rpact Cheatsheet

Design → **Plan** → **Analyze** · From trial rules and planning assumptions to observed data

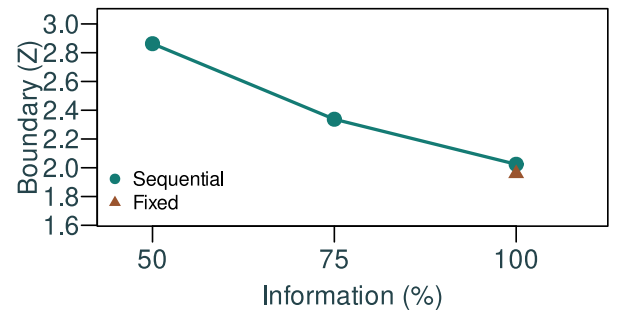
Choose a design

Fixed · One planned analysis. **Group sequential** · Multiple planned analyses with stage-specific boundaries.

```
library(rpact)

fixedDesign <- getDesignFixed(
    alpha = 0.025, beta = 0.20, sided = 1
)
design <- getDesignGroupSequential(
    kMax = 3,
    alpha = 0.025, beta = 0.20, sided = 1,
    informationRates = c(0.50, 0.75, 1),
    typeOfDesign = "OF"
)
```

`alpha` is the overall significance level. `1 - beta` is target power. `"OF"` selects O'Brien–Fleming boundaries. Check settings and defaults with `design |> print()`.



Plan sample size or power

How many participants? Use `getSampleSize…()`. **Power at a given size?** Use `getPower…()`. Pick the suffix by endpoint.

Endpoint	Function suffix
Continuous	`Means`
Binary	`Rates`
Time to event	`Survival`
Count data	`Counts`

Use the same assumed response rates for both designs.

```
fixedPlan <- fixedDesign |>
    getSampleSizeRates(pi1 = 0.60, pi2 =
0.45)
seqPlan <- design |>
    getSampleSizeRates(pi1 = 0.60, pi2 =
0.45)

design |>
    getPowerRates(
        pi1 = 0.60, pi2 = 0.45,
        maxNumberOfSubjects = 360
    ) |>
    summary()
```

`seqPlan$numberOfSubjects` gives **cumulative** sample sizes across both groups. Here they are approximately **176.9 / 265.4 / 353.8** before rounding to feasible stage sizes.

Analyze observed data

Supply participant and responder counts **for each stage**. Reuse the design that defines the analyses.

```
data <- getDataset(
    n1 = c(90, 45, 45),
    n2 = c(90, 45, 45),
    events1 = c(51, 26, 28),
    events2 = c(45, 19, 18)
)
result <- design |>
    getAnalysisResults(dataInput = data)
result |> summary()
```

Read the output in this order.

1. **Test action** · Continue, stop, or reject according to the design. Here the first two analyses continue and the final analysis rejects.
2. **Effect** · The final cumulative response-rate difference is approximately **12.8 percentage points**.
3. **Uncertainty** · The final adjusted confidence interval is approximately **1.3 to 22.7 percentage points**.

Work with objects · Use `print()` for values, `summary()` for key results, `plot()` for graphics, and `as.data.frame()` for tabular data where supported.

Extend the workflow

Refine rules → Compare scenarios → Report results

Refine interim rules

Alpha spending ties cumulative Type I error spending to the information available at each analysis. **Futility** defines when low interim evidence can trigger early stopping.

```
spendingDesign <- getDesignGroupSequential(
    kMax = 2, alpha = 0.025,
    beta = 0.20, sided = 1,
    informationRates = c(0.50, 1),
    typeOfDesign = "asOF",
    futilityBounds = 0,
    bindingFutility = FALSE
)
spendingDesign |> summary()
```

`"asOF"` selects O'Brien–Fleming-type alpha spending. The futility boundary is on the Z scale at the interim look.

Simulate operating characteristics for a proposed rule. Use enough replicates for the precision you need.

```
spendingDesign |>
    getSimulationRates(
        pi1 = 0.60, pi2 = 0.45,
        plannedSubjects = c(180, 360),
        maxNumberOfIterations = 1000,
        seed = 123
    ) |>
    summary()
```

Compare planning scenarios

Use the same endpoint, effect assumption, alpha and target power to compare designs. Start with a planning object and pipe it into `compare()`.

```
fixedPlan |>
    compare(seqPlan) |>
    summary()
```

For a quick side-by-side summary, use the versus operator `%vs%`:

```
fixedPlan %vs% seqPlan
```

Total participants	Fixed	Sequential
Maximum	345.6	353.8
Expected at 60% vs 45%	345.6	287.7

Planning values before rounding. Early stopping can lower the expected sample size even when the maximum is higher.

Use a named collection to build up scenarios.

```
comparison <- getComparison(fixed =
fixedPlan)
comparison |> add(sequential = seqPlan)
comparison |> fetch("sequential")
comparison |> as.data.frame()
```

Use `plot()` on the comparison to explore selected results visually.

Report in Quarto or R Markdown

With **knitr**, rpact's `knit_print()` methods automatically format supported objects as Markdown, much like `knitr::kable()` does for tables.

Put an rpact object or its summary on the last line of an executable R chunk. This example shows the code and its rendered output.

```
fixedDesign |> summary()
```

Fixed sample analysis

Fixed design, one-sided significance level 2.5%, power 80%, undefined endpoint.

Stage	Fixed
Stage level (one-sided)	0.0250
Efficacy boundary (z-value scale)	1.960

Find a plot type. Use `plotTypes(output = "numcap")` to list the available plot numbers and captions for an rpact object. Pass the number you want to `plot(type = ...)`.

```
seqPlan |> plotTypes(output = "numcap")
```

Use `as.data.frame()` to continue working with the result as tabular data, where supported.