**Notes on the documentation and code requests**

Here are a few recommendations to change the appearance of the function.

1. First, I think the argument names should be simplified. See my recommendations below.
2. The argument names should map to the “keys” in the documentation (e.g., winning\_bid 🡪 winning\_bid, not price)
3. I would also like to allow the user to input initial guesses separately, or as a single vector (init\_params). The length of init\_params must equal 3 + then number of X variables. If init\_params is entered along with any other init\_, use the init\_params and display a warning that other init\_ values will be ignored.
4. The documentation should specifically mention the type of model that will be estimated. For example, if you look up “?dgamma”, you can see that the equation for the density is listed. See below for my suggested start to the text.
5. I think it would be nice to provide a helper function that provides the likelihood for a given parameter vector. This will also help the user diagnose how long the estimation might take.

# Main function: estimate auction model

auction\_model(dat = NULL,

winning\_bid = NULL,

n\_bids = NULL,

init\_mu = NULL,

init\_alpha = NULL,

init\_sigma = NULL,

init\_beta = NULL,

init\_params = NULL,

u\_dist = NULL,

num\_cores = 1)

# User can input either (init\_mu, init\_alpha, init\_sigma, init\_beta) or

# init\_params

# Print/evaluate likelihood

auction\_model\_likelihood(dat = NULL,

winning\_bid = NULL,

n\_bids = NULL,

mu = NULL,

alpha = NULL,

sigma = NULL,

beta = NULL,

params = NULL,

u\_dist = NULL,

num\_cores = 1)

**Sketch of documentation language:**

This function estimates a first-price auction model with conditional independent private values. The model allows for unobserved heterogeneity that is common to all bidders in addition to observable heterogeneity. The winning bid (Y) takes the form

Y = B \* U \* h(X)

where B Is the proportional winning bid, U is the unobserved heterogeneity, and h(X) controls for observed heterogeneity. The model is log-linear so that log(Y) = log(B) + log(U) + log(h(X)) and log(h(X)) = beta1 \* X1 + beta2 \* X2 + … .

The (conditionally) independent private costs are drawn from a Weibull distribution with parameters mu and alpha. The CDF of this distribution is given by

F(c) = 1 – exp(- (c \* 1/mu \* Gamma(1 + 1/alpha))^(alpha))

The unobserved heterogeneity can take on several different distributions, which are selected by the argument u\_dist. It is normalized to have a mean of 1, with a free parameter sigma representing the standard deviation.

# Print/evaluate likelihood

auction\_generate\_data(obs = NULL,

n\_bids = NULL,

mu = NULL,

alpha = NULL,

sigma = NULL,

beta = NULL,

params = NULL,

u\_dist = NULL,

x\_vars = NULL,

new\_x\_meanlog = NULL,

new\_x\_sdlog = NULL)

* Function returns a data.frame with (winning\_bid, n\_bids, x1, x2, x3,…)
* User must specify (**mu, alpha, sigma, beta**) or (**params**)
* User must specify **obs** (the number of observations)
* **n\_bids** may be a vector of number of bids. If specified, the length must be equal to **obs**. If not specified, they are drawn with replacement [sample(2:10, obs, replace=TRUE) ]
* the length of **beta** must be equal to the number of columns in **x\_vars** plus the length of **new\_x\_meanlog**
* If **u\_dist** is not specified, use rlnorm
* **x\_vars** is a data.frame of control variables. The number of observations must be equal to **obs.**
* **new\_x\_meanlog**, if specified, generates a set of new x variables from a lognormal distribution. Each variable has a mean equal to the corresponding entry in the new\_x\_meanlog vector. For example, new\_x\_meanlog = c(1, 2, 4) will generate three x variables.
* **new\_x\_sdlog**, if specified, provides the standard deviations for the new x variables. It must match the length of new\_x\_meanlog if specified. If not specified, use new\_x\_sdlog = rep(1, length(new\_x\_meanlog))

Code sketch for auction\_generate\_data

# Generate x\_vars to pass to function [Will be outside of function!]

w = rlnorm(obs)

x1 = rlnorm(obs) + .5\*w

x2 = .1\*rlnorm(obs) + .3\*w

x\_vars = data.frame(x1, x2)

# Code sketch to use to build auction\_generate\_data function

obs = 200

mu = 5

alpha = 2

sigma = .5

beta = c(.3, .2, .1, .4, .5)

new\_x\_meanlog = c(2, 1, .5)

new\_x\_sdlog = c(1, 1, 1)

# Number of bids

v.n = sample(2:10, obs, replace=TRUE)

# Winning bids

v.w\_bid = rep(NA, obs)

v.w\_cost = rep(NA, obs)

for(i in 1:obs){

costs = (mu/gamma(1+1/alpha))\*(-log(1-runif(v.n[i])))^(1/alpha)

min\_cost = min(costs)

v.w\_cost[i] = min(costs)

v.w\_bid[i] = f.bid\_function(cost=min\_cost, num\_bids=v.n[i], mu=mu, alpha=alpha)

}

stopifnot(mean(is.na(v.w\_bid)) == 0)

# Unobserved heterogeneity

sdlog = sqrt(log(sigma^2+1))

v.u = rlnorm(n = obs, meanlog = -1/2\*sdlog^2, sdlog = sdlog )

# Observed heterogeneity

new\_x\_vars = matrix(NA, obs, length(new\_x\_meanlog))

for(i.new\_x in 1:length(new\_x\_meanlog)){

new\_x\_vars[, i.new\_x] = rlnorm(obs,

meanlog = new\_x\_meanlog[i.new\_x],

sdlog = new\_x\_sdlog[i.new\_x])

}

all\_x\_vars = data.frame(x\_vars, new\_x\_vars)

v.h\_x = exp(colSums(beta\*t(log(all\_x\_vars))))

v.winning\_bid = v.w\_bid\*v.u\*v.h\_x

dat = data.frame(winning\_bid = v.winning\_bid, n\_bids = v.n, all\_x\_vars)

return(dat)