phase4_balance_validation.R

abdulbasir

2025-12-09

```r
# phase 4: balance and structural validation
#
# checking for structural issues, imbalance, and independence assumptions
# in the 2x2 factorial design

# sourcing configuration and utilities
source("config.R")
source("utils.R")

# loading required libraries
suppressPackageStartupMessages({
  library(dplyr)
  library(readr)
})

print_section_header("Phase 4: Balance And Structural Validation")
```

```
##
## ============================================================================
## Phase 4: Balance And Structural Validation
## ============================================================================
```

```r
# 1. loading classified data
input_file = file.path(RESULTS_DIR, "fda_analysis_clean_classified.csv")
classified_data = load_csv(input_file)

# 2. filtering to analysis-ready sample (excluding uncertain)
analysis_data = classified_data %>%
  filter(therapeutic_area != "Uncertain")

cat(paste("Analysis-ready sample: n =", nrow(analysis_data), "\n"))
```

```
## Analysis-ready sample: n = 1038
```

```r
cat(paste("Excluded uncertain:", sum(classified_data$therapeutic_area == "Uncertain"), "\n"))
```

```
## Excluded uncertain: 297
```

```r
# 3. creating era subsets
analysis_data = analysis_data %>%
  mutate(regulatory_era = assign_regulatory_era(`Approval Year`))

era_counts = analysis_data %>%
  count(regulatory_era) %>%
  arrange(regulatory_era)

cat("Era-stratified subsets created\n")
```

```
## Era-stratified subsets created
```

```r
for (i in 1:nrow(era_counts)) {
  cat(sprintf("%15s: n = %3d\n", era_counts$regulatory_era[i], era_counts$n[i]))
}
```

```
##     Early-PDUFA: n = 336
##       Mid-PDUFA: n = 181
##     Post-FDASIA: n = 314
##       Pre-PDUFA: n = 207
```

```r
# 4. generating contingency tables by era
era_list = c("Pre-PDUFA", "Early-PDUFA", "Mid-PDUFA", "Post-FDASIA")

for (era_name in era_list) {
  era_subset = analysis_data %>% filter(regulatory_era == era_name)

  cat(paste("\n", era_name, " (n=", nrow(era_subset), "):\n", sep = ""))

  if (nrow(era_subset) > 0) {
    contingency_table = table(
      era_subset$therapeutic_area,
      era_subset$`Review Designation`
    )
    print(addmargins(contingency_table))
  } else {
    cat("  No data in this era\n")
  }
}
```

```
##
## Pre-PDUFA (n=207):
##
##            Priority Standard Sum
##   Oncology       20        5  25
##   Other          81      101 182
##   Sum           101      106 207
##
## Early-PDUFA (n=336):
##
##            Priority Priority (indication [A] only) Standard Sum
##   Oncology       32                              0       19  51
```

```
##   Other         105                                1      179 285
##   Sum           137                                1      198 336
##
## Mid-PDUFA (n=181):
##
##            Priority Priority (indication [A] only) Priority (indication [B] only) Standard Sum
##   Oncology       45                               0                             0       13  58
##   Other          60                               2                             1       60 123
##   Sum           105                               2                             1       73 181
##
## Post-FDASIA (n=314):
##
##            Priority Priority (indication [A] only) Priority (used priority review voucher) Standard
##   Oncology      124                               2                                       1       38
##   Other         114                               0                                       3       32
##   Sum           238                               2                                       4       70
##
##            Sum
##   Oncology 165
##   Other    149
##   Sum      314
```

```r
# 5. flagging low power scenarios
for (era_name in era_list) {
  era_subset = analysis_data %>% filter(regulatory_era == era_name)
  n_total = nrow(era_subset)

  # CRITICAL BUG FIX #1: simplifying review designation to priority vs standard
  review_simplified = ifelse(
    grepl("Priority", era_subset$`Review Designation`),
    "Priority",
    "Standard"
  )

  if (n_total > 0) {
    contingency = table(era_subset$therapeutic_area, review_simplified)

    # extracting cell counts safely
    onc_pri = ifelse("Oncology" %in% rownames(contingency) && "Priority" %in% colnames(contingency),
                     contingency["Oncology", "Priority"], 0)
    onc_std = ifelse("Oncology" %in% rownames(contingency) && "Standard" %in% colnames(contingency),
                     contingency["Oncology", "Standard"], 0)
    oth_pri = ifelse("Other" %in% rownames(contingency) && "Priority" %in% colnames(contingency),
                     contingency["Other", "Priority"], 0)
    oth_std = ifelse("Other" %in% rownames(contingency) && "Standard" %in% colnames(contingency),
                     contingency["Other", "Standard"], 0)

    min_cell = min(c(onc_pri, onc_std, oth_pri, oth_std)[c(onc_pri, onc_std, oth_pri, oth_std) > 0])
    if (length(min_cell) == 0) min_cell = 0

    # CRITICAL BUG FIX #2: adding factor variation check
    n_therapeutic_areas = length(unique(era_subset$therapeutic_area))
    n_review_types = length(unique(review_simplified))
```

```
    sufficient_n = n_total >= 20
    sufficient_cells = min_cell >= 5
    sufficient_factor_variation = (n_therapeutic_areas >= 2) && (n_review_types >= 2)

    feasible = sufficient_n && sufficient_cells && sufficient_factor_variation

    cat(paste("\n", era_name, ":\n", sep = ""))
    cat(paste("  Sample size: n =", n_total, "\n"))
    cat(sprintf("  Cell counts: onc-pri=%d, onc-std=%d, oth-pri=%d, oth-std=%d\n",
                onc_pri, onc_std, oth_pri, oth_std))
    cat(paste("  Min cell:", min_cell, "\n"))
    if (!feasible) {
      if (!sufficient_factor_variation) {
        cat("  Warning: insufficient factor variation for interaction test\n")
      } else {
        cat("  Warning: low power for ANOVA\n")
      }
    } else {
      cat("  Viable for ANOVA testing\n")
    }
  }
}
```

```
##
## Pre-PDUFA:
##   Sample size: n = 207
##   Cell counts: onc-pri=20, onc-std=5, oth-pri=81, oth-std=101
##   Min cell: 5
##   Viable for ANOVA testing
##
## Early-PDUFA:
##   Sample size: n = 336
##   Cell counts: onc-pri=32, onc-std=19, oth-pri=106, oth-std=179
##   Min cell: 19
##   Viable for ANOVA testing
##
## Mid-PDUFA:
##   Sample size: n = 181
##   Cell counts: onc-pri=45, onc-std=13, oth-pri=63, oth-std=60
##   Min cell: 13
##   Viable for ANOVA testing
##
## Post-FDASIA:
##   Sample size: n = 314
##   Cell counts: onc-pri=127, onc-std=38, oth-pri=117, oth-std=32
##   Min cell: 32
##   Viable for ANOVA testing
```

```
# 6. analyzing temporal clustering
era_crosstab = table(
  analysis_data$therapeutic_area,
  analysis_data$regulatory_era
)
```

```r
cat("\nTemporal clustering: therapeutic area x regulatory era\n")
```

```
##
## Temporal clustering: therapeutic area x regulatory era
```

```r
cat("Absolute counts:\n")
```

```
## Absolute counts:
```

```r
print(addmargins(era_crosstab))
```

```
##
##            Early-PDUFA Mid-PDUFA Post-FDASIA Pre-PDUFA  Sum
##   Oncology          51        58         165        25  299
##   Other            285       123         149       182  739
##   Sum              336       181         314       207 1038
```

```r
cat("\nRow percentages (% within therapeutic area):\n")
```

```
##
## Row percentages (% within therapeutic area):
```

```r
era_row_pct = prop.table(era_crosstab, margin = 1) * 100
print(round(era_row_pct, 1))
```

```
##
##            Early-PDUFA Mid-PDUFA Post-FDASIA Pre-PDUFA
##   Oncology        17.1      19.4        55.2       8.4
##   Other           38.6      16.6        20.2      24.6
```

```r
cat("\nColumn percentages (% within era):\n")
```

```
##
## Column percentages (% within era):
```

```r
era_col_pct = prop.table(era_crosstab, margin = 2) * 100
print(round(era_col_pct, 1))
```

```
##
##            Early-PDUFA Mid-PDUFA Post-FDASIA Pre-PDUFA
##   Oncology        15.2      32.0        52.5      12.1
##   Other           84.8      68.0        47.5      87.9
```

```r
# 7. creating 2x2 contingency table
# CRITICAL BUG FIX #1: simplify BEFORE creating table
review_simplified = ifelse(
  grepl("Priority", analysis_data$`Review Designation`),
```

```
  "Priority",
  "Standard"
)

crosstab_2x2 = table(
  analysis_data$therapeutic_area,
  review_simplified
)
crosstab_2x2 = addmargins(crosstab_2x2)

cat("2x2 design balance (therapeutic area x review type):\n")
```

```
## 2x2 design balance (therapeutic area x review type):
```

```
print(crosstab_2x2)
```

```
##           review_simplified
##            Priority Standard  Sum
##   Oncology      224       75  299
##   Other         367      372  739
##   Sum           591      447 1038
```

```
# 8. calculating imbalance ratios
onc_priority = crosstab_2x2["Oncology", "Priority"]
onc_standard = crosstab_2x2["Oncology", "Standard"]
other_priority = crosstab_2x2["Other", "Priority"]
other_standard = crosstab_2x2["Other", "Standard"]

ratio_onc_other_priority = ifelse(onc_priority > 0, other_priority / onc_priority, Inf)
ratio_onc_other_standard = ifelse(onc_standard > 0, other_standard / onc_standard, Inf)
ratio_priority_standard_onc = ifelse(onc_standard > 0, onc_priority / onc_standard, Inf)
ratio_priority_standard_other = ifelse(other_standard > 0, other_priority / other_standard, Inf)

cat("Therapeutic area imbalance:\n")
```

```
## Therapeutic area imbalance:
```

```
cat(sprintf("  Priority reviews: other:oncology = %.2f:1\n", ratio_onc_other_priority))
```

```
##   Priority reviews: other:oncology = 1.64:1
```

```
cat(sprintf("  Standard reviews: other:oncology = %.2f:1\n", ratio_onc_other_standard))
```

```
##   Standard reviews: other:oncology = 4.96:1
```

```
cat("\nReview type imbalance:\n")
```

```
##
## Review type imbalance:
```

```r
cat(sprintf("  Oncology drugs: priority:standard = %.2f:1\n", ratio_priority_standard_onc))
```

```
##   Oncology drugs: priority:standard = 2.99:1
```

```r
cat(sprintf("  Other drugs: priority:standard = %.2f:1\n", ratio_priority_standard_other))
```

```
##   Other drugs: priority:standard = 0.99:1
```

```r
# 9. analyzing expedited program overlap
review_simplified = ifelse(
  grepl("Priority", analysis_data$`Review Designation`),
  "Priority",
  "Standard"
)

expedited_programs = c(
  "Orphan Drug Designation",
  "Accelerated Approval",
  "Breakthrough Therapy Designation",
  "Fast Track Designation"
)

for (program in expedited_programs) {
  if (program %in% names(analysis_data)) {
    program_binary = ifelse(
      analysis_data[[program]] == "Yes" | analysis_data[[program]] == "yes",
      "Yes",
      "No"
    )

    overlap_crosstab = prop.table(
      table(review_simplified, program_binary),
      margin = 1
    ) * 100

    cat(paste("\n", program, ":\n", sep = ""))
    print(round(overlap_crosstab, 1))
  }
}
```

```
##
## Orphan Drug Designation:
##                   program_binary
## review_simplified   No  Yes
##          Priority 40.4 59.6
##          Standard 80.8 19.2
##
## Accelerated Approval:
##                   program_binary
## review_simplified   No  Yes
##          Priority 78.0 22.0
##          Standard 98.4  1.6
```

```
##
## Breakthrough Therapy Designation:
##                   program_binary
## review_simplified   No  Yes
##          Priority 80.4 19.6
##          Standard 98.2  1.8
##
## Fast Track Designation:
##                   program_binary
## review_simplified   No  Yes
##          Priority 67.0 33.0
##          Standard 93.7  6.3
```

```
# 10. checking company clustering
if ("Applicant" %in% names(analysis_data)) {
  applicant_counts = table(analysis_data$Applicant)
  total_companies = length(applicant_counts)
  total_approvals = nrow(analysis_data)
  companies_with_multiple = sum(applicant_counts > 1)
  max_approvals = max(applicant_counts)
  max_company = names(applicant_counts)[which.max(applicant_counts)]

  threshold = total_approvals * INDEPENDENCE_THRESHOLD
  major_companies = applicant_counts[applicant_counts > threshold]

  cat(paste("Total pharmaceutical companies:", total_companies, "\n"))
  cat(paste("Total drug approvals:", total_approvals, "\n"))
  cat(sprintf("Companies with multiple approvals: %d (%.1f%%)\n",
              companies_with_multiple,
              100 * companies_with_multiple / total_companies))
  cat(sprintf("Maximum approvals per company: %d (%s)\n", max_approvals, max_company))
  cat(sprintf("Average approvals per company: %.2f\n", total_approvals / total_companies))

  if (length(major_companies) > 0) {
    cat(sprintf("\nCompanies with >%.0f%% of sample:\n", INDEPENDENCE_THRESHOLD * 100))
    for (i in 1:length(major_companies)) {
      company = names(major_companies)[i]
      count = major_companies[i]
      cat(sprintf("  %s: %d approvals (%.2f%%)\n",
                  company, count, 100 * count / total_approvals))
    }
  } else {
    cat(sprintf("\nNo company exceeds %.0f%% threshold\n", INDEPENDENCE_THRESHOLD * 100))
    cat("Independence assumption reasonably satisfied\n")
  }
} else {
  cat("Applicant column not found - skipping company clustering analysis\n")
}
```

```
## Total pharmaceutical companies: 514
## Total drug approvals: 1038
## Companies with multiple approvals: 144 (28.0%)
## Maximum approvals per company: 28 (Pfizer)
## Average approvals per company: 2.02
```

```
##
## No company exceeds 5% threshold
## Independence assumption reasonably satisfied
```

```
# 11. saving results
results_data = data.frame(
  metric = c(
    "onc_priority", "onc_standard", "other_priority", "other_standard",
    "ratio_onc_other_priority", "ratio_onc_other_standard",
    "ratio_priority_standard_onc", "ratio_priority_standard_other"
  ),
  value = c(
    onc_priority, onc_standard, other_priority, other_standard,
    ratio_onc_other_priority, ratio_onc_other_standard,
    ratio_priority_standard_onc, ratio_priority_standard_other
  )
)

output_file = file.path(RESULTS_DIR, "balance_report.csv")
save_csv(results_data, output_file)
```

```
## saving results to: /Users/abdulbasir/Downloads/Experimental AI/fda-oncology-approval-analysis/results
```

```
cat("\nPhase 4 complete\n")
```

```
##
## Phase 4 complete
```