Integrating Proposed Changes into the `subscreenfunnel_fix` Function

July 14, 2025

Here's how to integrate the proposed changes into your `subscreenfunnel_fix` function:

1 Step 1: Replace the Current Wrong Approach

Find this section in your function:

```
# REMOVE THIS SECTION:
if (stratified) {
  trts <- unique(as.data.frame(data)[,treat])
  lowest_nr_subject_by_trt <- min(nrow(data[data[treat] == trts[1],]),
     nrow(data[data[treat] == trts[2],]))
  sampsize <- ifelse(
    sampsize > lowest_nr_subject_by_trt,
    lowest_nr_subject_by_trt,
    sampsize
  )
  all_samples <- replicate(nperm, weightedSampler(data_trimmed, treat,
     sampsize))
} else if (!stratified) {
  all_samples <- replicate(nperm, dplyr::slice_sample(data_trimmed, n =
      sampsize))
}
```

2 Step 2: Add the Corrected Approach

Replace with this:

```
# CORRECT: Permute treatment assignments within subgroups
generate_permuted_data <- function(data, treat, nperm) {
  lapply(1:nperm, function(i) {
    permuted_data <- data
    permuted_data[, treat] <- sample(data[, treat])  # Permute
       treatment labels
    return(permuted_data)
  })
}

# Generate permuted datasets
all_samples <- generate_permuted_data(data_trimmed, treat, nperm)
```

3 Step 3: Update the Sampling Logic

Also update the section that processes the samples:

```
# OLD CODE:
tmp <- furrr::future_map2(rep(1:nperm,length(nsamp)), rep(nsamp, each =
    nperm), function(x, y) {
  tmp1 <- data.frame(eval_function(data.frame(all_samples[,x])[1:y,]))
  tmp1$n <- y
  tmp1
}, .options=furrr::furrr_options(seed = TRUE))

# NEW CODE:
tmp <- furrr::future_map2(rep(1:nperm,length(nsamp)), rep(nsamp, each =
    nperm), function(x, y) {
  # Sample y subjects from the x-th permuted dataset
  sampled_data <- dplyr::slice_sample(all_samples[[x]], n = y)
  tmp1 <- data.frame(eval_function(sampled_data))
  tmp1$n <- y
  tmp1
}, .options=furrr::furrr_options(seed = TRUE))
```

4 Complete Fixed Function

Here's your complete function with all the corrections:

```
subscreenfunnel_fix <- function(
  data,
  H,
  eval_function,
  min_start = 2,
  n_support_points = 50,
  nperm = 200,
  alpha = 0.001,
  stratified = TRUE,
  treat = "treat",
  endpoints = NULL,
  verbose = TRUE,
  nkernel = 1,
  ...
) {

  #### WARNING & ERROR MESSAGES ####
  if (!is.logical(verbose) || is.na(verbose)) {
    stop("parameter verbose needs to be logical (TRUE/FALSE) and non-
        missing!")
  }
  if (verbose == TRUE) {
    cat("\n",
      "subscreenfunnel started at ",
      format(Sys.time(),
        format = "%F %R %Z"
      )
```

```
  )
  pt1 <- Sys.time()
}

sampsize <- H$results_total$N.of.subjects

# Generate vector of support points between min and max sample sizes
sqrtvec <- seq(
  sqrt(min_start),
  by = (sqrt(sampsize) - sqrt(min_start))/n_support_points,
  length.out = (n_support_points+1)
)
nsamp <- matrix(round(sqrtvec^2), nrow = 1)

# First we only remove covariates, since they are not of interest,
data_trimmed <- data[,c(treat, endpoints)]

# CORRECTED: Permute treatment assignments within subgroups
generate_permuted_data <- function(data, treat, nperm) {
  lapply(1:nperm, function(i) {
    permuted_data <- data
    permuted_data[, treat] <- sample(data[, treat])  # Permute
       treatment labels
    return(permuted_data)
  })
}

# Generate permuted datasets (replaces the old all_samples logic)
all_samples <- generate_permuted_data(data_trimmed, treat, nperm)

future::plan("multisession", workers = nkernel)

# CORRECTED: Sample from permuted datasets
tmp <- furrr::future_map2(rep(1:nperm,length(nsamp)), rep(nsamp, each
    = nperm), function(x, y) {
  # Sample y subjects from the x-th permuted dataset
  sampled_data <- dplyr::slice_sample(all_samples[[x]], n = y)
  tmp1 <- data.frame(eval_function(sampled_data))
  tmp1$n <- y
  tmp1
}, .options=furrr::furrr_options(seed = TRUE))

# FIXED: Replace rbind.fill with dplyr::bind_rows
tmp2 <- dplyr::bind_rows(tmp)

tmp3 <- tmp2 %>%
  dplyr::as_tibble() %>%
  dplyr::group_by(n) %>%
  dplyr::summarise_all(~quantile(., probs = c(alpha/2, 1-(alpha/2)),
     na.rm = TRUE)) %>%
  dplyr::mutate(alpha = c(alpha/2, 1-(alpha/2))) %>%
  dplyr::ungroup()

H$funnel_quantiles <- data.frame(tmp3)
```

```
  pt2 <- Sys.time()
  if (verbose == TRUE) {
    cat("\n", "Time for funnel calculation(s): ", pt2 - pt1)
  }
  return(H)
}
```

5 Key Changes Made:

- **Removed** the problematic `replicate(nperm, slice_sample(...))` approach
- **Added** the `generate_permuted_data()` function that properly permutes treatment assignments
- **Updated** the sampling logic to work with the list of permuted datasets
- **Fixed** the `rbind.fill` issue with `dplyr::bind_rows`
- **Updated** the deprecated `dplyr::funs()` to the modern syntax

6 What This Achieves:

- **Proper null hypothesis simulation**: Treatment assignments are permuted while maintaining the original data structure
- **Correct funnel boundaries**: The quantiles now represent the true null distribution
- **Better statistical validity**: Subgroup effects are tested against the appropriate null distribution

The funnel plot will now correctly represent what you'd expect to see under the null hypothesis of no treatment effect.