsurvreg as the description of the function mentions, which facilitates implementation.
6. Note that flexsurvreg reports results for two different parameterizations (res vs res.t): either 1) transformed to the real line (using log scale for certain parameters) or 2) on the natural scale. I recommend clearly documenting that for coef 1, 2, and 3, the former is assumed as input for the draw\_tte function. Related to my comment #4, it is unclear how “hr” is defined for AFT models.
   1. The functions have been parametrized to match the outcomes of coef() from flexusrvreg, and it’s clearly documented in the argument (“defined as in coef() output on a flexusrvreg object). This should also help with the PSA implementations, though for the PSA we’ll probably just use normboot.flexsurvreg for the relevant functions (which has the transform parameter so the scale does not really matter)
7. The conditional statement checking whether theta is missing to determine whether a Poisson or Poisson-gamma process should be implemented does not work correctly when calling rpoisgamma() via draw\_tte(). E.g., draw\_tte(n\_chosen=10,"poisgamma",coef1=1,obs\_time=1,return\_ind\_rate=FALSE) yields and error whereas rpoisgamma(n=10, rate = 1,obs\_time=1,return\_ind\_rate=FALSE) runs as expected. The reason being, the draw\_tte() function sets theta=coef2=NULL, and thus theta is no longer considered missing in the function rpoisgamma().
   1. Good catch, modified function so theta is by default NULL and the function checks for it being null as opposed to missing

# draw\_beta

1. Consider changing “value” to a more descriptive variable name
   * Switched to mean\_v
2. Should the option to apply treatment effects (similar to draw\_tte) be implemented?
   * Note this function is used when mean and se only are known, i.e., in the case of published utilities or similar. We normally don’t do beta regressions for the utilities, so I would not think this is a functionality needed.
3. Validated and confirmed that the function as currently coded is accurate and works as intended. However, it could be written more succinctly / efficiently. If we take advantage of vectorization in R, there is no need for a for loop. See below for proposed revisions.
   * Good point, updated

draw\_beta\_cc <- function(value,se,seed=NULL) {

if(!is.null(seed)){

set.seed(seed)

}

alpha <- ((1 - value) / (se^2) - (1 / value)) \* value ^ 2

beta <- alpha \* ((1 / value) - 1)

out <- rbeta(length(value),alpha,beta)

return(out)

}

# draw\_gamma

1. Consider changing “value” to a more descriptive variable name
   * See draw\_beta
2. Should the option to apply treatment effects (similar to draw\_tte) be implemented?
   * See draw\_beta
3. Validated and confirmed that the function as currently coded is accurate and works as intended. However, it could be written more succinctly / efficiently. If we take advantage of vectorization in R, there is no need for a for loop. See below for proposed revisions.
   * Updated, good point

draw\_gamma\_cc <- function(value,se,seed=NULL) {

if(!is.null(seed)){

set.seed(seed)

}

scale <- se^2 / value

shape <- value / scale

out <- ifelse(se==0,value,rgamma(length(value),shape,scale=scale))

return(out)

}

# draw\_resgompertz

1. Line 174, minor note on the name of the variable: these are not quantiles but survival probabilities.
   1. Renamed

# rpoisgamma

1. The conditional statement checking whether theta is missing to determine whether a Poisson or Poisson-gamma process should be implemented does not work correctly when calling rpoisgamma() via draw\_tte(). E.g., draw\_tte(n\_chosen=10,"poisgamma",coef1=1,obs\_time=1,return\_ind\_rate=FALSE) yields and error whereas rpoisgamma(n=10, rate = 1,obs\_time=1,return\_ind\_rate=FALSE) runs as expected. The reason being, the draw\_tte() function sets theta=coef2=NULL, and thus theta is no longer considered missing in the function rpoisgamma().
   1. Changed, see draw\_tte
2. Recommend being more explicit in the function’s definition re: Poisson vs Poisson-gamma distributions. E.g., include an argument to select the distribution.
   1. I would not use this as the argument is not really necessary. We could, however, just separate the function into two, one for poisson, one for poisson-gamma.
3. For the negative binomial process, I wonder if we are missing an integration step here for the rate parameter (). If targeting the posterior distribution of a Poisson-Gamma distribution in a Bayesian analysis, we would typically integrate over all possible rates ([see here](https://en.wikipedia.org/wiki/Negative_binomial_distribution#Gamma%E2%80%93Poisson_mixture)). Do we have any previous examples of NB processes that can be used for cross-validation?
   1. Jack produced a vignette showing some examples. If we averaged it out, however, wouldn’t we just be taking a constant rate across individuals then?
4. Line 246, for the rexp() function n should be set equal n\*t\_reps as opposed to just t\_reps.
   1. As a simple test, if I set the theta/size parameter equal to an arbitrarily large number (e.g., 100,000,000) then a negative binomial distribution with mean should be approx. equal to the Poisson distribution with mean . I.e., ppois(q=X, lambda=Y) and pnbinom(q=X, mu=Y, size=100000000000000000) are equal. If I set the same seed for the rexp() calls, then rpoisgamma(n, rate=1, obs\_time=1, theta=10000000000, seed=1234) should produce the approximately the same results as rpoisgamma(n, rate=1, obs\_time=1, seed=1234), which is achieved after applying the correction described above.
      1. The function has been modified to add the n\*t\_reps. Furthermore, the seed has been set so that it’s called just before the lapply, so that the outcomes are equal. This means the seed if condition is written twice.
5. Note, the data.table library has to be loaded prior to running this function if returning a data frame.
   1. Good point, it has been added to the importfrom argument in the function documentation. As it’s part of a package, it will be imported automatically.
6. Given how the ds list is structured, any observations with no simulated event time (returned as numeric(0)) are dropped when the rbindlist() function is called in line 265. This is presumably intentional, but just want to flag it in case.
   1. Thank you for mentioning this. Yes, this is intentional, and replicates the behavior of the original function Jack developed.