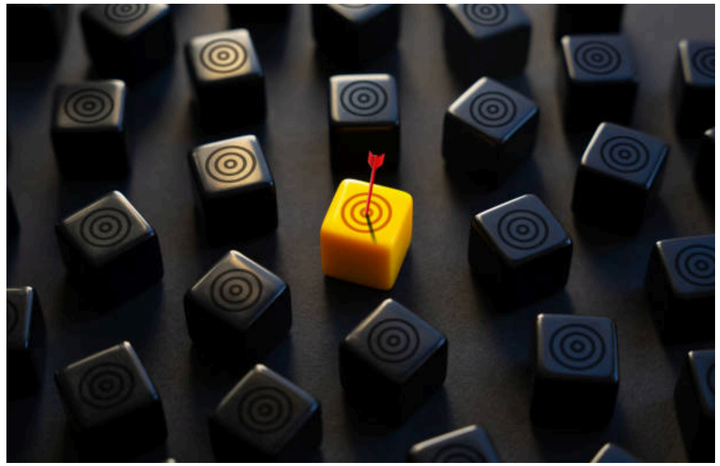
The design and development of the easyBIM package

R for HTA Workshop

28th June, 2024

Agenda

1. **Introduction to easyBIM**
2. Methods
3. Reflections and next steps



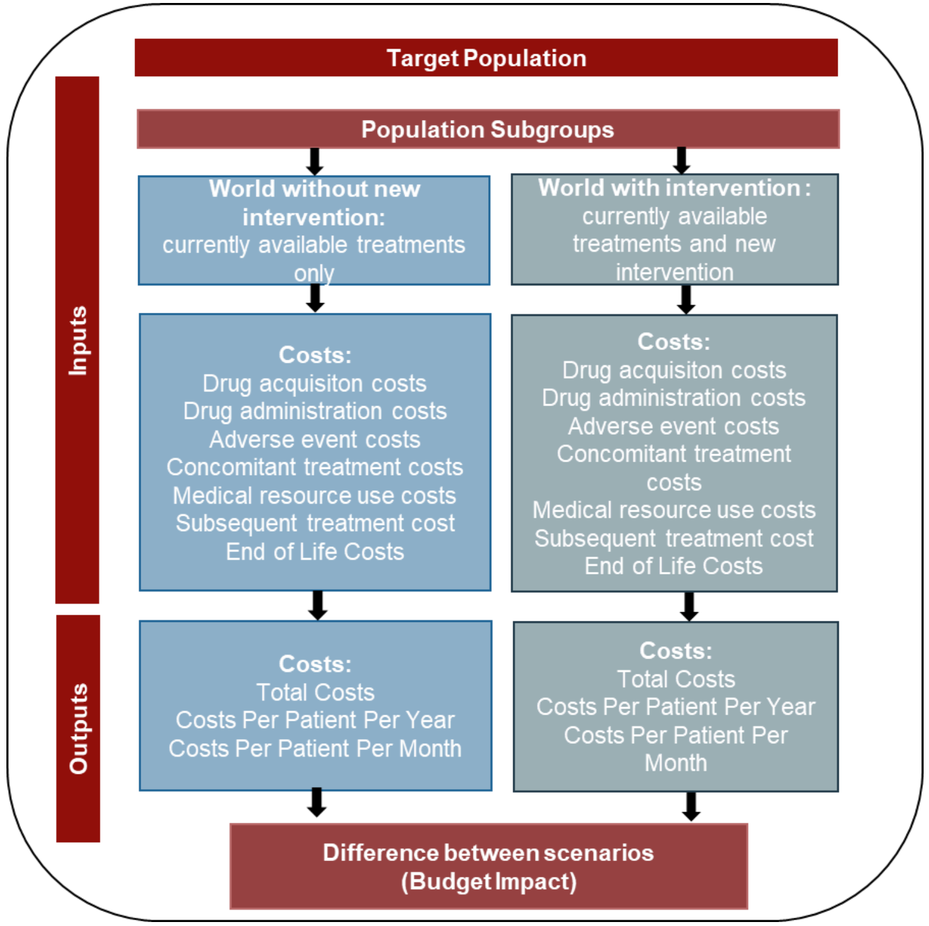
What are BIMs?

- **Budget impact analysis** estimates the financial consequences of adopting new interventions within a specific healthcare setting.
- Purpose of budget impact analysis:
  - Necessary for reimbursement
  - Communicating value to stakeholders

How are BIMs usually built?

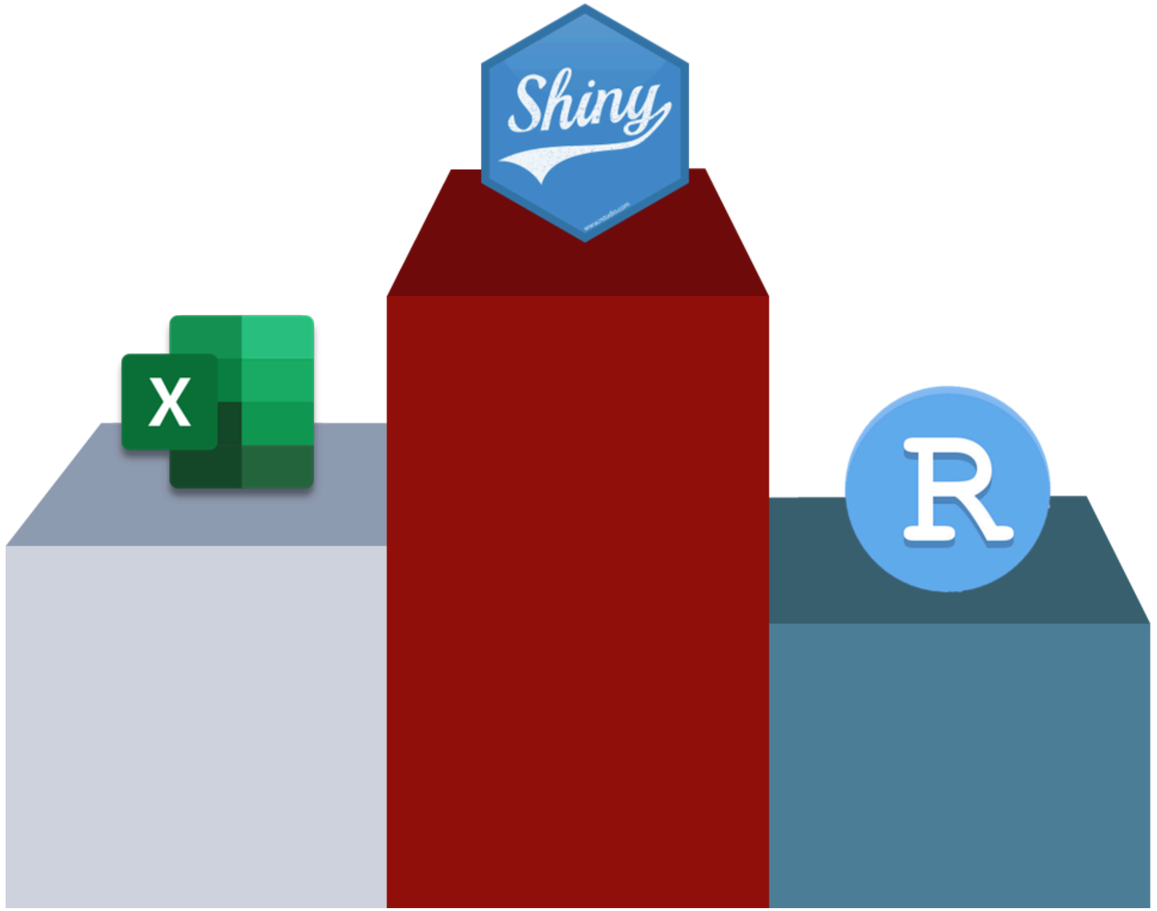
- BIMs are typically built in Excel. Either as a standalone model, or in the same workbook as the CEM
- In some cases, such as when the BIM is intended to be used as a field tool to communicate value to clinicians, an engaging and user-friendly interface can be important.

Example budget impact model structure:



Why we built easyBIM: expected use cases

easyBIM is intended to be an easy-to-use R package for health economic budget impact analysis.



A model engine for Shiny BIMs

If a BIM is built in R, it's often as a means to build a user-friendly Shiny app.

A tool to QC Excel models

If easyBIM is quick and easy to use (and covers the functionality needed to replicate the Excel model), then it may be useful as a QCing tool.

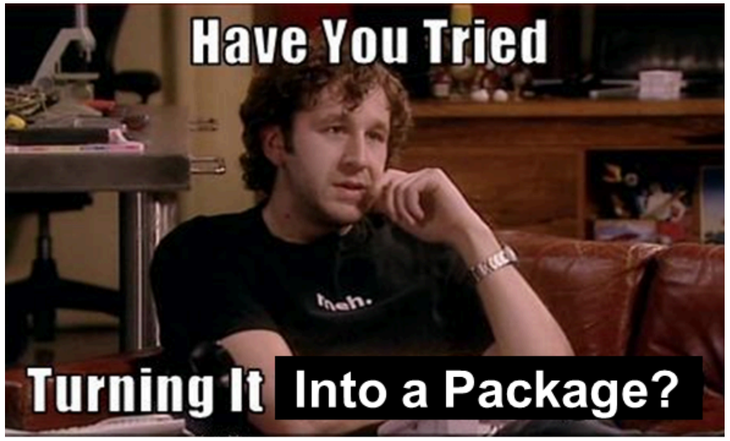
If you already have an R-based CEM

You may want to use the outputs of the CEM as inputs for the BIM.

BIMs tend to be relatively simple models, making them an ideal R learning opportunity.

BUILDING A PACKAGE

- **Building packages** was a hot topic at last year's R for HTA workshop
- One takeaway we had from last year was that if you have a well-documented, and well-structured R project, then you're already not far off from a package.
- Some of the benefits of building a package include:
  - Code Organization and Reusability
  - Collaboration and Sharing
  - Documentation



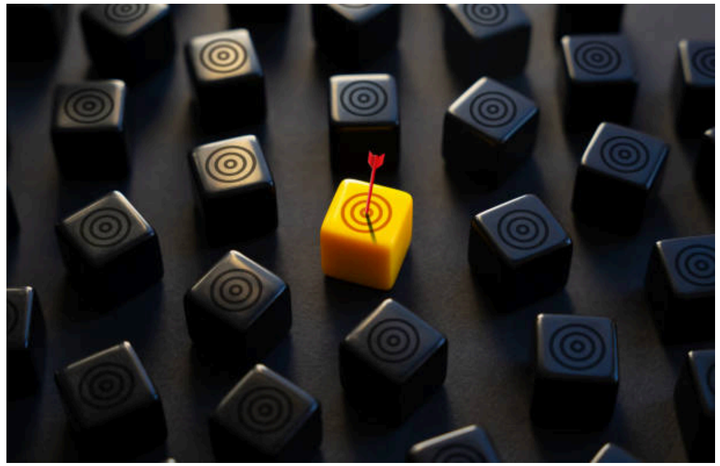
PERSONAL MOTIVATIONS

EXPLORING DIFFERENT PACKAGES

- We hoped to use this project as a way of exploring packages and systems, and to learn best-practices for developing R packages.
- Packages we experimented with included:
  - S7
  - cli (for improving the print() output for the classes in our package)

Agenda

1. Introduction to easyBIM
2. **Methods**
3. Reflections and next steps



easyBIM capabilities

Elements	Features identified in reviewed BIMs	easyBIM
Epidemiology	Patient funnel	✓
	Distinction between incident and prevalent populations. *E.g., separate dosage schedule for prevalent populations vs incident populations.*	✓
	Subgroups which determine dosage schedules and treatment eligibility.	✓
Acquisition costs	Titration	✗
	RDI/Compliance	✓
	MoM used to estimate distribution across different dose sizes	✓
	Estimate optimal (cost-minimizing) combination of vials	✓
	Dose bands and vial combination tables	✓
	Include both monotherapies and joint-therapies	✓
Administration costs	Variable costs over time horizon	✗
	Costs applied either per dose or per dispense	✓
Other costs	Monitoring, health state costs; AEs; premedication; subsequent treatment	✓
	Costs where the frequency changes across years	✗
Sensitivity analysis	OWSA; scenario analysis	✗

Building a basic BIM

- In the following section, we'll walkthrough each of the steps required to build a **simple BIM** with the following features:
  - Includes acquisition costs, administration costs, monitoring costs and treatment-related-adverse-events (TRAEs)
  - Multiple formulations for each treatment
  - Treatment specific time-on-treatment (ToT)

> **Note**
>
> - In this example we won't go through all features of easyBIM. For example, we won't cover:
>   - Vial combination optimisation
>   - Methods of moment distribution of dose size dependent on patient characteristics such as patient weight and BSA
>   - Subgroup analysis

Necessary arguments in `run_model()`

```r
# Run model
base_case <- easyBIM::run_model(
  currency = "$",
  time_horizon = 5,
  default_time_interval = "monthly",
  patient_weight = list(mean = 80),
  patient_funnel = patient_funnel_instance,
  market_shares = market_shares,
  bim_treatment_list = bim_treatment_list,
  list_admin_cost = list_admin_cost
)
```

Example BIM: Creating a simple patient funnel

```r
# Create a vector
funnel_proportions <- c("Overall Prevalence" = 0.005,
                        "Proportion diagnosed" = 0.75,
                        "Proportion adults" = 0.2143,
                        "Proportion treated" = 0.7)

# Create an instance of the patient_funnel class
patient_funnel_instance <- easyBIM::patient_funnel(
  total_population = 67000000,
  funnel_proportions = funnel_proportions,
  annual_population_growth = 0.004
)

patient_funnel_instance
```

Console output

```
── Patient funnel ──────────────────────────────────────────────────────────────

ℹ If you wish to use your own patient funnel, you can directly input the number of patients
treated by defining the optional argument `number_patients_treated` in
`easyBIM::run_model()` .

── Patient funnel ──

→ Total population size: 67,000,000

                    Proportions     Count
Overall Prevalence        0.500% 335000.00
Proportion diagnosed     75.000% 251250.00
Proportion adults        21.430%  53842.88
Proportion treated       70.000%  37690.01

── Number of patients treated in each year of time horizon: ──

  year_1   year_2   year_3   year_4   year_5
37690.01 37992.14 38144.10 38296.68 38449.87
```

- easyBIM includes functionality to build a basic patient funnel.
- In some cases, patient funnels can be considerably more complex, especially when considering subgroups and treatment lines.
- For this reason, `number_patients_treated` has been included as an optional argument in `run_model()`.
  - This enables the user to use their own patient funnel and overwrite the number of patients treated in each of year of the time horizon with their own values.

Add to `run_model()`:

```r
# Run model
base_case <- easyBIM::run_model(
  currency = "$",
  time_horizon = 5,
  default_time_interval = "monthly",
  patient_weight = list(mean = 80),
  patient_funnel = patient_funnel_instance,
  market_shares = market_shares,
  bim_treatment_list = bim_treatment_list,
  list_admin_cost = list_admin_cost
)
```

Example BIM: Market share inputs

```r
# Create a list for both scenarios
market_share_new <- list(
  year_1 = c(0.2, 0.3, 0.5),
  year_2 = c(0.25, 0.3, 0.45),
  year_3 = c(0.3, 0.3, 0.4),
  year_4 = c(0.4, 0.3, 0.4),
  year_5 = c(0.5, 0.3, 0.4)
)

market_share_ref <- list(
  year_1 = c(0, 0.35, 0.65),
  year_2 = c(0, 0.35, 0.65),
  year_3 = c(0, 0.35, 0.65),
  year_4 = c(0, 0.31, 0.65),
  year_5 = c(0, 0.31, 0.65)
)

market_shares <- list(market_share_new = market_share_new,
                      market_share_ref = market_share_ref)
```

⚠ Warning

```
! Market shares must sum to 1 for each year.
ℹ Market shares in list_market_share_new for years, "4, 5", were normalized to sum to 1.
ℹ Market shares in list_market_share_ref for years, "4, 5", were normalized to sum to 1.
→ To see the updated market shares, see market_shares (an output of `easyBIM::run_model()`)
```

- There are 3 treatments included in this example model.
- Market shares are inputted as lists: a list for the scenario with the new intervention, and a list for the scenario without the new intervention

Add to `run_model()`:

```r
# Run model
base_case <- easyBIM::run_model(
  currency = "$",
  time_horizon = 5,
  default_time_interval = "monthly",
  patient_weight = list(mean = 80),
  patient_funnel = patient_funnel_instance,
  market_shares = market_shares,
  bim_treatment_list = bim_treatment_list,
  list_admin_cost = list_admin_cost
)
```

Example BIM: Creating bim_treatment objects

- `easyBIM::bim_treatment` is an S7 class. It essentially performs like a list with some differences:
  - Custom `print()` method
  - Property classes are automatically validated
    - E.g., `bim_treatment@basis` is set to `class_character`. If the user makes the input a list, it will return an error.

bim_treatment inputs for the comparator cmp_1 + cmp_2

```
tx_1 <- bim_treatment(name = "cmp_1 + cmp_2",
                      components = c("cmp_1", "cmp_2"),
                      list_price =  list("cmp_1" = c(100, 150, 20
                                         "cmp_2" = c(15, 20, 25))
                      units_per_pack = list("cmp_1" = c(1, 1, 1),
                                            "cmp_2" = c(1, 1, 1))
                      mg_per_unit = list("cmp_1" = c(40, 60, 70),
                                         "cmp_2" = c(50, 60, 70))
                      ToT = 2.41,
                      dose_size = c(1, 3),
                      basis = c("mg/kg", "mg"),
                      admin_route = c("immuno_admin", "immuno_adm
                      frequency = c("Q3W", "Q2W")
)
```

Console output

```
── bim_treatment object: cmp_1 + cmp_2 ─────────────────────────────────────────

── Acquisition unit costs ──

ℹ Each element of the vectors below corresponds to a different formulation.

                cmp_1       cmp_2
list_price      100,150,200 15,20,25
units_per_pack  1,1,1       1,1,1
mg_per_unit     40,60,70    50,60,70

── Acquisition and administration costs summary ──

                         cmp_1                cmp_2
dose_size                1                    3
basis                    mg/kg                mg
frequency                Q3W                  Q2W
admin_route              immuno_admin         immuno_admin
time_on_treatment        2.41                 2.41
cost_per_dose            Not calculated yet.  Not calculated yet.
num_doses_per_year       Not calculated yet.  Not calculated yet.
acq_cost_per_patient     Not calculated yet.  Not calculated yet.
admin_cost_per_patient   Not calculated yet.  Not calculated yet.

── Vial combinations ──

Not calculated yet.
```

Example BIM: Creating bim_treatment objects

- `easyBIM::bim_treatment` is an S7 class. It essentially performs like a list with some differences:
  - Custom `print()` method
  - Property classes are automatically validated
    - E.g., `bim_treatment@basis` is set to `class_character`. If the user makes the input a list, it will return an error.

```
        tx_2 <- bim_treatment(name = regimen_names[2],
                              components = c("cmp_1", "cmp_2"),
                              list_price =  list("cmp_1" = c(100, 150, 20
                                                 "cmp_2" = c(15, 20, 25))
                              units_per_pack = list("cmp_1" = c(1, 1, 1),
                                                    "cmp_2" = c(1, 1, 1))
                              mg_per_unit = list("cmp_1" = c(40, 60, 70),
                                                 "cmp_2" = c(50, 60, 70))
                              ToT = 2.41,
                              dose_size = c(1, 3),
                              admin_route = c("immuno_admin", "immuno_adm
                              basis = c("mg/kg", "mg"),
                              frequency = c("Q3W", "Q2W")
        )
```

- Warnings and error catches like the below help with transparency and show the user additional functionality to consider using.

> ⚠ **Warning**
>
> ```
> ! The same @ToT (2.41) is assumed to apply to all treatments in regimen cmp_1 + cmp_2.
> → For treatment specific inputs, include a @ToT input for each treatment in the regimen.
> ```

Add to `run_model()`:

```
        # Run model
        base_case <- easyBIM::run_model(
          currency = "$",
          time_horizon = 5,
          default_time_interval = "monthly",
          patient_weight = list(mean = 80),
          patient_funnel = patient_funnel_instance,
          market_shares = market_shares,
          bim_treatment_list = bim_treatment_list,
          list_admin_cost = list_admin_cost
        )
```

Example BIM: Administration costs

Administration cost inputs

```
list_admin_cost <- list(
  immuno_admin = c(306.90, "per_dose"),
  oral = c(24, "per_dispense")
)
```

- Administration costs
  - Administration costs can be applied each dose (**per_dose**), or per dispense (**per_dispense**), which is set at once per month.
  - Next steps: allow for a user-defined frequency.

Add to `run_model()`:

```
# Run model
base_case <- easyBIM::run_model(
  currency = "$",
  time_horizon = 5,
  default_time_interval = "monthly",
  patient_weight = list(mean = 80),
  patient_funnel = patient_funnel_instance,
  market_shares = market_shares,
  bim_treatment_list = bim_treatment_list,
  list_admin_cost = list_admin_cost
)
```

Example BIM: Results

print(base_case)

Summary of inputs

- Acquisition cost inputs
- Dosage schedule inputs
- Intermedirary outputs (e.g. cost per dose)

Results

Plots

Console output

```
! The same @ToT (2.41) is assumed to apply to all treatments in regimen cmp_1 + cmp_2.
→ For treatment specific inputs, include a @ToT input for each treatment in the regimen.

! Market shares must sum to 1 for each year.
ℹ Market shares in list_market_share_new for years, "4, 5", were normalized to sum to 1.
ℹ Market shares in list_market_share_ref for years, "4, 5", were normalized to sum to 1.
→ To see the updated market shares, see market_shares (an output of `easyBIM::run_model()`)

── Model Inputs ──────────────────────────────────────────────────────────────────────────────────

── Acquisition Cost Inputs: Pack Costs test ──

   regimen_name treatments  list_price units_per_pack mg_per_unit
1     Product Y  Product Y           5              1        2000
2 cmp_1 + cmp_2      cmp_1 100,150,200          1,1,1    40,60,70
3 cmp_1 + cmp_2      cmp_2    15,20,25          1,1,1    50,60,70
4          cmp_3      cmp_3        2633              1         240

── Dosage schedule inputs: ──

   regimen_name components dose_size basis frequency  ToT  admin_route vial_sharing
1     Product Y  Product Y        50 mg/kg       Q1W 4.50 immuno_admin         TRUE
2 cmp_1 + cmp_2      cmp_1         1 mg/kg       Q3W 2.41 immuno_admin         TRUE
3 cmp_1 + cmp_2      cmp_2         3    mg       Q2W 2.41 immuno_admin         TRUE
4          cmp_3      cmp_3       240    mg       BID 7.77         oral         TRUE

── Intermediary Outputs: Drug Costs ──

   regimen_name components cost_per_dose doses_per_ToT acq_cost_per_patient admin_cost_per_patient
1     Product Y  Product Y        $10.00     19.566964                $196              $6,005.10
2 cmp_1 + cmp_2      cmp_1       $200.00      3.493065                $699              $1,072.02
3 cmp_1 + cmp_2      cmp_2         $0.90      5.239598                  $5              $1,608.03
4          cmp_3      cmp_3     $2,633.00    472.998750           $1,245,406                $186.48
```

Example BIM: Results

print(base_case)

Summary of inputs

Results

- Cost per patient, for each cost category
- Number of patients treated
- Total costs and budget impact
- Cost breakdown by category
- Cost breakdown by sub-treatment and category

Plots

Console output

```
! The same @ToT (2.41) is assumed to apply to all treatments in regimen cmp_1 + cmp_2.

→ For treatment specific inputs, include a @ToT input for each treatment in the regimen.

── Model Results ──────────────────────────────────────────────────────────────────────────────────────────────────────

── Budget Impact Summary Results ──

                                                        Year 1          Year 2           Year 3
Eligible population for treatment with Product Y      37690.01        37992.14         38144.1
Population expected to receive Product Y                  7538         9498.03        11443.23
Total cost: without Product Y                  $30,894,495,965 $30,886,607,341  $31,005,095,862
Total cost: with Product Y                     $23,892,903,630 $21,474,533,899  $19,191,600,358
Net budget impact                              -$7,001,592,335 -$9,412,073,442 -$11,813,495,504

── Costs per patient ──

        Regimen acquisition_cost Admin_Cost monitoring_costs_PFS   TRAEs      total
1 cmp_1 + cmp_2             $703  $2,680.05                $2,100  $54.60     $5,538
2         cmp_3       $1,245,406    $186.48                $1,900 $122.00 $1,247,614
3     Product Y             $196  $6,005.10                $8,850  $30.70    $15,081

── Annual cost breakdown ──

── Scenario with new intervention

   Year acquisition_costs  admin_costs new.monitoring_costs_PFS  new.TRAEs           total
 Year 1   $23,479,105,864  $79,084,072             $333,556,611 $1,157,083 $23,892,903,630
 Year 2   $21,301,904,941  $90,771,102              $79,783,485 $2,074,371 $21,474,533,899
 Year 3   $19,012,241,488 $102,231,492              $72,473,798 $4,653,581 $19,191,600,358

── Scenario without new intervention

   Year acquisition_costs admin_costs ref.monitoring_costs_PFS  ref.TRAEs           total
 Year 1   $30,519,859,839 $39,922,432             $333,556,611 $1,157,083 $30,894,495,965
 Year 2   $30,764,507,035 $40,242,450              $79,783,485 $2,074,371 $30,886,607,341
 Year 3   $30,887,565,063 $40,403,420              $72,473,798 $4,653,581 $31,005,095,862

── Additional outputs ─────────────────────────────────────────────────────────────────────────────────────────────────

[1] "See 'yearly_cost_breakdown' in the object returned by 'run_model' for a breakdown of each cost-category per regimen"
```

Example BIM: Results

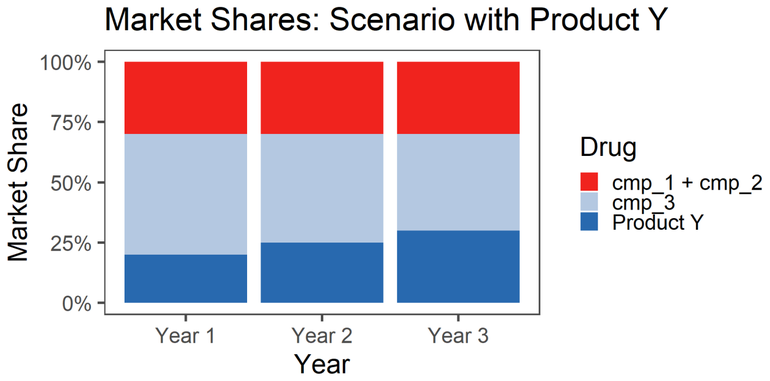
print(base_case)

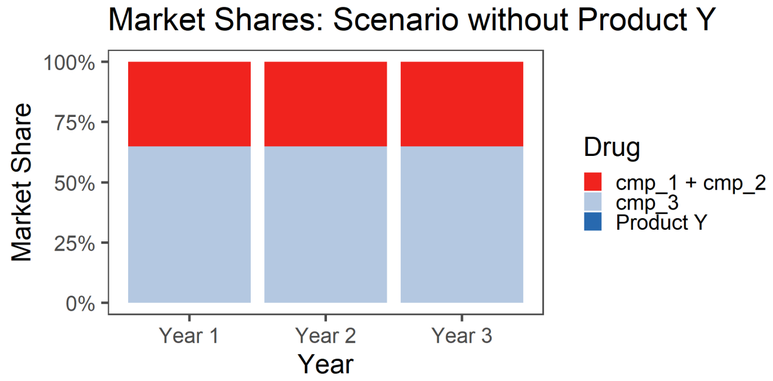
Summary of inputs

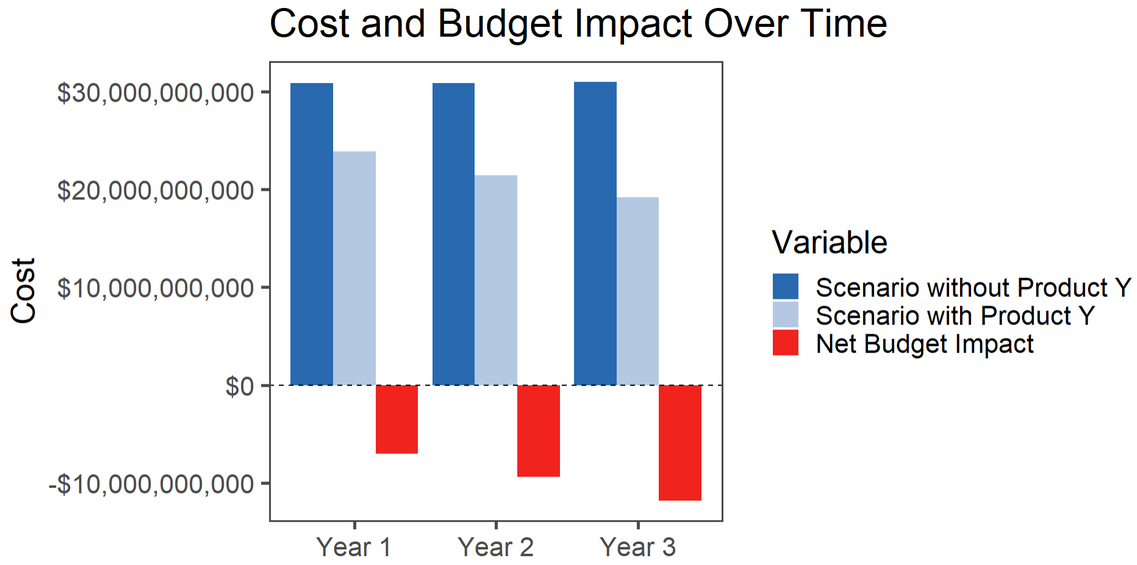
Results

Plots

- Market shares
- Budget impact over time

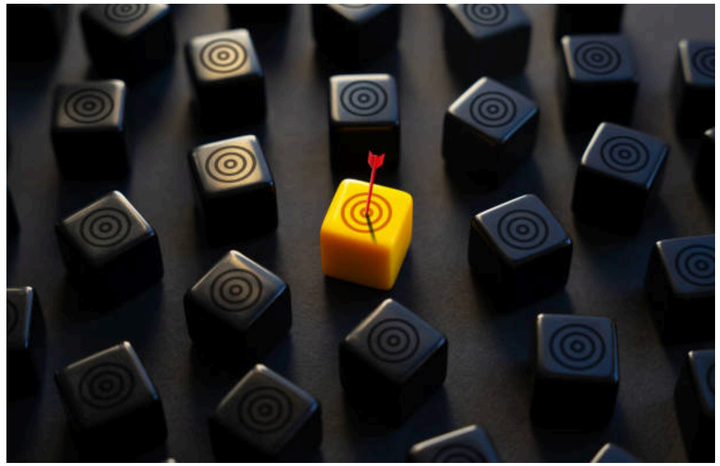






Agenda

1. Introduction to easyBIM
2. Methods
3. **Reflections and next steps**



Learnings and reflections

S7

easyBIM doesn't yet make full use of the potential benefits of S7.

S7 from a developer perspective:

- ➕ **Intuitive Syntax:** Simplifies the process of defining classes and methods.
- ➖ **S7 is still experimental** and is not yet part of base R, so substantial future changes can occur.

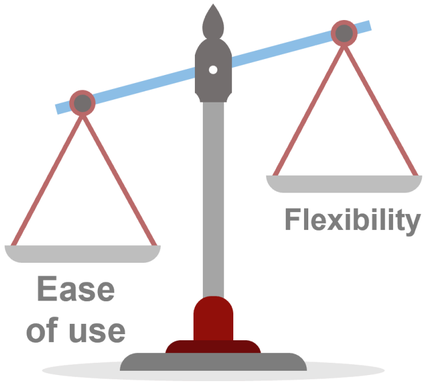
S7 from a user perspective:

- ➕ **Print methods** (although this is a benefit of classes in general, not specific to S7).
- ➕ **Error Prevention:** S7 enforces type checks, ensuring that properties are set correctly.
- ➕ **Documentation:** S7 classes inherently provide information about their structure and properties.

Functionality of easyBIM

Ease of use:

- ➕ **Predefined functions** simplify user interaction, reduce errors, and provide quick results for common scenarios.
- ➖ **Dependency on Assumptions:** Predefined functions rely on assumptions built into the package.



Flexibility:

- **Customization options** allow adaptation to unique user needs.
- ➖ **Complexity:** Increased flexibility often comes with a steeper learning curve.
- ➖ **Requires comprehensive documentation** to guide users in using flexible features effectively.

Next steps

SHINY

- Dynamic UI will be necessary.
- Additional inputs will be needed for sources.
- We'd like to make it easy to define treatment objects via inputs tables (e.g. via an Excel-based dosage schedule input table).

DOCUMENTATION

- We will continue to improve the documentation, including more examples and a vignette.

SHARE

- We plan to share easyBIM on GitHub, and we will be open to collaboration!

assertHE FUNCTION NETWORK

- We'd like to visualise how all the functions are connected, so we plan to use assertHE's function network.

TESTING and VALIDATION

- … assertHE's function network will reveal that we haven't done any formal testing yet! This will be a next step.
- Previously, we replicated existing models in easyBIM to compare results. We plan on conducting more of these validation exercises.

Appendix: Additional cost categories

Additional cost category inputs

```
# In this example, there are costs which should only apply
# while the patients are in the progression-free-survival stat
median_PFS <- list("PRODUCT Y" = 3,
                   "cmp_1 + cmp_2" = 2,
                   "cmp_3" = 1)

# Repeated costs (e.g. monitoring costs)
inputs_monitoring_costs_PFS <- repeated_cost(
  element_names = c("Inpatient visit",
                    "Hospitalization",
                    "Outpatient visit"),
  # It is not necessary to name each element of the vectors be
  # Names have been added for demonstration purposes.
  frequency = list("PRODUCT Y" = c("Inpatient visit" = 0,
                                   "Hospitalization" = 1,
                                   "Outpatient visit" = 2),
                   "cmp_1 + cmp_2"  = c("Inpatient visit" = 0,
                                        "Hospitalization" = 0,
                                        "Outpatient visit" =3)
                   "cmp_3"= c("Inpatient visit" = 0,
                              "Hospitalization" = 2,
                              "Outpatient visit" = 0)),
  time_interval = "monthly",
  num_intervals = median_PFS, # Number of time intervals
  unit_cost = c(150, 190, 70) # Per event
)
```

- Additional cost categories
  - easyBIM includes a class '**repeated_cost**' which can be used to include costs such as monitoring costs and TRAEs. Combining costs with frequency over a specified time period.

Add to `run_model()`:

```
# Run model
base_case <- easyBIM::run_model(
  currency = "$",
  time_horizon = 5,
  default_time_interval = "monthly",
  patient_weight = list(mean = 80),
  patient_funnel = patient_funnel_instance,
  market_shares = market_shares,
  bim_treatment_list = bim_treatment_list,
  list_admin_cost = list_admin_cost,
  additional_cost_categories = additional_cost_categories
)
```