Solution for Technical Provisions in R

Gabriel Foix
• Introduction to Technical Provisions (TP)
• Motivation
• Our Solution in R
• Example
• Conclusions
• Usually TP are the largest item on the balance sheet of a (re-)insurer

• Calculated TP enter the (market-consistent) balance sheet directly
  • Key input in the SCR calculation
  • Driver of the Profit & Loss Attribution

• Solvency II requires TP to be the “best estimate” of the current liabilities relating to insurance contracts (claims and premium provisions) plus a risk margin
Under Solvency II, TP consist of the present value claim provisions, premium provisions (best estimate), and risk margin.

The present value of the in- and out-going cash-flows can be calculated by (1) applying a payment pattern (PP) to the undiscounted reserves, (2) discounting them at their appropriate rate (currency) and finally (3) aggregating.
Motivation

Project carried out for a global insurer with presence in most of Europe

Some figures:

- +25 independent business units (BU)
- +4000 payment patterns (BU x LoB x measure x segment x variables)
- +20 currencies
- Different sources of information (accounting/finance, actuarial/reserving, credit risk etc)

Old process

- **Error-prone, time-consuming** and **not scalable** (MS Excel based)
- Not fulfilling the Solvency II governance requirements:
  
  "Insurers have process in place to ensure the appropriateness, completeness and accuracy of the data and calculations in their Technical Provisions"

- **Very difficult to trace / audit**
- Unable to face upcoming challenges like IFRS17
Motivation: Process ex-ante

Process diagram of the old Excel based approach (pivot tables, array formulas, etc)
Motivation: Process ex-ante

Example: payment pattern input

<table>
<thead>
<tr>
<th>lob</th>
<th>measure</th>
<th>segment</th>
<th>unit</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accident</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0097415</td>
<td>0.0284478</td>
<td>0.0498269</td>
<td>0.0534631</td>
</tr>
<tr>
<td>Health (Sickness)</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Marine, Aviation, Transp</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0167187</td>
<td>0.0715947</td>
<td>0.1192711</td>
<td>0.1682678</td>
</tr>
<tr>
<td>Credit/Mortgage and Su</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0073956</td>
<td>0.0512336</td>
<td>0.0903283</td>
<td>0.1554165</td>
</tr>
<tr>
<td>Crime/Fidelity/Pecunian</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0172448</td>
<td>0.0215113</td>
<td>0.0539542</td>
<td>0.0682998</td>
</tr>
<tr>
<td>Motor - 3rd Party/Liabilit</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0887523</td>
<td>0.1648557</td>
<td>0.1973106</td>
<td>0.1951389</td>
</tr>
<tr>
<td>Motor - All Other</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.1072621</td>
<td>0.1844396</td>
<td>0.2515469</td>
<td>0.2483029</td>
</tr>
<tr>
<td>Property</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0419536</td>
<td>0.1215095</td>
<td>0.1854559</td>
<td>0.2180664</td>
</tr>
<tr>
<td>Property - Engineering L</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0621928</td>
<td>0.1447307</td>
<td>0.2242805</td>
<td>0.2462537</td>
</tr>
<tr>
<td>Liability - Primary - Prod</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0155705</td>
<td>0.0562602</td>
<td>0.0936267</td>
<td>0.1125019</td>
</tr>
<tr>
<td>Liability - Primary - Non</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0165636</td>
<td>0.045626</td>
<td>0.0553627</td>
<td>0.0453</td>
</tr>
<tr>
<td>Liability - Excess Polic</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Professional Indemnity</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Work Comp/EL</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.0621928</td>
<td>0.1447307</td>
<td>0.2242805</td>
<td>0.2462537</td>
</tr>
<tr>
<td>Work Comp/EL - High D</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Multi-Peril</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Legal Expenses</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.00824</td>
<td>0.0478303</td>
<td>0.0783058</td>
<td>0.1002509</td>
</tr>
<tr>
<td>Assistance</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.1072621</td>
<td>0.1844396</td>
<td>0.2515469</td>
<td>0.2483029</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>ceded</td>
<td>corp</td>
<td>unit1</td>
<td>0.1489704</td>
<td>0.2052116</td>
<td>0.2709924</td>
<td>0.231669</td>
</tr>
<tr>
<td>Accident</td>
<td>ceded</td>
<td>corp</td>
<td>unit2</td>
<td>0.1504051</td>
<td>0.1500153</td>
<td>0.2016272</td>
<td>0.1317237</td>
</tr>
</tbody>
</table>

Key = Dim1 & Dim2 & Dim3 & Dim4
TP package with new S3 pattern objects…

@param keys \texttt{data.frame} providing the keys for the pattern.
@param pmat \texttt{matrix} representing the actual pattern. Columns represent the quarterly or annual developments.

```r
#' @export
pattern = function(keys, pmat) {
  assert_that(is.matrix(pmat))
  assert_that(nrow(keys) == nrow(pmat))
  assert_that(nrow(unique(keys)) == nrow(keys))
  p = list(keys = keys, pattern = pmat)
  class(p) = "pattern"
  p
}
```

Unique key
Our Solution in R

...plus the corresponding S3 Group Generic Functions...

Ops(e1, e2)
+,-,*,/,%%, ...

```r
@export
Ops.pattern = function(e1, e2 = NULL) {
  FUN = get(.Generic, envir = parent.frame(), mode = "function")
  if(is.null(e2)) {
    pattern(e1$keys, FUN(e1$pattern))
  } else {
    if(is(e2, "pattern")) {
      pa = alignPattern(e1, e2, fill = 0)
      pattern(pa$p1$keys, FUN(pa$p1$pattern, pa$p2$pattern))
    } else if(is.atomic(e2)) {
      pattern(e1$keys, FUN(e1$pattern, e2))
    } else {
      stop("Unsupported second operand type: ", typeof(e2))
    }
  }
}
```

```r
@export
Math.pattern = function(x, ...) {
  FUN = get(.Generic, envir = parent.frame(), mode = "function")
  pattern(x$keys, FUN(x$pattern, ...))
}
```

Math(x, \dots)
cumsum, cumprod...
Our Solution in R

...plus some other methods...

\[
\begin{align*}
\text{s3method}("[", \text{pattern}) \\
\text{s3method}(\text{Math}, \text{pattern}) \\
\text{s3method}(\text{Ops}, \text{pattern}) \\
\text{s3method}(\text{Summary}, \text{pattern}) \\
\text{s3method}(\text{aggregate}, \text{pattern}) \\
\text{s3method}(\text{as.data.frame}, \text{pattern}) \\
\text{s3method}(\text{c}, \text{pattern}) \\
\text{s3method}(\text{cbind}, \text{pattern}) \\
\text{s3method}(\text{replaceNA}, \text{pattern}) \\
\text{s3method}(\text{split}, \text{pattern})
\end{align*}
\]

...and of course functions

\[
\begin{align*}
\text{importClaimsPP} & \leftarrow \text{function(file)} \right) \\
\text{claimsPP} = \\
\text{readNamedRegionFromFile(file, name = "SU.PP.UPR")} \\
\text{subset}(\text{measure} \%\% \text{c("gross", "ceded")}) \\
\text{plyr:::splitter_d(}.(\text{measure})) \\
\text{papply(}.function(\text{pp}) \\
\text{keys} = \text{pp}[, \text{c("Segment", "lobcbs")}] \\
\text{names}(\text{keys}) = \text{c("Segment", "PN")} \\
\text{idx} = \text{grep("Q[0-9]+", names(}\text{pp})} \\
\text{pattern(}.\text{keys, as.matrix(pp[, idx, drop = FALSE]))} \\
\text{return(claimsPP)}
\end{align*}
\]

XLConnect
Our Solution in R: Example

1. Read original Excel files
2. Calculate BE Claims & Premiums
3. Discount & aggregate
4. Write to Excel files as before

Calculate Best Estimate (BE)

```r
message("* Full cash flow for SII Best Estimate Reserve")
fullcfBeReserve = mapply(
  cpCashFlow, # Calculates the future cash flow pattern given a claims payment pattern and an amount pattern.
  claimsPP[gross_ceded], # @param pp Claims payment \code{pattern}
  splitPatternAt( # @param amount Amount \code{pattern} (BE res - Fut ULAE)
    patternFromDF( # @return Cash flow \code{pattern}
      reservesAllComb, # params for run-off pattern calculation
      keyCols = c(keyCols, "AY"),
      patternCols = c("G.BE.Res", "C.BE.Res")
    ), # MoreArgs = list(year = year, quarter = quarter)
    ),
  ,
  )
```

© Mirai Solutions GmbH
Combine claims and premiums provisions discounting at the EIOPA rates per currency

```r
discountedCashFlow = papply(obj$UndiscountedCashFlow[[1]], function(p) {
  keys = cbind(objkeys, p$keys)
  # Relevant discount factors
  subDiscFact = discFact[discFact$curr %in% curr,]
  rownames(subDiscFact) = subDiscFact$curr
  idx = which(names(subDiscFact) == "curr")
  # Cash flow discounting
  disc = as.matrix(curBlend[, curr]) %*% as.matrix(subDiscFact[curr,-idx])
  res$pattern[p$keys, disc[, seq_len(ncol(p$pattern)), drop=F]]
  res
})

discountedAmount = papply(discountedCashFlow, pRowSums)
```

Aggregate and write out TP respecting
- Same format/granularity as previous processes
- Adding additional reports (intermediate calculation results, aggregate summaries, variable dependencies)
Conclusions

Project achievements
• Accurate calculations (fixed several mistakes)
• Automated and centralized process that frees up resources in the local business units
• Data consistency, save time in audit processes
• Improved flexibility, allows Analysis of Change and other what-if scenarios

R contribution
• Business process clarity, reduce (isolate) complexity
• Code readability, accessibility
• Respecting input and output formats already in place
• Exportable modules to be reused in other applications
• Reproducibility, testability (code versioning, unit testing!)