Polished

Taqi

```r
# Shifts spectrum by 2*Sqrt(N) (So that is centered about 0)
# Note: make spectrum a probability distribution by figuring out how to stat count normalize by entries
normalize_spectrum <- function(spectrum, N){
  spectrum$Re <- (spectrum$Re - 2*sqrt(N))*(sqrt(2)*(N)^(1/6))
  spectrum
}
# Labels a spectrum by adding a value of a given label.
# Ex. Beta = 2, p = 0.35
label_spectrum <- function(spectrum, label, value){
  spectrum[[5]] <- rep(value, nrow(spectrum))
  colnames(spectrum)[5] <- label
  spectrum
}
```

```r
spectrum.histogram2 <- function(array, ..., component = NA, label, bins = 100, mat_str = ""){
  # Process spectrum of the matrix/ensemble
  if(class(array) == "list" || class(array) == "matrix"){array_spectrum <- spectrum(array, ...)}
  else{array_spectrum <- array} # Else, the array is a precomputed spectrum (avoid computational waste
  # Infer plot title string from which type of array (matrix/ensemble)
  title_str <- .plot_title(class(array), prefix = "Spectrum", mat_str)
  # Plot parameters
  color0 <- "mediumpurple3"
  num_entries <- nrow(array) # Get number of entries to normalize
  if(class(component) == "logical"){component <- c("Re", "Im")} # Set default to both components
  # Plot lambda function
  component_plot <- function(component){
    # Plot parameters
    component_str <- paste(" (",component,")",collapse = "")
    # Plot
    array_spectrum %>%
      ggplot(mapping = aes_string(x = component)) +
      geom_histogram(mapping = aes(fill = {{ label }}), bins = bins) +
      labs(x = component, y = "Frequency", title = paste(title_str, component_str, sep = ""))
  }
  # Get list of plots
  plots <- purrr::map(component, component_plot)
  # If we have both components and patchwork is loaded, attach plots to each other
  if(length(plots) == 2){plots[[1]] / plots[[2]]} else if(length(plots) == 1){plots[[1]]}
  # Return the list of plots
  else{plots}
}
```

```r
beta_ens <- RME_beta(N = 20, beta = 2, size = 1000)
spec <- beta_ens %>% spectrum(order = 1)
```

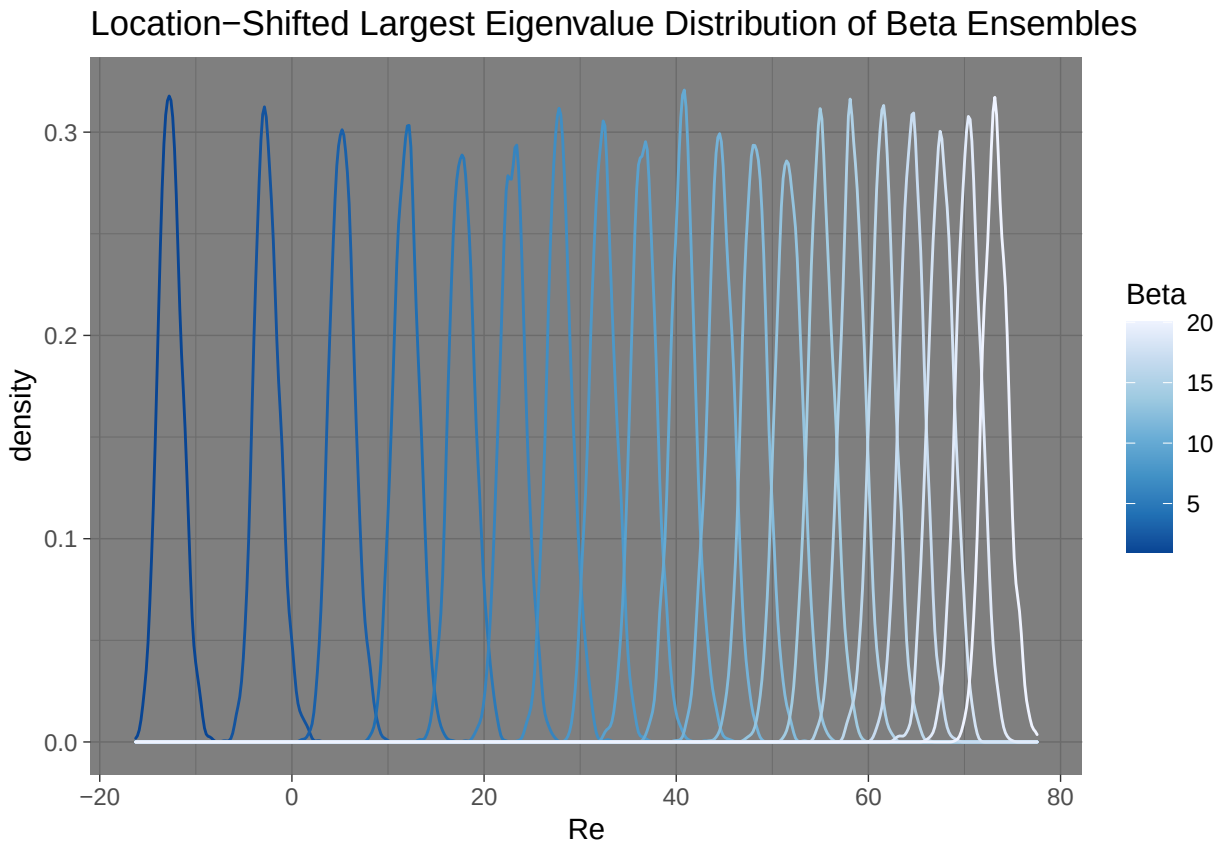
```r
# Computes the tidy, renormalized spectrum of a beta distribution object
spectrum_beta <- function(array, beta, normalize = T){
  if(class(array) == "matrix"){P <- array} else{P <- array[[1]]}
  N <- nrow(P) # Get N
  spec <- array %>% spectrum(order = 1) # Evaluate the spectrum of matrix (ensemble)
  # Normalize by shifting the left by 2*sqrt(N) and label it
  if(normalize){spec <- spec %>% normalize_spectrum(N)}
  spec <- spec %>% label_spectrum(label = "Beta", value = beta)
  spec
}
```

```r
.spectrum_beta <- function(beta, N, size, normalize = T){
  RME_beta(N, beta, size) %>% spectrum_beta(beta, normalize)
}
```

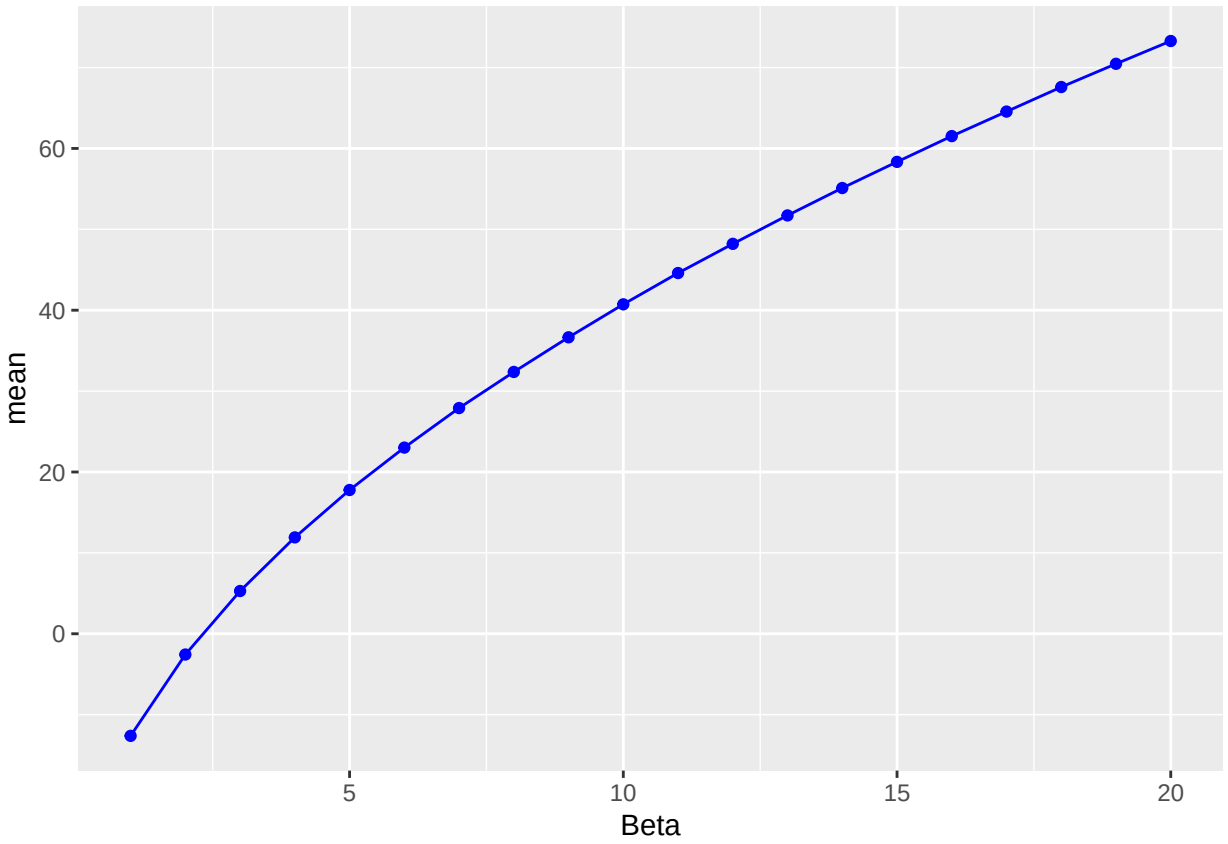
Step-wise Sequence of Beta Ensemble Largest Eigenvalue Distributions

```r
N <- 50
size <- 2500
beta_vec <- seq(1, 20, by = 1)
super_beta_ens <- purrr::map_dfr(beta_vec, .spectrum_beta, N = N, size = size)
```

```r
super_beta_ens %>%
  ggplot(aes(x = Re, group = Beta, color = Beta)) +
  geom_density() +
  scale_color_distiller(palette = "Blues") +
  theme_dark() +
  labs(title = "Location-Shifted Largest Eigenvalue Distribution of Beta Ensembles")
```



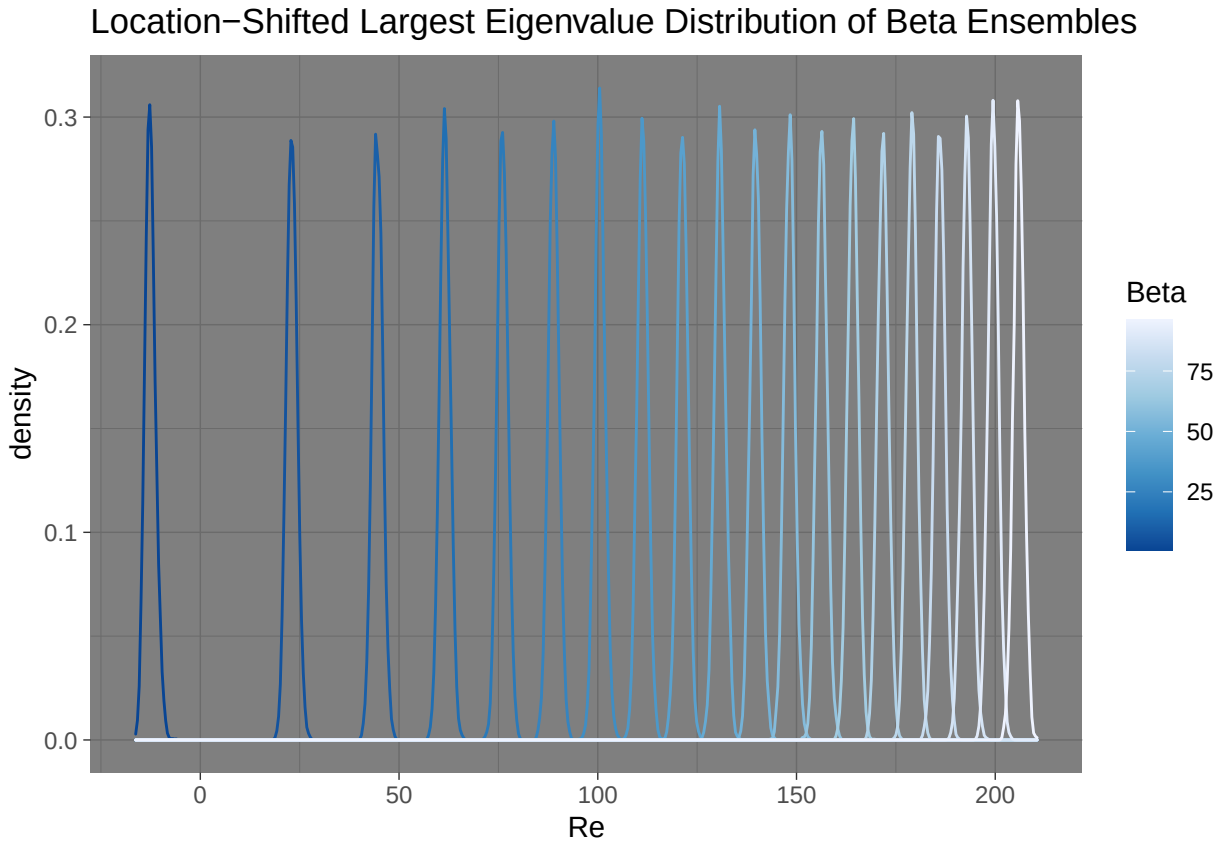

```
super_beta_ens %>%
  group_by(Beta) %>%
  summarize(mean = mean(Re), var = var(Re)) %>%
  ggplot(aes(x = Beta, y = mean)) +
  geom_point(color = "blue") +
  geom_line(color = "blue")
```



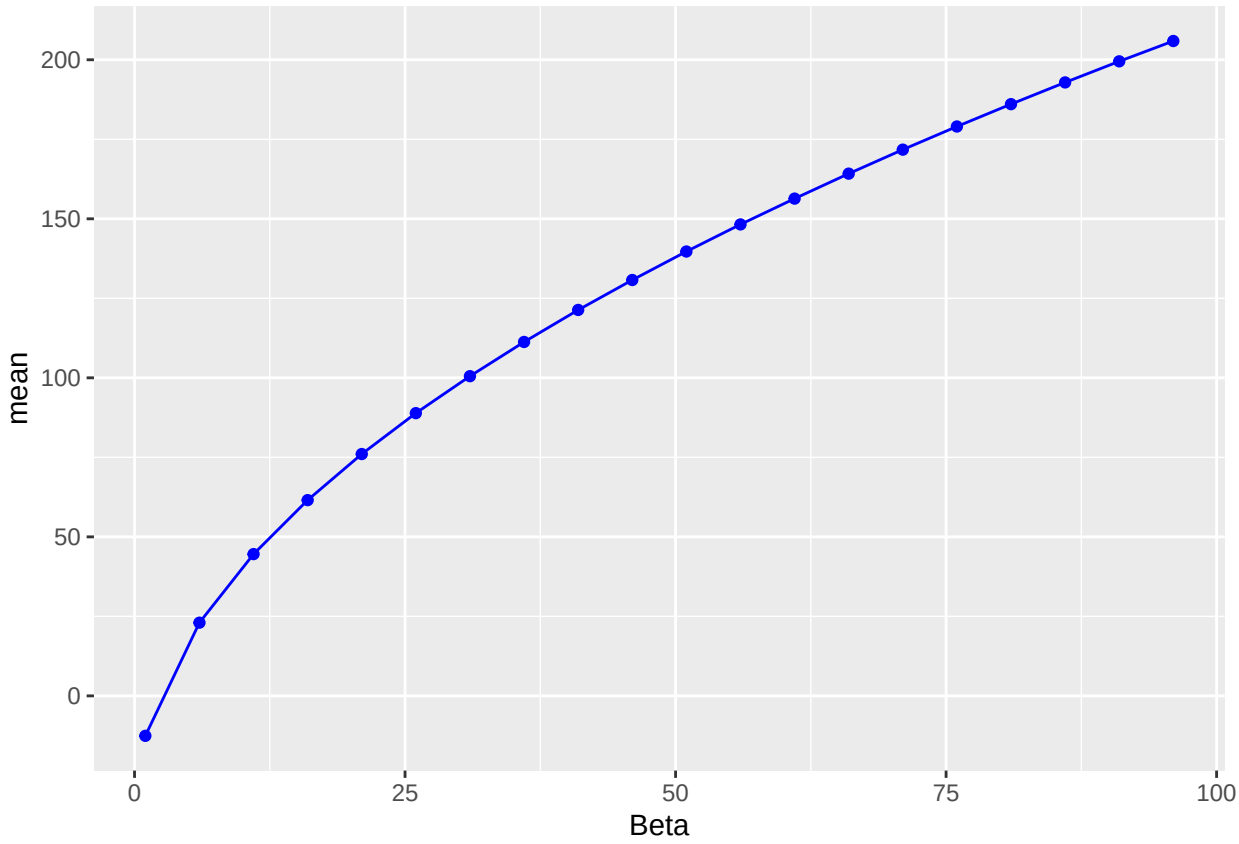
Leaping Sequence of Beta Ensemble Largest Eigenvalue Distributions

```
N <- 50
size <- 2500
beta_vec <- seq(1, 100, by = 5)
super_beta_ens <- purrr::map_dfr(beta_vec, .spectrum_beta, N = N, size = size)
```

```
super_beta_ens %>%
  ggplot(aes(x = Re, group = Beta, color = Beta)) +
  geom_density() +
  scale_color_distiller(palette = "Blues") +
  theme_dark() +
  labs(title = "Location-Shifted Largest Eigenvalue Distribution of Beta Ensembles")
```




```
super_beta_ens %>%
  group_by(Beta) %>%
  summarize(mean = mean(Re), var = var(Re)) %>%
  ggplot(aes(x = Beta, y = mean)) +
  geom_point(color = "blue") +
  geom_line(color = "blue")
```


200
150
100
50
0
mean
0
25
50
75
100
Beta