

gemtcにおけるArm-levelとRelative effectデータの構造と関数設定

このプレゼンテーションでは、ベイズアン ネットワークメタアナリシスのためのRパッケージ gemtcで使用可能な2種類のデータ形式（Arm-levelとRelative effect）について、構造・関数設定・推定の流れを整理します。実務上の注意点や推奨される使い分けも明示します。

gemtcはPlummer M, et al: rjags, Plummer M: JAGSを使用します。

van Valkenhoef G, Kuiper j: gemtc

<https://cran.r-project.org/web/packages/gemtc/>



gemtcで扱える2種類のデータ形式

Arm-level data

各治療群の観測値を直接扱うデータ形式です。

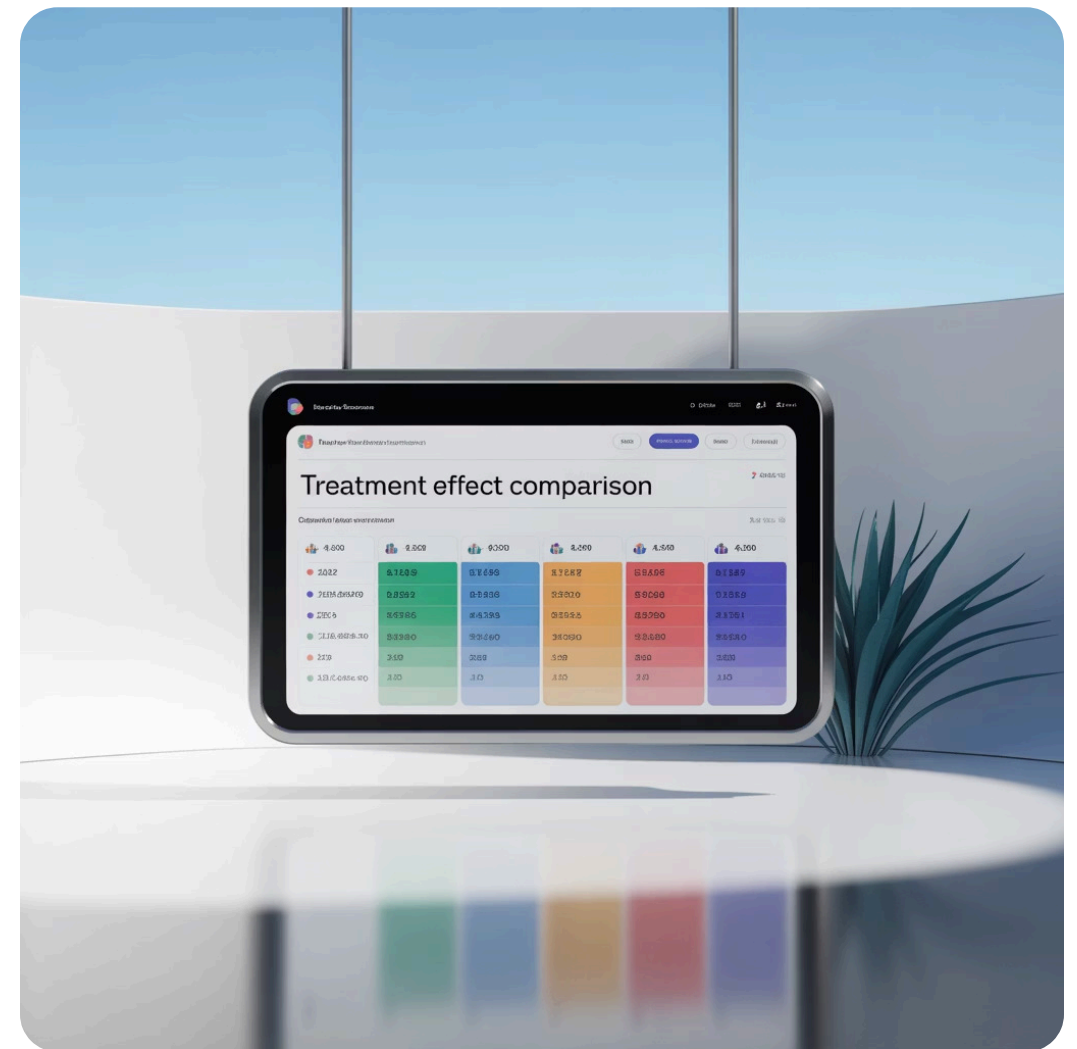
- 反応者数と母数などの原始データ
- 群ごとの結果を直接モデル化
- より詳細な情報量を保持



Relative effect data

治療間の差分と標準誤差を扱うデータ形式です。

- logORとSE、Mean DifferenceとSEなどの要約統計量
- 治療効果の差を直接モデル化
- 既存メタ解析の結果を再利用可能



これらの形式は、モデル構造・情報量・関数設定に影響します。適切な選択が分析精度を左右します。

Arm-level data：構造と関数設定

データ構造例

```
study treatment responders sampleSize
```

1	A	30	50
1	B	20	50
2	A	40	60
2	C	35	60

連続変数アウトカムの場合、study, treatment, mean, std.dev, sampleSize

関数設定

```
network <- mtc.network(data.ab = data)
model <- mtc.model(network,
  likelihood = "binomial",
  link = "logit")
result <- mtc.run(model)
```

ポイント

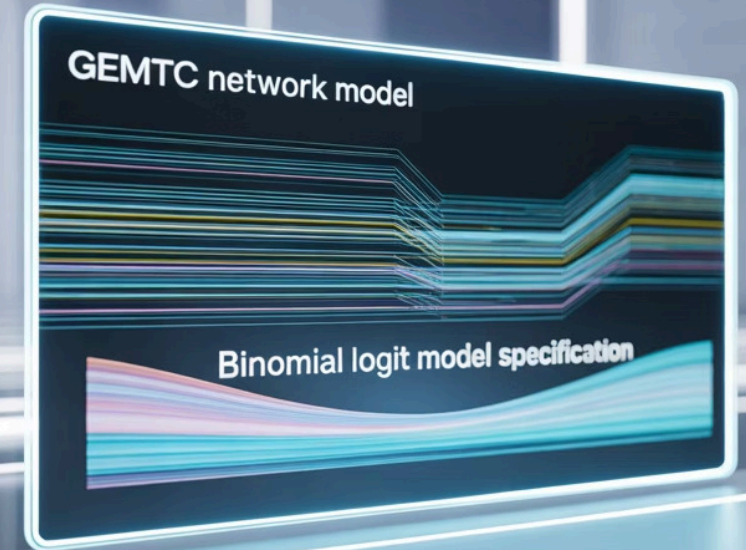
data.abパラメータで指定します

likelihoodと**link**はアウトカムに応じて選択します

- 二値データ OR：
binomial + logit, RR：
binomial + log
- 連続データ：normal + identity

SMDの場合はnormal + smd

原始データの全情報を活用できるため、より精度の高い推定が可能です



```
== <<"rsu1a1IC">
== network modek model">
==
== <<relative effect data input
== normal identity
== model speccliccation,>
==
```

Relative effect data：構造と関数設定

データ構造例

	study	treatment	diff	std.err
1	A	NA	0.07	
1	B	-0.3	0.15	
2	A	NA	0.2	
2	B	-0.4	0.12	
2	C	-0.1	0.2	

*log(HR), SE{log(HR)}

関数設定

```
network <- mtc.network(data.re = data)
model <- mtc.model(network,
likelihood = "normal",
link = "identity")
result <- mtc.run(model)
```

ポイント

data.reパラメータで指定します

要約データに適した分布とリンク関数を選択します

- 通常は**normal**と**identity**の組み合わせ
- 効果指標の特性に注意が必要

論文等から抽出した要約統計量を直接活用できます

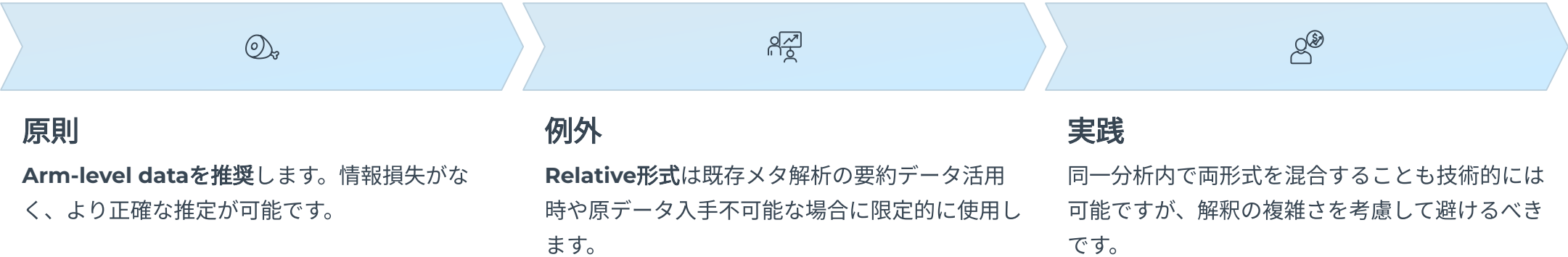
各研究内で1行目が対照アームとなり、diffはNA、std.errは対照アームの標準誤差 $SD/\sqrt{(\text{sampleSize})}$

SMDの場合はHedge's adjusted gと $1/\sqrt{(\text{sampleSize})}$

Arm-level vs Relative effect：比較と推奨

項目	Arm-level	Relative effect
元データ	原始的（群ごとの観測）	要約（差分と標準誤差）
情報量	完全	一部損失
柔軟性	高い	限定的
gemtc関数の引数	data.ab	data.re
推定精度	高	相対的に低

推奨





まとめと実務上の注意点

まとめ

- データ形式に応じて関数設定が明確に分かれます
 - Arm-level: **data.ab**パラメータ
 - Relative effect: **data.re**パラメータ
- Arm-level形式は柔軟性と精度に優れ、基本的に推奨されます
- Relative形式は要約データ活用時に限定的に使用します
- 両形式ともにgemtcの基本的な分析フローは共通です

実務上の注意点

⚠ **treatment名は一貫性を保つ必要**があります（gemtc内部でノード化される）

✖ **欠損比較や非対称ネットワーク**に注意してください。ネットワークの接続性を確認しましょう。

i **mtc.network()**の出力構造を確認してから推定へ進むことで、エラーを事前に防止できます。

分析前に**plot(network)**でネットワーク構造を可視化することを推奨します。