Comparison of Sampling Approaches

July 15, 2025

Here's a detailed comparison showing the code differences between the two approaches:

1 Key Differences

1.1 1. Sampling Strategy (Major Change)

First Approach:

```r
weightedSampler <- function(dat, treat = 'T', size = 10) {
  sizePerTrt <- round(size/length(trts))
  trt1 <- slice_sample(dat[dat[,treat]== trts[1],] ,n =sizePerTrt)
  trt2 <- slice_sample(dat[dat[,treat]== trts[2],] ,n =sizePerTrt)
  samp <- data.frame(matrix(ncol = ncol(dat), nrow = size))
  colnames(samp) <- colnames(dat)
  samp[2*(1:sizePerTrt),] <- trt1
  samp[(2*(1:sizePerTrt)-1),] <- trt2
  samp
}

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, slice_sample(data_trimmed, n =
      sampsize))
}
```

Second Approach:

```r
# CORRECTED: Permute treatment assignments
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)
```

1.2 2. Data Processing and Indexing

First Approach:

```
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))
```

Second Approach:

```
# CORRECTED: Create proper index vectors and access list correctly
perm_indices <- rep(1:nperm, length(nsamp))
sample_sizes <- rep(as.vector(nsamp), each = nperm)

tmp <- furrr::future_map2(perm_indices, sample_sizes, function(perm_idx
    , samp_size) {
  # Access the perm_idx-th permuted dataset and sample samp_size rows
  sampled_data <- dplyr::slice_sample(all_samples[[perm_idx]], n = samp
      _size)
  tmp1 <- data.frame(eval_function(sampled_data))
  tmp1$n <- samp_size
  tmp1
}, .options=furrr::furrr_options(seed = TRUE))
```

1.3 3. Data Binding Method

First Approach:

```
tmp2 <- do.call("rbind.fill", tmp)
```

Second Approach:

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

1.4 4. Summarization Function

First Approach:

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

Second Approach:

```
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()
```

2 Summary of Changes

- Stratified sampling removed: The second approach eliminates the complex stratified sampling logic.
- Treatment permutation: Instead of sampling subjects, the second approach permutes treatment assignments.
- Cleaner indexing: More explicit and clear indexing with named variables.
- Modern dplyr syntax: Uses $\sim$ instead of deprecated `funs()` and `bind_rows()` instead of `rbind.fill()`.
- Simplified data structure: Works with a list of permuted datasets rather than a complex array structure.

The second approach appears to be a "fixed" version that addresses potential issues with the original sampling strategy and modernizes the code syntax.