Solution for Technical Provisions in R

Gabriel Foix

Mirai Solutions GmbH
Tödistrasse 48
CH-8002 Zurich
Switzerland

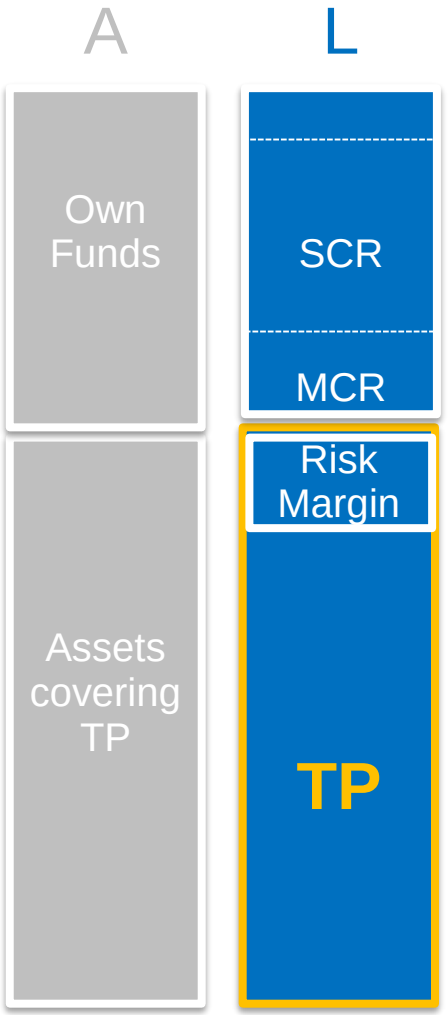
info@mirai-solutions.com
www.mirai-solutions.com

Index

- Introduction to Technical Provisions (TP)
- Motivation
- Our Solution in R
- Example
- Conclusions

Introduction Technical Provisions (TP)

- 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



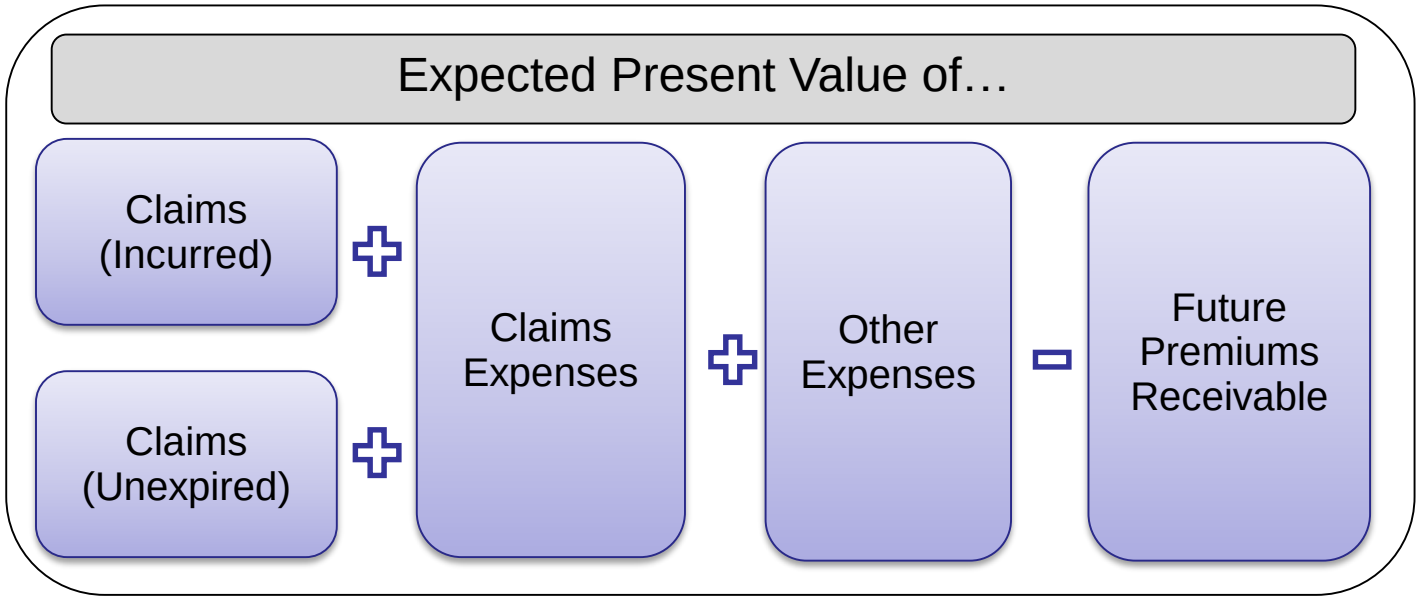
Introduction Technical Provisions (TP)

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.

Future cash-flows to claim events already occurred (past)

Future claim events on existing policies



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)

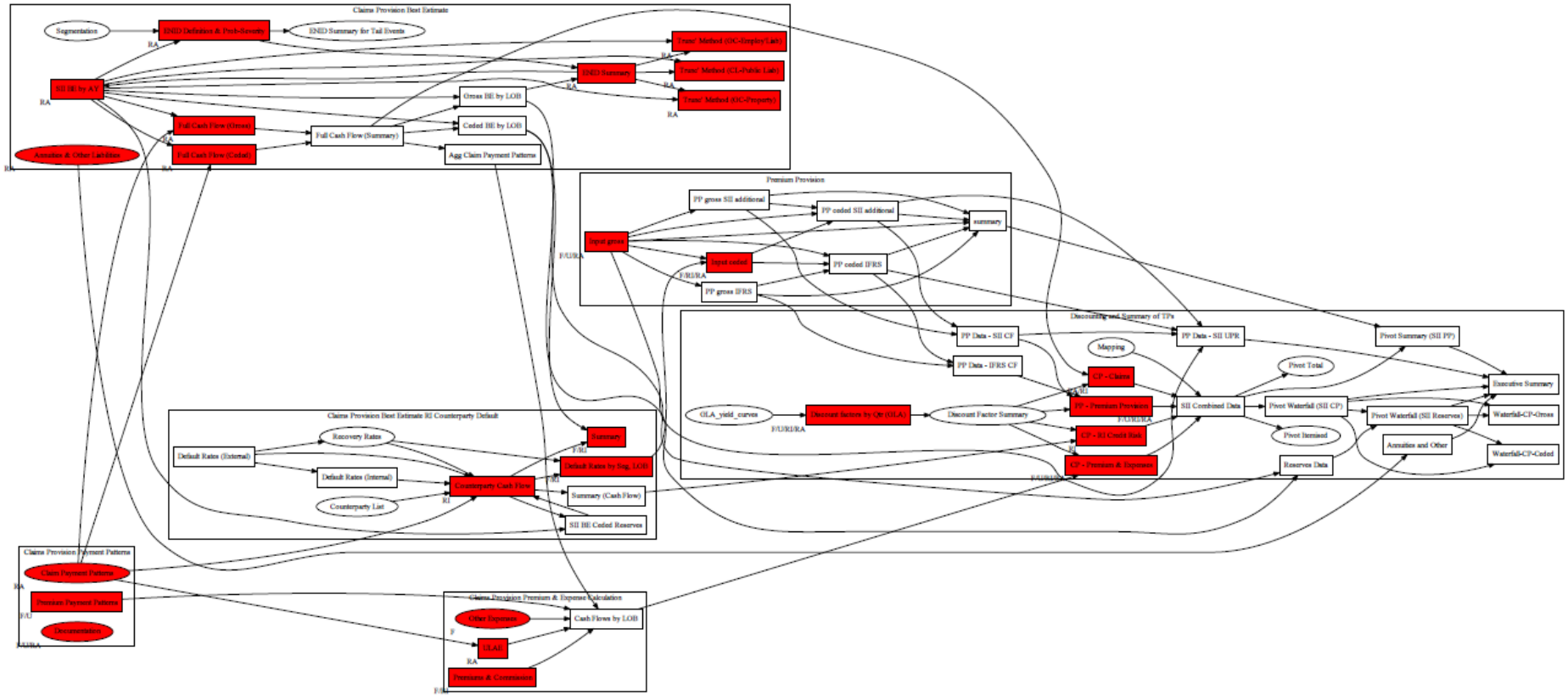
Old process

- **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

Dimension1: Lines of Business (LoB)

Dim2: Measures

Dim3: Segment

Dim4: Business Units

Dim5: t (quarters)

Key = Dim1 & Dim2 & Dim3 & Dim4

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

Our Solution in R

TP package with new S3 pattern objects…

@param keys \code{data.frame} providing the keys for the pattern.

@param pmat \code{matrix} representing the actual pattern. Columns represent the quarterly or annual developments.

```
#' @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…

```
S3method("[",pattern)
S3method(Math,pattern)
S3method(Ops,pattern)
S3method(Summary,pattern)
S3method(aggregate,pattern)
S3method(as.data.frame,pattern)
S3method(c,pattern)
S3method(cbind,pattern)
S3method(replaceNA,pattern)
S3method(split,pattern)
........
```

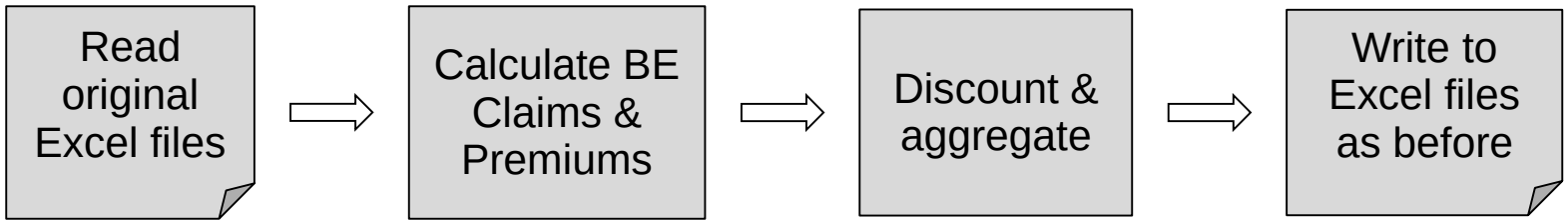
…and of course functions

```
export(alignPattern)
export(importClaimsPP)
export(importExtCounterparties)
export(multiplyPattern)
export(patternFromDF)
export(readIFRSData)
export(run_off)
export(writeTPReport)
......
```

XLConnect

```
importClaimsPP <- function(file)) {
  claimsPP =
    readNamedRegionFromFile(file, name = "SU.PP.UPR") %>%
    subset(measure %in% c("gross", "ceded")  %>%
    plyr:::splitter_d(.(measure)) %>%
    papply(function(pp) {
      keys = pp[, c("Segment", "lobmcbs")]
      names(keys) = c("Segment", "PN")
      idx = grep("Q[0-9]+", names(pp))
      pattern(keys, as.matrix(pp[, idx, drop = FALSE]))
    })
  return(claimsPP)
}
```

Our Solution in R: Example



Calculate Best Estimate (BE)

```
message("* Full cash flow for SII Best Estimate Reserve")
fullCfBeReserve =
  mapply(
    cpCashFlow,
    claimsPP[gross_ceded],
    splitPatternAt(
      patternFromDF(
        reservesAllComb,
        keyCols = c(keyCols, "AY"),
        patternCols = c("G.BE.Res", "C.BE.Res")
      )
    ),
    # params for run-off pattern calculation
    MoreArgs = list(year = year, quarter = quarter)
  )
```

```
#' Calculates the future cash flow pattern given a claims
#' payment pattern and an amount pattern.
#' @param pp Claims payment \code{pattern}
#' @param amount Amount \code{pattern} (BE res - Fut ULAE)
#' @return Cash flow \code{pattern}
cpCashFlow = function(pp, amount, ...) {...}
```

Our Solution in R: Example

Combine claims and premiums provisions discounting at the EIOPA rates per currency

```
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 = p * 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

Mirai Solutions
smarter analytics - better decisions

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!)