R/Insurance Webinars Jan 2024

For the R Consortium's R/Adoption Series

Welcome

1. From Excel to programming in R (today's topic)
2. From programming in R to putting R into production
3. R performance culture
4. High performance programming in R

Delivered on behalf of the R Consortium by Georgios Bakoloukas and Benedikt Schamberger, Actuarial Control, Group Risk Management, Swiss Re

Background

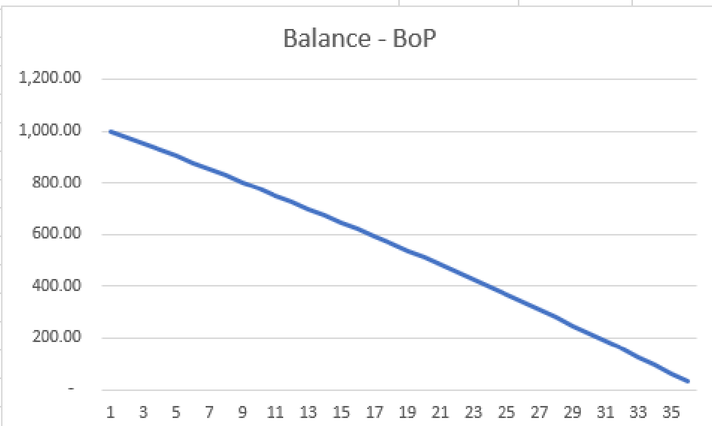
- Swiss Re internal R community sponsored by our Group Chief Actuary Philip Long (Atelier programme)
- 2000+ community with 500+ regular coders who also support each other
- The case we see today appeared in our Microsoft Teams community channel by an actuary in a high-growth market
- Views expressed belong solely to the speakers and not necessarily to the speaker’s employer

Running example for webinars 1 & 2

- Insurer covers the remaining balance of loans in case of death/disability of the borrower
- Requires a quote for a portfolio of caa. 300,000 policies
- Has provided information on a) loan amount b) loan duration and c) interest rate for each policy
- Problem: The actuary needs to calculate the sum-insured profile for each policy as it amortises
- A solution in Excel and a potential solution in R

A credit life insurance quote

Data input	
Loan Amount	1,000
Loan Term (in years)	3
Loan Term (in months)	36
Parameter input	
Interest Rate (Annual Percentage Rate)	10%
Monthly interest rate	0.83%
Modelling and Output	
Equivalent monthly payment	
EMI	32.27
Total Payments	1,161.62
Total Interest	161.62
Key	
Inputs	
Distinct calc in column	



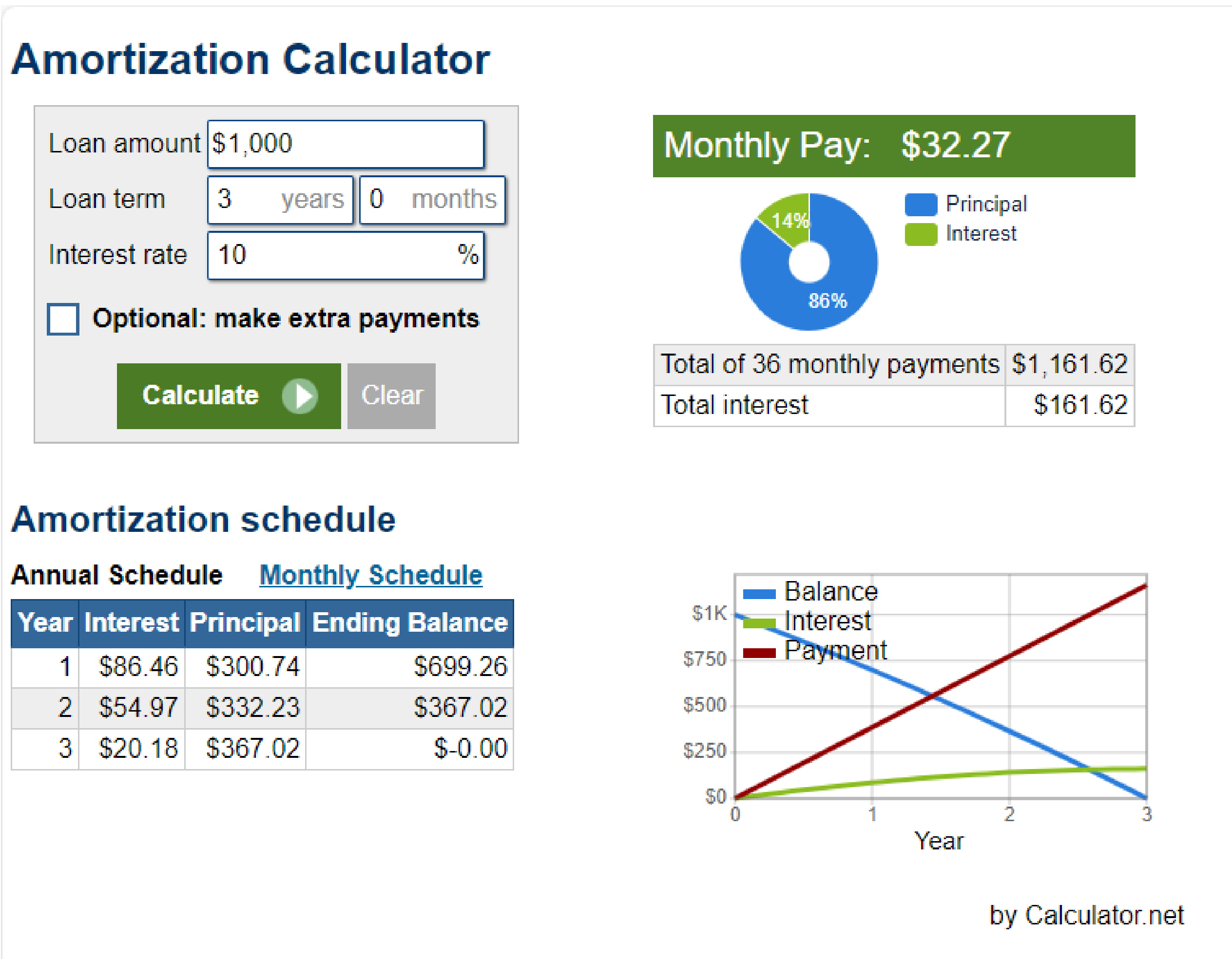
Modelling and output

Monthly cashflows

Time-months	Balance - BoP	Interest	Principal	Balance - EoP	Time-years
1	1,000.00	8.33	23.93	976.07	1
2	976.07	8.13	24.13	951.93	1
3	951.93	7.93	24.33	927.60	1
4	927.60	7.73	24.54	903.06	1
5	903.06	7.53	24.74	878.32	1
6	878.32	7.32	24.95	853.37	1
7	853.37	7.11	25.16	828.22	1
8	828.22	6.90	25.37	802.85	1
9	802.85	6.69	25.58	777.27	1
10	777.27	6.48	25.79	751.48	1
11	751.48	6.26	26.00	725.48	1
12	725.48	6.05	26.22	699.26	1
13	699.26	5.83	26.44	672.82	2
14	672.82	5.61	26.66	646.16	2
15	646.16	5.38	26.88	619.27	2
16	619.27	5.16	27.11	592.17	2
17	592.17	4.93	27.33	564.84	2
18	564.84	4.71	27.56	537.28	2
19	537.28	4.48	27.79	509.49	2
20	509.49	4.25	28.02	481.46	2
21	481.46	4.01	28.25	453.21	2
22	453.21	3.78	28.49	424.72	2
23	424.72	3.54	28.73	395.99	2
24	395.99	3.30	28.97	367.02	2
25	367.02	3.06	29.21	337.81	3
26	337.81	2.82	29.45	308.36	3
27	308.36	2.57	29.70	278.67	3
28	278.67	2.32	29.94	248.72	3
29	248.72	2.07	30.19	218.53	3
30	218.53	1.82	30.45	188.08	3
31	188.08	1.57	30.70	157.38	3
32	157.38	1.31	30.96	126.42	3
33	126.42	1.05	31.21	95.21	3
34	95.21	0.79	31.47	63.74	3
35	63.74	0.53	31.74	32.00	3
36	32.00	0.27	32.00	-	3

Graphical user interfaces available

eg https://www.calculator.net/amortization-calculator.html

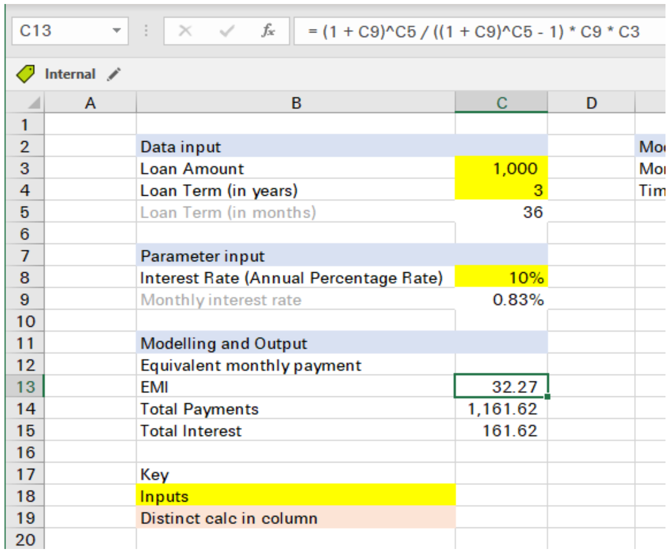


How to calculate the Equivalent Monthly Installment (EMI)

$$EMI = \frac{(1+i)^n}{(1+i)^n - 1} \times i \times L$$

For the derivation and an intuitive understanding see https://math.stackexchange.com/questions/279844/how-the-formula-for-emi-is-derived

Calculating EMI in Excel and R is similar



C13 | fx = (1 + C9)^C5 / ((1 + C9)^C5 - 1) * C9 * C3

Internal

	A	B	C	D	
1					
2		Data input			Mo
3		Loan Amount	1,000		Mo
4		Loan Term (in years)	3		Tim
5		Loan Term (in months)	36		
6					
7		Parameter input			
8		Interest Rate (Annual Percentage Rate)	10%		
9		Monthly interest rate	0.83%		
10					
11		Modelling and Output			
12		Equivalent monthly payment			
13		EMI	32.27		
14		Total Payments	1,161.62		
15		Total Interest	161.62		
16					
17		Key			
18		Inputs			
19		Distinct calc in column			
20					

```r
# Data and parameter input
A <- 1000
n_yr <- 3
int_yr <- 0.1

# Intermediate calculation
n <- n_yr * 12
i <- int_yr / 12

emi <- (1 + i)^n / ((1 + i)^n - 1) * i * A
emi
```

```
[1] 32.26719
```

Loan balance calculation in Excel - 1

SUM | =F5+G5-C$13

Internal

	A	B	C	D	E	F	G	H	I	J
1										
2		Data input			Modelling and output					
3		Loan Amount	1,000		Monthly cashflows					
4		Loan Term (in years)	3		Time-months	Balance - BoP	Interest	Principal	Balance - EoP	Time-years
5		Loan Term (in months)	36		1	1,000.00	8.33	23.93	=F5+G5-C$13	1
6					2	976.07	8.13	24.13	951.93	1
7		Parameter input			3	951.93	7.93	24.33	927.60	1
8		Interest Rate (Annual Percentage Rate)	10%		4	927.60	7.73	24.54	903.06	1
9		Monthly interest rate	0.83%		5	903.06	7.53	24.74	878.32	1
10					6	878.32	7.32	24.95	853.37	1
11		Modelling and Output			7	853.37	7.11	25.16	828.22	1
12		Equivalent monthly payment			8	828.22	6.90	25.37	802.85	1
13		EMI	32.27		9	802.85	6.69	25.58	777.27	1
14		Total Payments	1,161.62		10	777.27	6.48	25.79	751.48	1
15		Total Interest	161.62		11	751.48	6.26	26.00	725.48	1
16					12	725.48	6.05	26.22	699.26	1
17		Key			13	699.26	5.83	26.44	672.82	2
18		Inputs			14	672.82	5.61	26.66	646.16	2
19		Distinct calc in column			15	646.16	5.38	26.88	619.27	2
20					16	619.27	5.16	27.11	592.17	2

Loan balance calculation in Excel - 2

SUM | =F5+F5*C$9-C$13

Internal

	A	B	C	D	E	F	G	H	I	J
1										
2		Data input			Modelling and output					
3		Loan Amount	1,000		Monthly cashflows					
4		Loan Term (in years)	3		Time-months	Balance - BoP	Interest	Principal	Balance - EoP	Time-years
5		Loan Term (in months)	36		1	1,000.00	8.33	23.93	=F5+F5*C$9-C$13	
6					2	976.07	8.13	24.13	951.93	1
7		Parameter input			3	951.93	7.93	24.33	927.60	1
8		Interest Rate (Annual Percentage Rate)	10%		4	927.60	7.73	24.54	903.06	1
9		Monthly interest rate	0.83%		5	903.06	7.53	24.74	878.32	1
10					6	878.32	7.32	24.95	853.37	1
11		Modelling and Output			7	853.37	7.11	25.16	828.22	1
12		Equivalent monthly payment			8	828.22	6.90	25.37	802.85	1
13		EMI	32.27		9	802.85	6.69	25.58	777.27	1
14		Total Payments	1,161.62		10	777.27	6.48	25.79	751.48	1
15		Total Interest	161.62		11	751.48	6.26	26.00	725.48	1
16					12	725.48	6.05	26.22	699.26	1
17		Key			13	699.26	5.83	26.44	672.82	2
18		Inputs			14	672.82	5.61	26.66	646.16	2
19		Distinct calc in column			15	646.16	5.38	26.88	619.27	2
20					16	619.27	5.16	27.11	592.17	2

Loan balance calculation in Excel - 3

SUM | =I5

Internal

	A	B	C	D	E	F	G	H	I	J
1										
2		Data input			Modelling and output					
3		Loan Amount	1,000		Monthly cashflows					
4		Loan Term (in years)	3		Time-months	Balance - BoP	Interest	Principal	Balance - EoP	Time-years
5		Loan Term (in months)	36		1	1,000.00	8.33	23.93	976.07	1
6					2	=I5	8.13	24.13	951.93	1
7		Parameter input			3	951.93	7.93	24.33	927.60	1
8		Interest Rate (Annual Percentage Rate)	10%		4	927.60	7.73	24.54	903.06	1
9		Monthly interest rate	0.83%		5	903.06	7.53	24.74	878.32	1
10					6	878.32	7.32	24.95	853.37	1
11		Modelling and Output			7	853.37	7.11	25.16	828.22	1
12		Equivalent monthly payment			8	828.22	6.90	25.37	802.85	1
13		EMI	32.27		9	802.85	6.69	25.58	777.27	1
14		Total Payments	1,161.62		10	777.27	6.48	25.79	751.48	1
15		Total Interest	161.62		11	751.48	6.26	26.00	725.48	1
16					12	725.48	6.05	26.22	699.26	1
17		Key			13	699.26	5.83	26.44	672.82	2
18		Inputs			14	672.82	5.61	26.66	646.16	2
19		Distinct calc in column			15	646.16	5.38	26.88	619.27	2
20					16	619.27	5.16	27.11	592.17	2

Loan balance calculation in Excel - 4

	B	C	D	E	F	I
1						
2	Data input			Modelling and output		
3	Loan Amount	1000		Monthly cashflows		
4	Loan Term (in years)	3		Time-months	Balance - BoP	Balance - EoP
5	Loan Term (in months)	=C4*12		1	=C3	=F5+F5*C$9-C$13
6				=E5+1	=I5	=F6+F6*C$9-C$13
7	Parameter input			=E6+1	=I6	=F7+F7*C$9-C$13
8	Interest Rate (Annual Percentage Rate)	0.1		=E7+1	=I7	=F8+F8*C$9-C$13
9	Monthly interest rate	=C8/12		=E8+1	=I8	=F9+F9*C$9-C$13
10				=E9+1	=I9	=F10+F10*C$9-C$13
11	Modelling and Output			=E10+1	=I10	=F11+F11*C$9-C$13
12	Equivalent monthly payment			=E11+1	=I11	=F12+F12*C$9-C$13
13	EMI	= (1 + C9)^C5 / ((1 + C9)^C5 - 1) * C9 * C3		=E12+1	=I12	=F13+F13*C$9-C$13
14	Total Payments	=C13*C5		=E13+1	=I13	=F14+F14*C$9-C$13
15	Total Interest	=SUM(G5:G148)		=E14+1	=I14	=F15+F15*C$9-C$13
16				=E15+1	=I15	=F16+F16*C$9-C$13
17	Key			=E16+1	=I16	=F17+F17*C$9-C$13
18	Inputs			=E17+1	=I17	=F18+F18*C$9-C$13
19	Distinct calc in column			=E18+1	=I18	=F19+F19*C$9-C$13
20				=E19+1	=I19	=F20+F20*C$9-C$13
21				=E20+1	=I20	=F21+F21*C$9-C$13
22				=E21+1	=I21	=F22+F22*C$9-C$13
23				=E22+1	=I22	=F23+F23*C$9-C$13
24				=E23+1	=I23	=F24+F24*C$9-C$13

$$P_{n+1} = P_n + P_n \times i_n - EMI$$

Vectorisation in R

R supports vectorised calculations. An example:

```r
# I have 2 vectors of values
x <- c(1, 3, 5, 7)
y <- c(2, 4, 6, 8)
x
```

```
[1] 1 3 5 7
```

```r
y
```

```
[1] 2 4 6 8
```

```r
# I want to add them together
# Because `+` is a vectorised operator, I can do:
z <- x + y
z
```

```
[1]  3  7 11 15
```

-> No need to copy-paste or drag-down; it appears once

It is not always easy to vectorise

Eg if subsequent values of a vector depend on the previous value of the same vector.

Writing an explicit iterative loop is a often a solution. The previous example:

```
z <- double(length = length(x))         # initialise output
for (j in 1:length(z)) {                # iterator
  z[j] <- x[j] + y[j]                   # body of loop
}
z
```

```
[1]  3  7 11 15
```

It works but often verbose

Recursion may help

Recursion can potentially succinctly describe the calculation

- We will explore a couple of functions that can help: `reduce()` and `accumulate()`
- But we will start with `sum()` and `cumsum()` which can be considered special cases of the above

`+`, `sum`, `cumsum`

+ and sum

+ is a binary operator for addition, under the hood is a function

```r
1 + 2
```

```
[1] 3
```

```r
`+`(1, 2)
```

```
[1] 3
```

Can’t use more than 2 arguments (binary operator)

```r
`+`(1, 2, 3)
```

```
Error in `+`(1, 2, 3): operator needs one or two arguments
```

Can apply + iteratively, thankfully we have: sum

```r
`+`(3, `+`(1, 2))     # inconvenient
```

```
[1] 6
```

```r
sum(1, 2, 3)
```

```
[1] 6
```

+ and `cumsum`

```r
x <- c(1, 2, 3)
x
```

```
[1] 1 2 3
```

Calculate cumulative sum

```r
c(x[1], x[1] + x[2], x[1] + x[2] + x[3])
```

```
[1] 1 3 6
```

Thankfully we have cumsum as a function

```r
cumsum(x)
```

```
[1] 1 3 6
```

base::Reduce

Reduce uses a binary function to successively combine the elements of a given vector

Define a vector

```r
x <- c(1, 2, 3)
x
```

```
[1] 1 2 3
```

Successively combine elements of x using a binary function

```r
Reduce(f = `+`, x = x)                          # with + it is like sum
```

```
[1] 6
```

Accumulate the successive reduce combinations

```r
Reduce(f = `+`, x = x, accumulate = TRUE)  # with + it is like cumsum
```

```
[1] 1 3 6
```

`purrr::reduce` and `accumulate`

```r
x <- c(1, 2, 3)
purrr::reduce(.x = x, .f = `+`)
```

```
[1] 6
```

```r
purrr::accumulate(.x = x, .f = `+`)
```

```
[1] 1 3 6
```

Compared to base R, purrr functions consistently use `.` as a prefix, are type stable, and all start with the data, followed by the function

`accumulate` exercise 1

Start with a vector of values

```r
x <- c(2, 3, 5)
x
```

```
[1] 2 3 5
```

Define a 2-argument function

```r
fn <- function(a, b) {a^2 + b}
```

Apply the function successively over the elements of x

```r
# first argument: result of previous application
# second argument: the next value of the vector
purrr::accumulate(.x = x, .f = fn)
```

```
[1]  2  7 54
```

`accumulate` exercise 2

- Apply a 1-argument function to a single value for k times
- Use `accumulate()` by neutralising the 2nd argument value

```
x <- 2                                # the single value
fn2 <- function(x, y) {x^2}           # 1-arg function; we don't need y
k <- 5                                # iterate k times
# first argument: the inital value provided by .init
purrr::accumulate(.x = 1:(k-1), .f = fn2, .init = x)
```

```
[1]     2     4    16   256 65536
```

Amortise

Using values of the first example:

A	i	n	emi
1000	0.008333333	36	32.26719

	D	E	F
1			
2		Modelling and output	
3		Monthly cashflows	
4		Time-months	Balance - BoP
5		1	1,000.00
6		2	976.07
7		3	951.93
8		4	927.60
9		5	903.06
10		6	878.32
11		7	853.37
12		8	828.22

we define: $P_{n+1} = P_n + P_n \times i_n - EMI$

```r
fn3 <- function(a, b) {a + a * i - emi}
```

And apply it

```r
P <- purrr::accumulate(1:(n-1), fn3, .init = A)
P[1:8]
```

```
[1] 1000.0000  976.0661  951.9328  927.5984  903.0612
878.3196  853.3717
[8]  828.2159
```

Putting it all together for one value

```r
# Data and parameter input ---------------------------------
A <- 1000
n_yr <- 3
int_yr <- 0.1
# Intermediate calculations --------------------------------
n <- n_yr * 12
i <- int_yr / 12
emi <- (1 + i)^n / ((1 + i)^n - 1) * i * A
# Define amortisation function -----------------------------
amortise <- function(a, b) {a + a * i - emi}
# apply it successively to the loan amount -----------------
P <- purrr::accumulate(1:(n-1), amortise, .init = A)
P[1:6]                 # print first few results
```

```
[1] 1000.0000  976.0661  951.9328  927.5984  903.0612  878.3196
```

Many working patterns are common between Excel and R. It often pays off to switch mindset from spreadsheet computing to programming (will see examples next week)

Next: From programming in R to putting R into production

Building on current example:

- Build functions to reuse logic and abstract away complexity
- Iterate over all data with functional programming approach
- Bundle functions into packages (programmer-to-programmer interface)
- Expose functions into Shiny (graphical user interface)
- Expose functions into Web APIs (computer-to-computer interface)

Join the R Consortium

R consortium

R Consortium Impact

- R Consortium Community **Grants** and Sponsorships Over **USD $1.4 Million**
- Organize large scale **collaborative projects**
  - R Validation Hub
  - R-Ladies
  - Diversity and Inclusion Working Group
- Co-host multidisciplinary **data science forums**
  - Stanford Data Institute
- Direct support for key **R events**
  - R/Medicine, R/Pharma, useR!, LatinR, more
- Direct support for **R User Groups**

R/Medicine 2023

R Ladies

useR!

LATINR

COVID-19 Data Forum

Organizations Can Become a Member Today!

Email Joseph Rickert at **director@r-consortium.org** to set up first call

Q&A