Mortgage Loan & Pool Amortization & Prepay Mechanics

Rei Shinozuka rei@reishinozuka.com

2025-05-16

Part I. Amortization and Prepayment Mechanics for Fixed-Rate Level-Pay Mortgages

Description and Definitions

Fixed-Rate Level-Pay mortgages comprise the great majority of loans originated in the US.

- Fixed-Rate: constant interest rate over the life of the loan
- Level-Pay: Loan is structured such that monthly payment is constant over the life of the loan.

The level-pay structure is facilitated by *amortizing* the balance over the life of the loan after paying monthly interest. The amount of the amortization is known as the *scheduled principal* and the remaining balance is known as the *scheduled balance*.

Most of the numerical conventions which follow are based on the reference *Standard Formulas for the Analysis of Mortgage-Backed Securities and Other Related Securities* (1999) published by the **Bond Market Association** (today known as **SIFMA**) available at the URL:

https://www.sifma.org/wp-content/uploads/2017/08/chsf.pdf

and denoted *SF1999* in any references below.

Basic Loan Amortization Mechanics

Loan Amortization

The below example shows scheduled amortization, which occurs when the borrower pays exactly the monthly payment each month. Principal and interest equal the monthly payment, which may be referred to as the *scheduled payment* with interest comprising the majority of the monthly payment in the early part of the loan. Under scheduled amortization, the balances at the end of each month are referred to as *scheduled balance*, the principal amount of the scheduled payment each month is referred to as the *scheduled principal*. The payment window is the term of the loan.

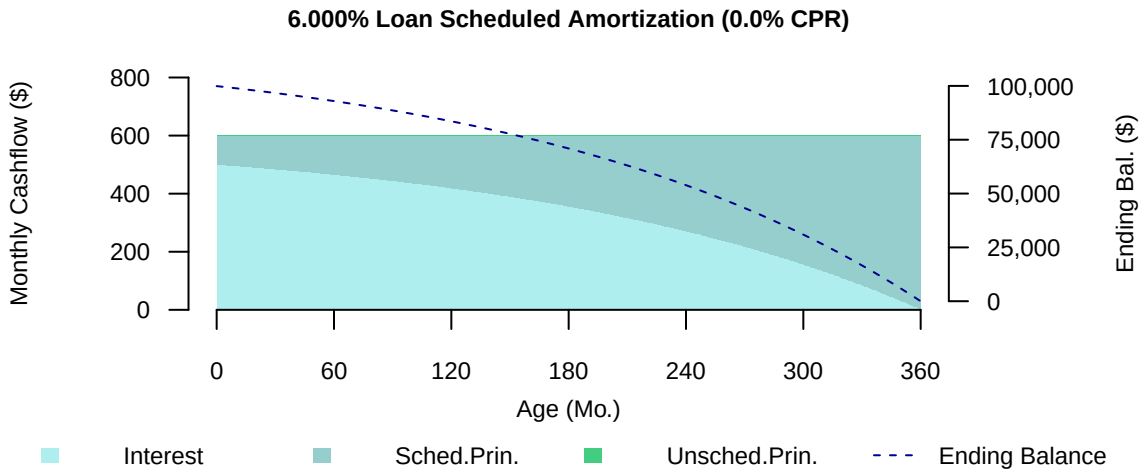

Figure 1: Scheduled Amortization

Early Loan Repayment

The borrower has the right to pay off the full balance of the loan at any time, to facilitate a refinancing or property sale, resulting in the loan being *paid in full*.

The below example shows a loan paying in full prior to loan maturity.

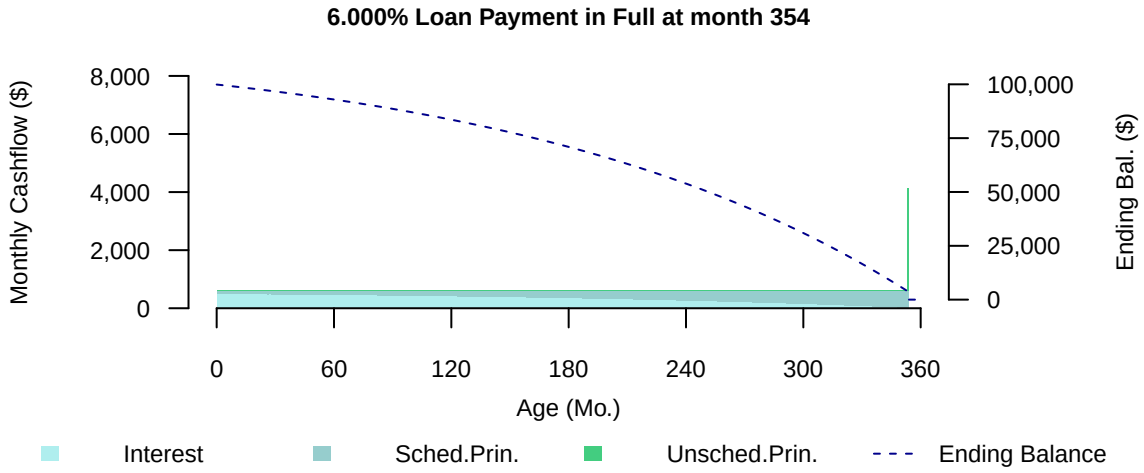

Figure 2: Borrower Repayment prior to Term

Loan Curtailments

The borrower may also partially prepay the remaining balance of the loan resulting in a *curtailment*. Curtailments are typically small relative to the monthly payment, often produced when borrowers round up the monthly payment to a multiple of $100 to simplify book keeping or to shorten the loan term.
Both payments-in-full and curtailments are types of *prepayments*, also known as *unscheduled principal* payments.

Curtailments do not change the scheduled payment (interest plus scheduled principal). The curtailments accelerate the paydown of remaining loan balance and hence reduce the duration of the payment window. The total interest paid is also reduced.

The below example shows a loan where the borrower pays a higher monthly amount $700 than the scheduled

amount. $599.55. The excess amount $100.45 is the monthly curtailment, which reduces the remaining balance and brings the final payment to an earlier date (in this example nearly 10 years.) This is the more typical manner in which curtailments occur.

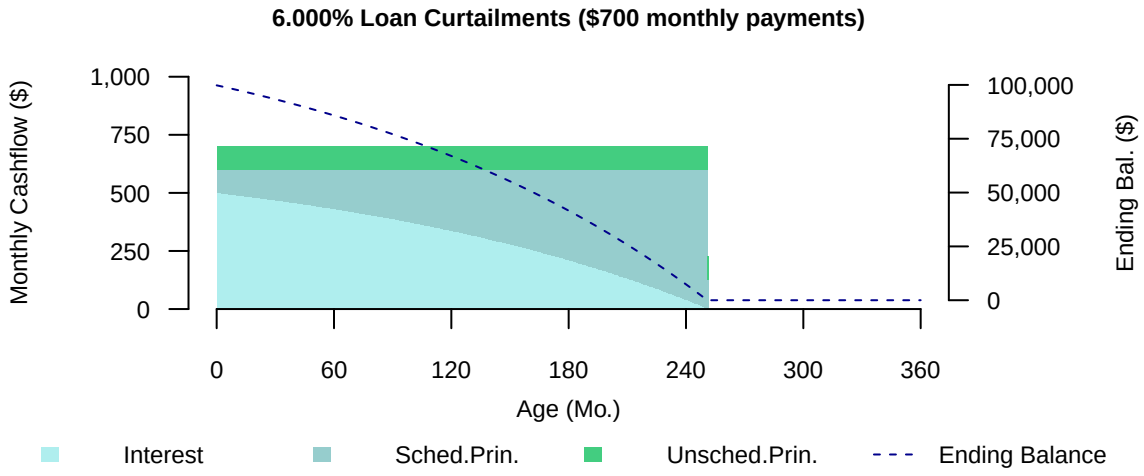

Figure 3: Borrower Rounding-Up Payment (Curtailment)

The below example shows a loan which curtails monthly as a percent of the scheduled remaining balance. This rate annualized is known as *conditional prepayment rate (CPR)*, described more fully in later sections. In contrast to the rounded-up monthly example above, in this example the rate of curtailment as a percentage of scheduled remaining balance is constant, but the *amount* of curtailments here decrease over time with balance.

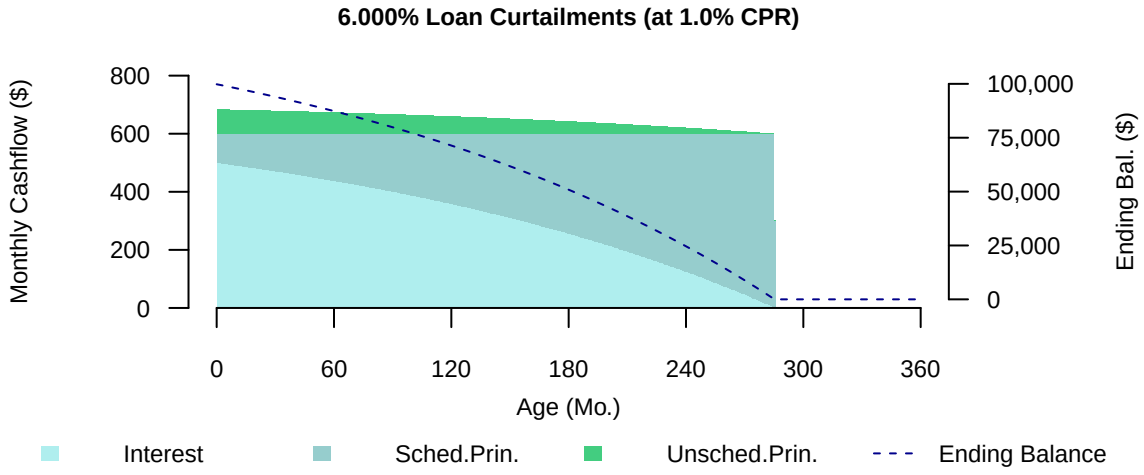

Figure 4: Borrower Curtailment as Percentage of Remaining Balance

Differences Modeling Cashflows and Prepayments of Loans and Pools

The growth of Mortgage-Based Securities (MBS) in the 1980s led to investors to evaluate mortgage loans as pools of loans. The borrower's option to refinance the loan is to the investor a short call option. Reflecting this exposure, MBS investor reporting attempts to describe historical prepayments and investor analysis requires assumptions to project future prepayments.

SF1999 describes the basic measure of pool prepayments is *single monthly mortality (SMM)* which is the monthly ratio of aggregate unscheduled principal payments to the pool's scheduled ending balance. More

commonly, prepayments are specified as a *conditional prepayment rate (CPR)* which is the compounded annualized SMM.

When SMM and CPRs are inferred from loan and pool cashflows, these are *historical prepayments.* When projecting future cashflows of a pool from a point in time to full paydown, SMMs and CPRs are *projected prepayments.* When a quantitative model is used to determine the borrower incentive to prepay from rates and other macroeconomic factors, the SMMs and CPRs are referred to as *model (projected) prepayments.*

Projected prepayments almost always utilize the *MBS convention.* The CPR assumption used to project pool cashflows abstracts the mortgage pool into a homogeneous asset which any proportion of remaining balance may be prepaid. This abstraction does not recognize individual loans within the pool. Rather, it assumes the pools consist of a very large number of infinitesimally sized loans. Actual behavior is that individual borrowers decide on a monthly basis to prepay their loans. The convention adopted by MBS investors assumes continuous prepayments to estimate the discrete behavior of actual loan pools. The difference between the continuous and discrete view is minimized when pools consist of the large number of loans (>250 loan count) typically required for MBS securitization.

To illustrate these distinctions, this section contrasts:

A. Loan Amortization and Prepayment

B. Pool Amortization and Prepayment under MBS Convention

C. Pool Amortization and Prepayment simulating discrete loans

Parameters used in Examples

- Original Term = 360
- Note rate = 6.00%
- Original Loan Balance = 100,000
- Monthly payment = 599.55
- Conditional Prepayment Rate (CPR) = 6.00%
- Pool Loan Count = 250

A. Loan Amortization and Prepayment

The single loan is modeled following its contractual terms.

CPRs for single loans are treated as:

1. Scheduled amortization (CPR == 0%)
2. Curtailments (0% < CPR < 100%) or
3. Payments in full (CPR == 100%)

Other characteristics of single loan amortization:

1. the Scheduled Payment does not change
2. Coupon does not change over time
3. Age and Remaining Term increment by 1 and -1 respectively every month
4. Nonzero CPR < 100% represents curtailments, which
    a. reduce the payment window by accelerating the final paydown to zero
    b. do not affect scheduled (level) payment

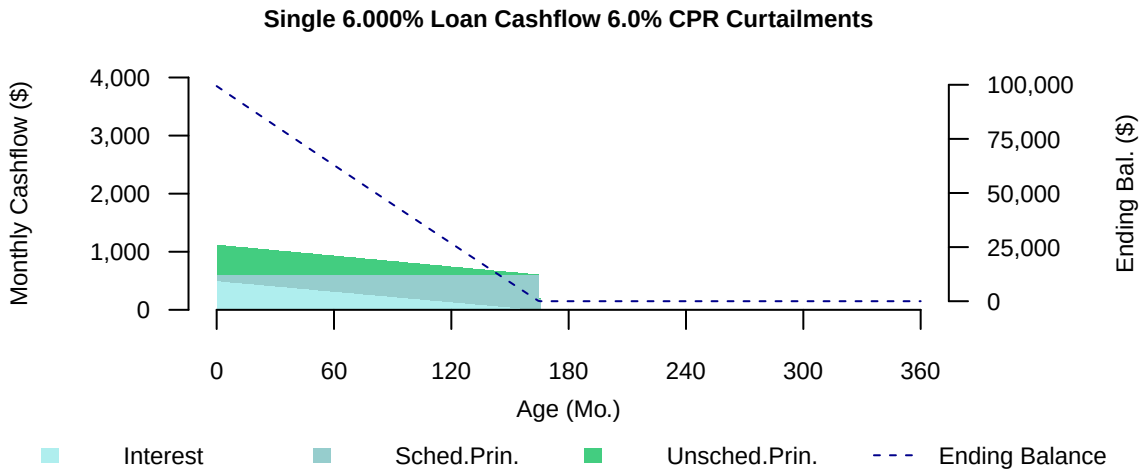

Figure 5: Loan Paydown

B. Pool Amortization and Prepayment under MBS Convention

The MBS convention assumes prepayments are *continuous*, implying pools consist of infinite number of loans each of *infinitesimal size*. There is no explicit modeling of loans and loan counts, though the latter may be inferred if average loan size is an attribute of the pool.

Balances and other quantities represent the aggregated values of pseudo-loans comprising the pool.

CPRs under MBS pool convention is treated as:

1. paid in full for a proportion of loans up to 100% (CPR $>$ 0%)
2. scheduled amortization (CPR == 0%)
3. There is no provision for curtailment.

Example:

1. Assume flat 6.00% CPR
2. 6.00% CPR is annualized equivalent of 0.51% Single Monthly Mortality (SMM).
3. Each month, 0.51% of the loans are assumuled to pay in full, reducing actual balances by that amount
4. The scheduled level payment is also reduced by 0.51% reflecting the paydowns.

Other characteristics of the pool:

1. the scheduled payment changes
2. coupon does not change over time
3. age and remaining term increment by 1 and -1 respectively every month
    a. age represents Weighted Average Loan Age (WALA)
    b. remaining term represents Weighted Average Maturity (WAM)
4. Prepayments do not affect the payment window

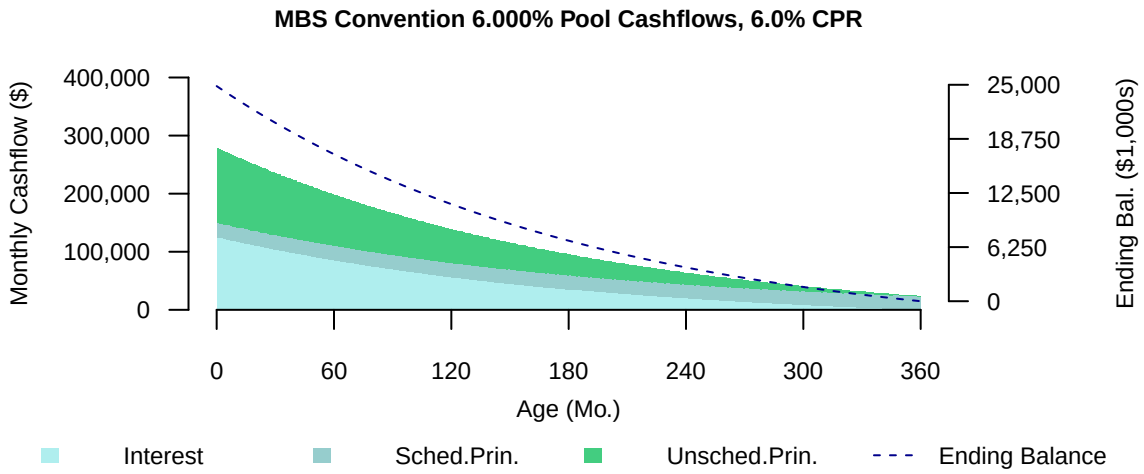

Figure 6: Pool Paydown using MBS Convention

C. Pool Amortization and Prepayment simulating discrete loans

The simulation models a pool as an aggregate of a *specific number* of loans of *specific size*. Though not captured in this simulation example, loans may have heterogeneous properties (different coupons, sizes, etc).

The simulation interprets an SMM as the probability for an individual loan being paid in full each month. There is no provision for curtailment.

Example:

1. The pool consists of 250 loans each of size 100,000.
2. Assume flat 6.00% CPR
3. 6.00% CPR is annualized equivalent of 0.51% Single Monthly Mortality (SMM).
4. Each month, each loan has a 0.51% probability of paying in full and 99.49% probability of amortizing.
5. The scheduled level payment is reduced by the scheduled level payments of those loans paying in full.

Table 1: 250 Loan Pool Simulation

User Time (sec)	System Time (ec)	Elapsed Time (sec)
0.798	0.001	0.799

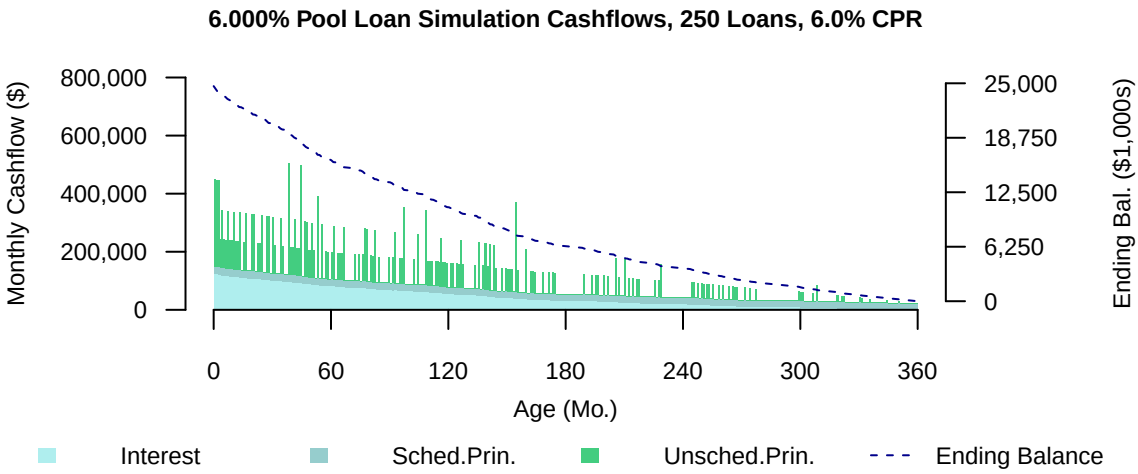

Figure 7: Pool Paydown by Loan Simulation

Comparing Loan Simulation against MBS Convention With higher loan counts, the discrete simulation approaches results of the MBS convention.

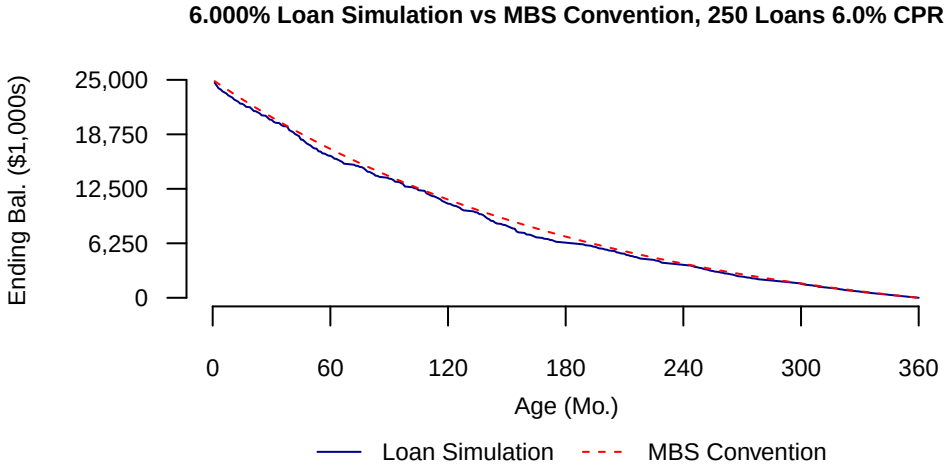

Figure 8: Comparing Loan Simulation vs. MBS Convention

Observed (Historical) Prepayments from Loan and Pool Cashflows

From loan and pool cashflows, observed CPRs may be calculated from pool balances and monthly unscheduled principal prayments. The pool loan simulation example will most closely resemble measurement of historical prepayments.

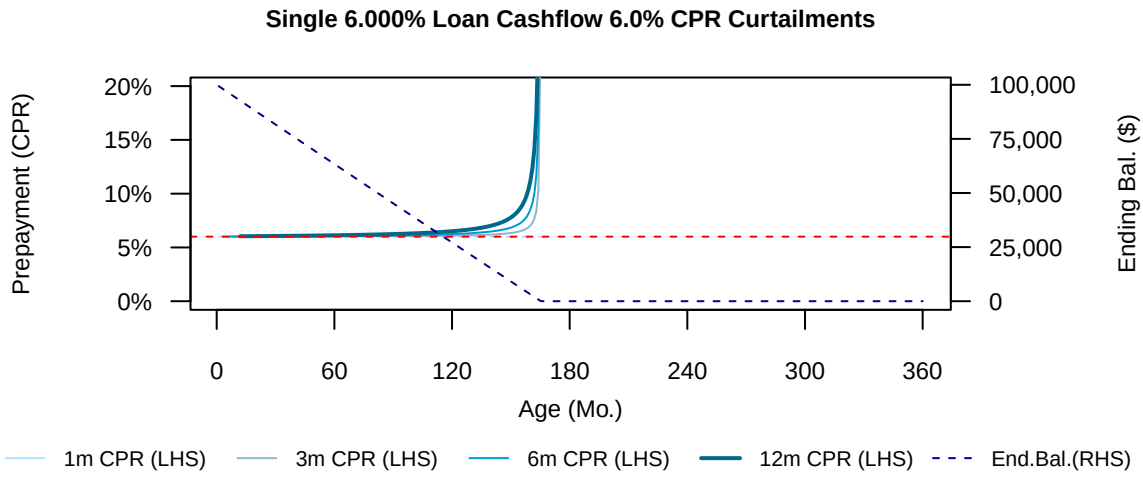

Figure 9: Loan Prepayment (Curtailment)

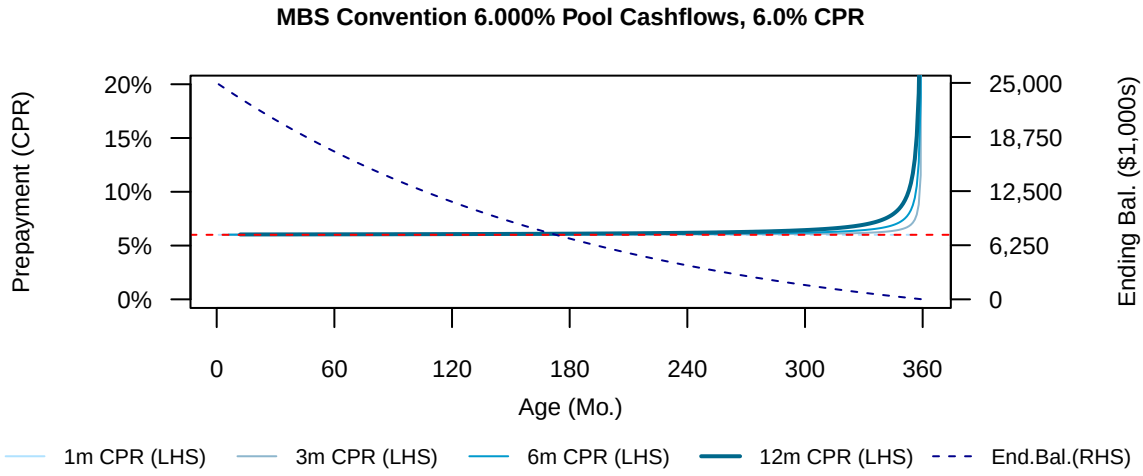

Figure 10: Pool Prepayment (paydown using MBS Convention)

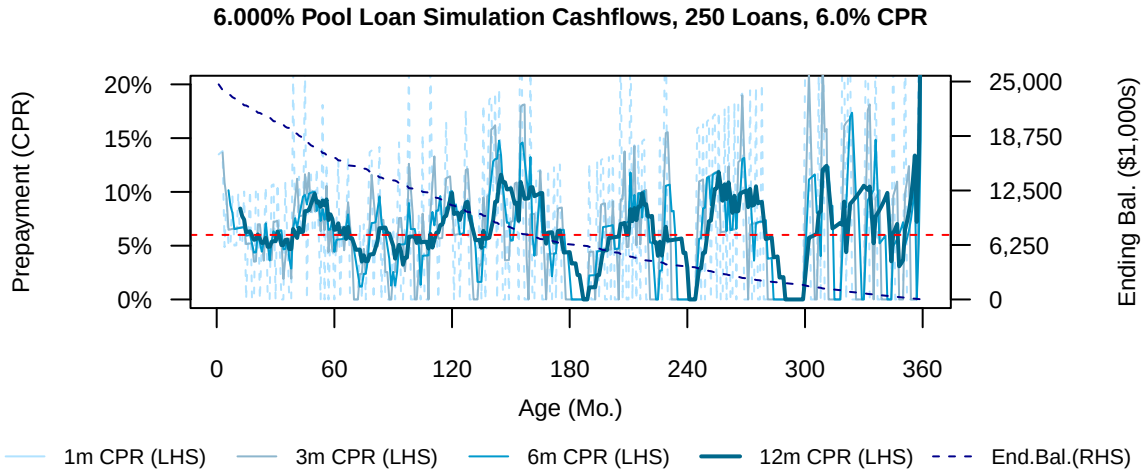

Figure 11: Pool Prepayment (paydown by Loan Simulation)

Loan Count and Pool Paydown Contrast small and large loan count pools paydowns through simulation.

Table 2: 100 Loan Pool Simulation

User Time (sec)	System Time (ec)	Elapsed Time (sec)
0.63	0.001	0.63

Table 3: 1000 Loan Pool Simulation

User Time (sec)	System Time (ec)	Elapsed Time (sec)
1.683	0.001	1.686

Pool Loan Count and Paydowns

Examples of small, medium and large pool paydowns are shown below, all modeling prepayments at a 6.00% CPR.

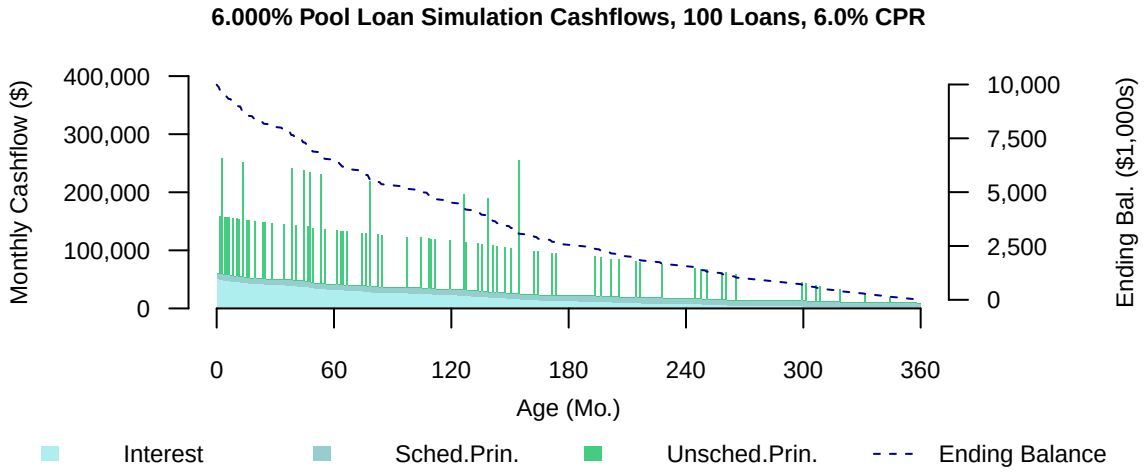

Figure 12: Small Pool Simulation

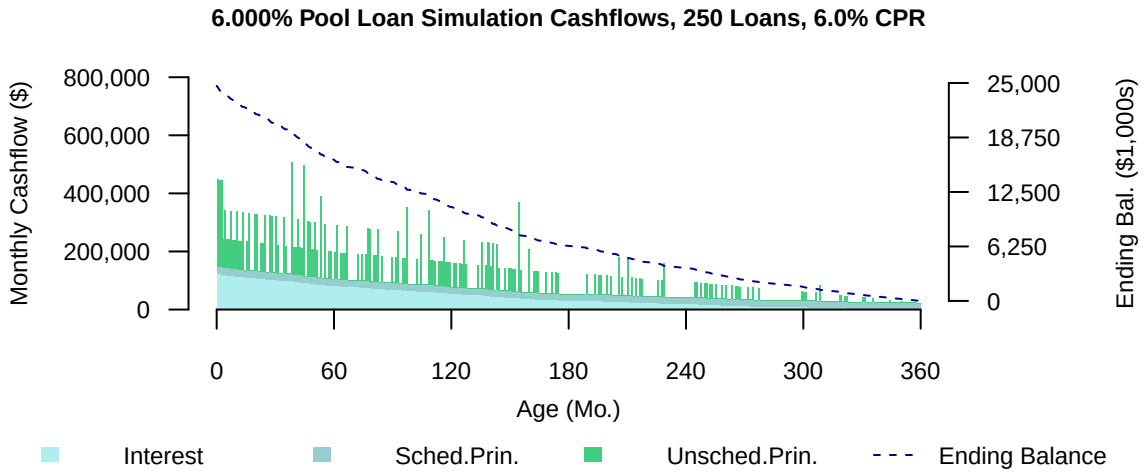

Figure 13: Medium Pool Simulation

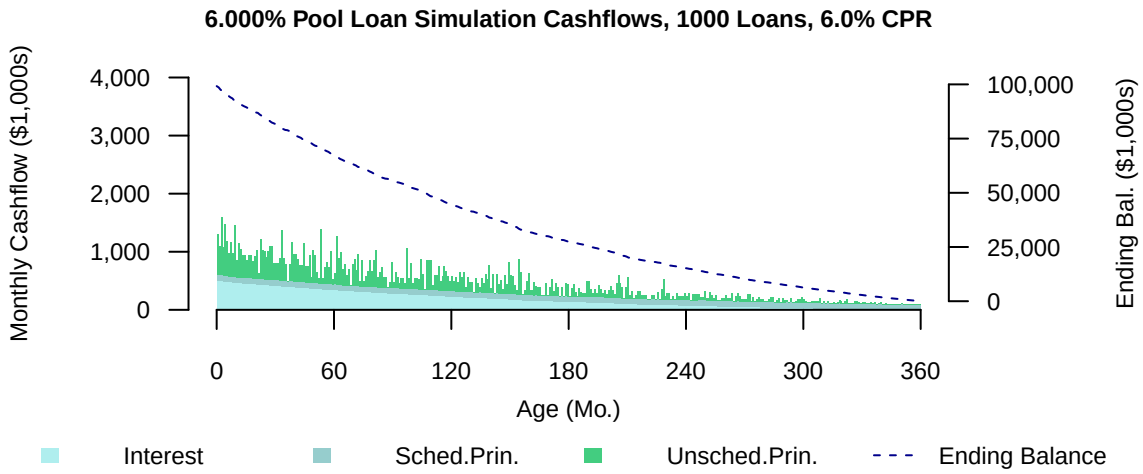

Figure 14: Large Pool Simulation

CPR derived from the small, medium and large simulations:

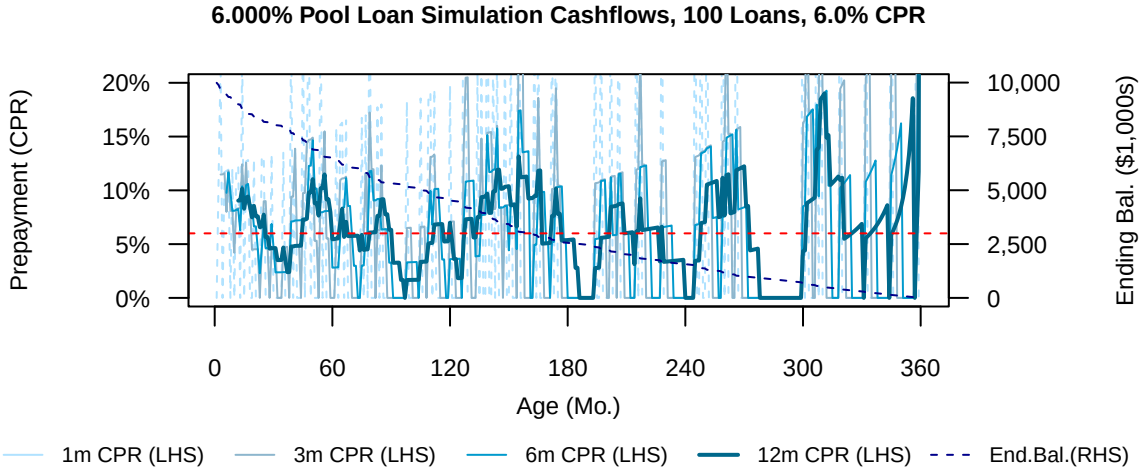

Figure 15: Small Pool Historical CPR

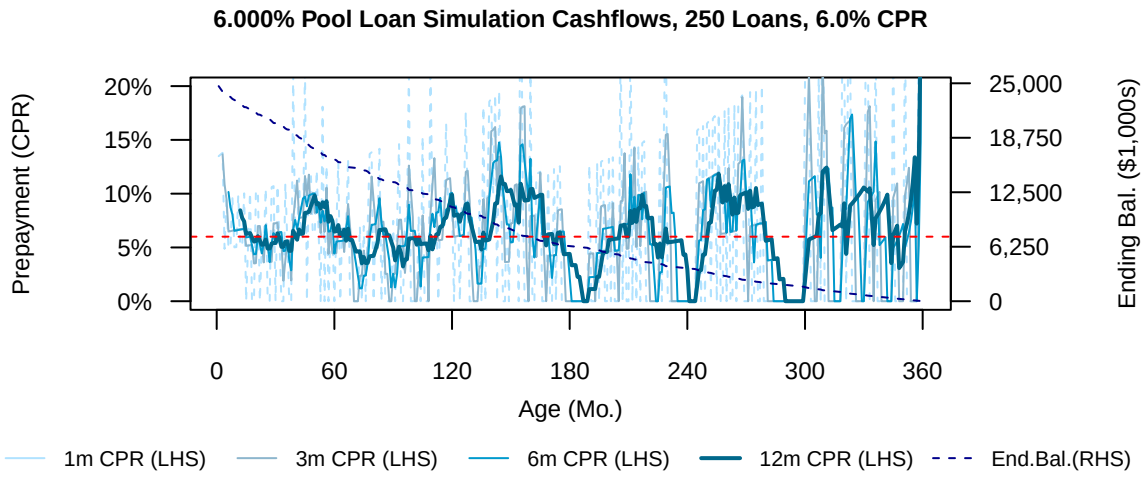

Figure 16: Medium Pool Historical CPR

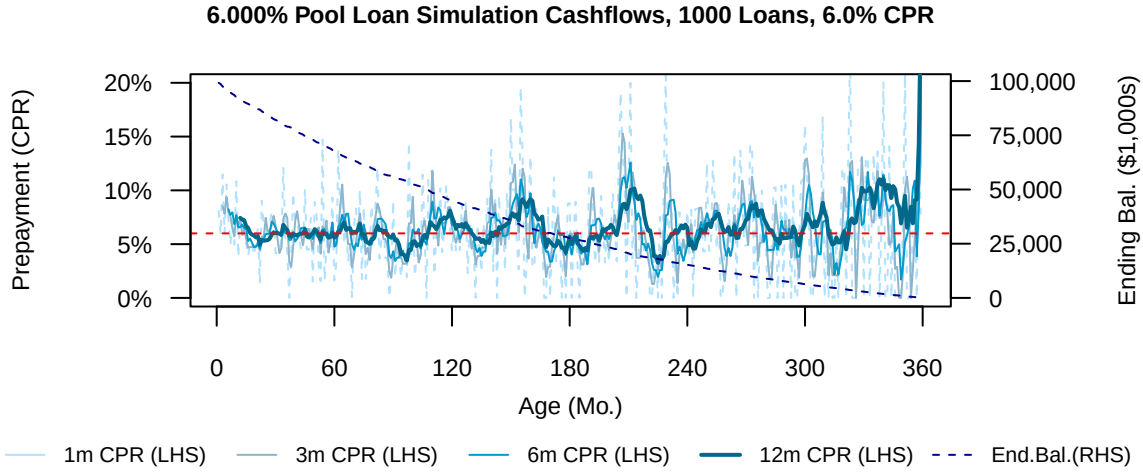

Figure 17: Large Pool Historical CPR

Part II. Simulation of Loan Pool Prepayments

Example C in the prior section showed single simulation of a pool of mortgage loans.

The simulations in this section execute 250 or more random trials of pool simulation in the manner of the last example. We extract historical prepayments from the simulations and observe the distribution of 1-, 3-, 6-, and 12-month CPR. For each prepayment term, the data are non-overlapping, i.e., the 12-month CPR values are spaced 12 months apart.

As is visible in the simulation CPR graphs shown under example C above, the variation of CPRs grows dramatically as the pool seasons and loan count drops. For this reason, we limit the CPRs in this analysis for the first n months, with the assumption that after 5-10 years, loans would be pooled together into a large seasoned cohort for the purpose of prepayment analysis.

We describe simulated data:

- Histogram
- Quantile-Quantile Plots
- Statistical Tests for Normality

In the statistical tests the null hypothesis H0 is that the data is normal. A p-value less than the significance threshold (0.01, 0.05) indicates evidence that H0 (normality) is not true.

Small Pool Statistics (100 Loans/Pool)

Table 4: 100 Loan Pool Simulation, 250 Trials

User Time (sec)	System Time (ec)	Elapsed Time (sec)
164.583	0.297	165.054

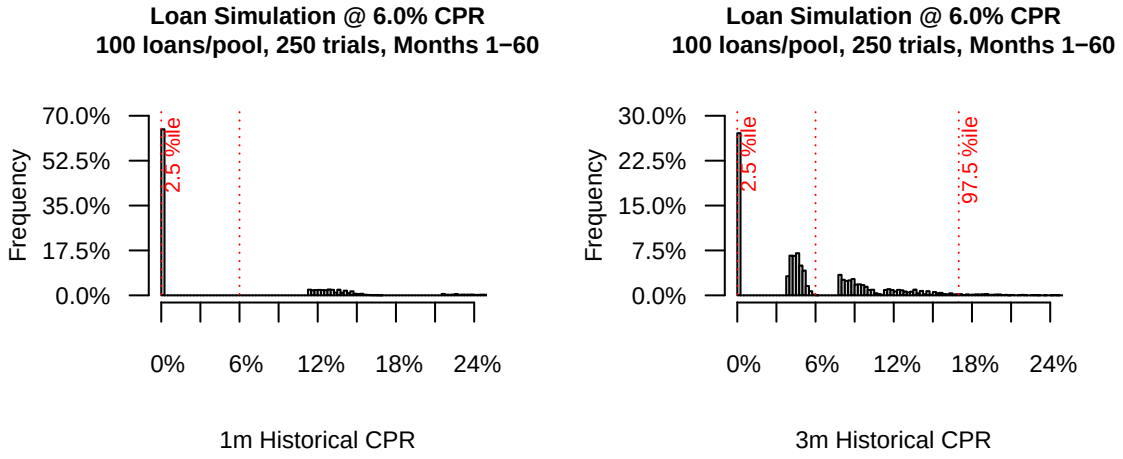

Figure 18: Small Pool 1m, 3m CPR

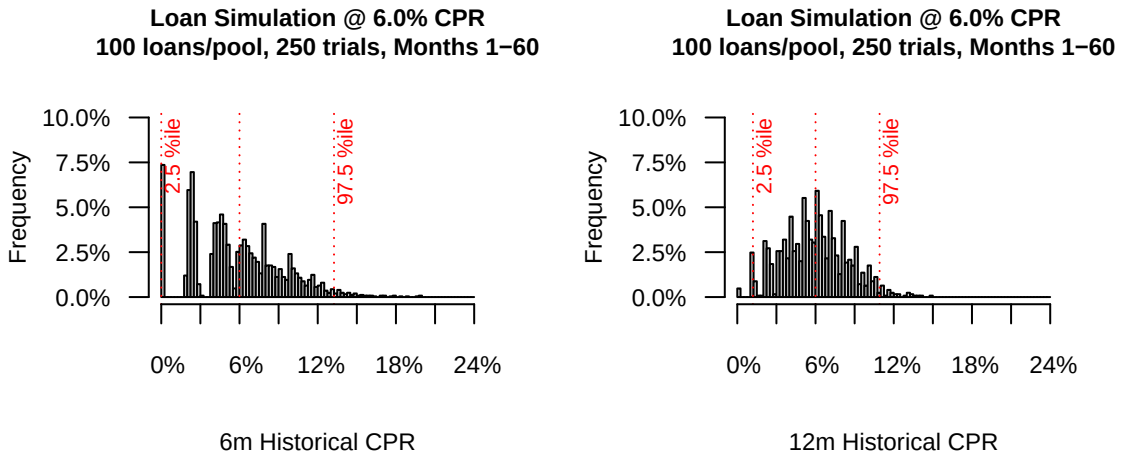

Figure 19: Small Pool 6m, 12m CPR

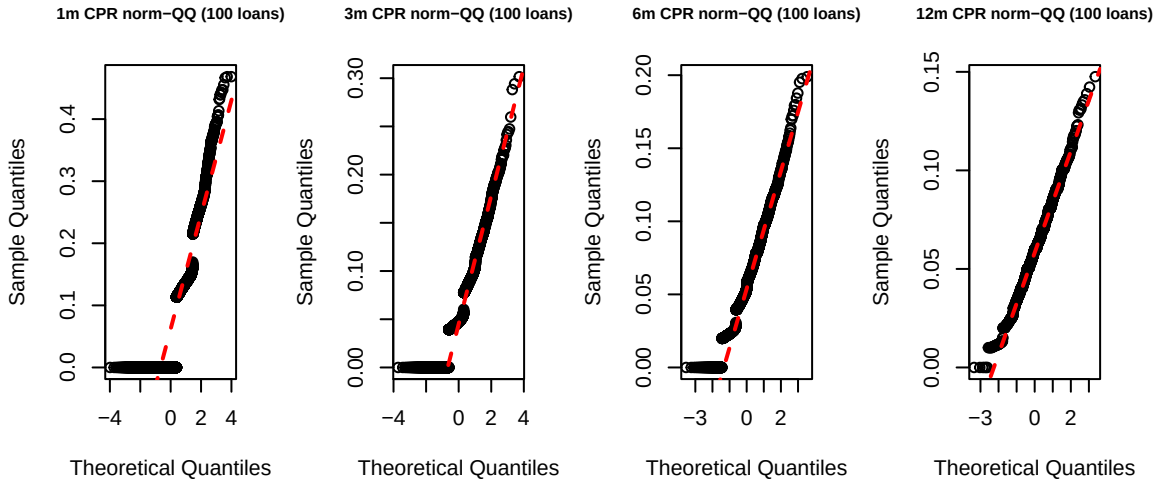

Figure 20: Small Pool CPR Q-Q

Table 5: 100 Loans/Pool Shapiro-Wilks (S-W), Kolmogorov-Smirnov (K-S), Anderson-Darling (A-D) tests

	S-W (W)	S-W p-value	K-S (D)	K-S p-value	A-D (A)	A-D p-value
1m CPR	0.7021	< 2e-16	0.3916	< 2e-16	679.0414	< 2e-16
3m CPR	0.9091	< 2e-16	0.1505	< 2e-16	135.9554	< 2e-16
6m CPR	0.9727	< 2e-16	0.0710	2.26e-11	14.9578	< 2e-16
12m CPR	0.9938	4.16e-05	0.0263	0.353	1.0988	0.00701

Medium Pool Statistics (250 Loans/Pool)

Table 6: 250 Loan Pool Simulation, 250 Trials

User Time (sec)	System Time (ec)	Elapsed Time (sec)
212.736	0.344	213.24

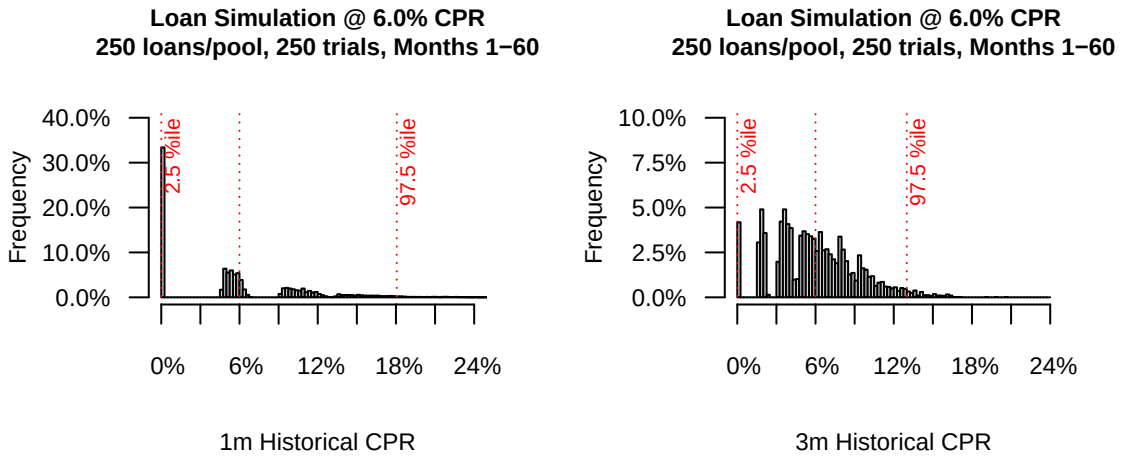

Figure 21: Medium Pool 1m, 3m CPR

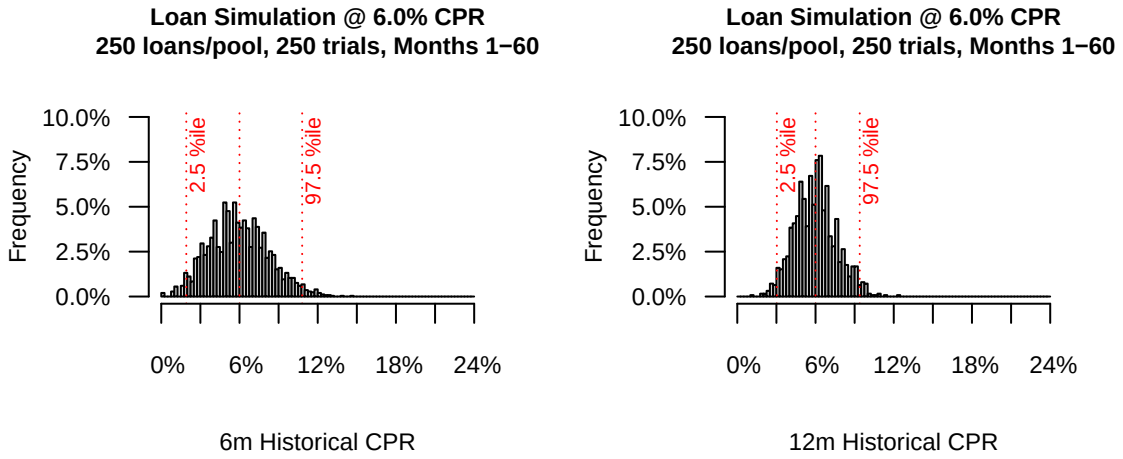

Figure 22: Medium Pool 6m, 12m CPR

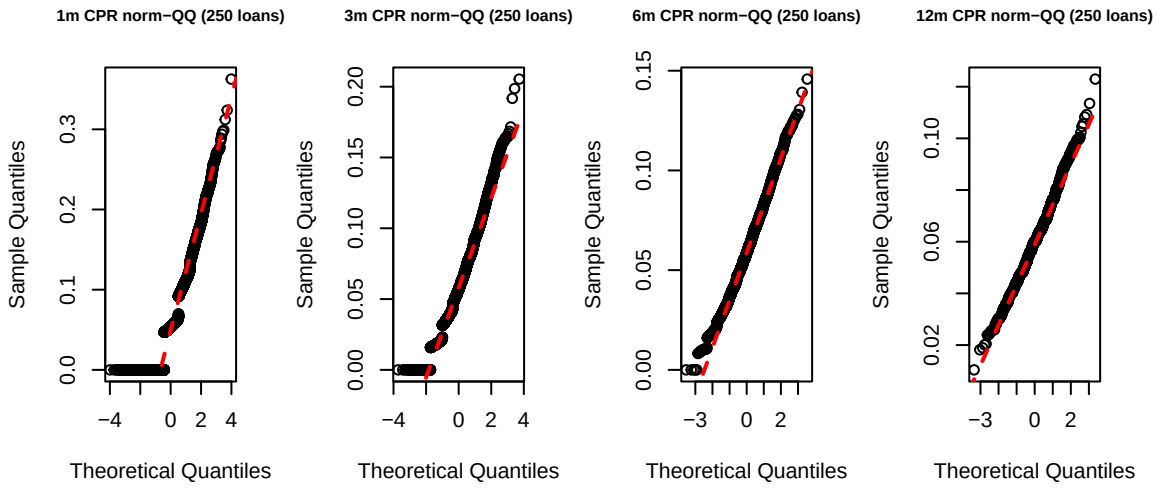

Figure 23: Medium Pool CPR Q-Q

Table 7: 250 Loans/Pool Shapiro-Wilks (S-W), Kolmogorov-Smirnov (K-S), Anderson-Darling (A-D) tests

	S-W (W)	S-W p-value	K-S (D)	K-S p-value	A-D (A)	A-D p-value
1m CPR	0.8786	< 2e-16	0.1953	< 2e-16	196.0840	< 2e-16
3m CPR	0.9799	< 2e-16	0.0499	3.07e-11	19.7356	< 2e-16
6m CPR	0.9938	8.99e-09	0.0291	0.0287	3.1510	6.72e-08
12m CPR	0.9952	0.000531	0.0376	0.0579	1.6260	0.000354

Medium-Large Pool Statistics (500 Loans/Pool)

Table 8: 500 Loan Pool Simulation, 250 Trials

User Time (sec)	System Time (ec)	Elapsed Time (sec)
288.518	0.804	289.657

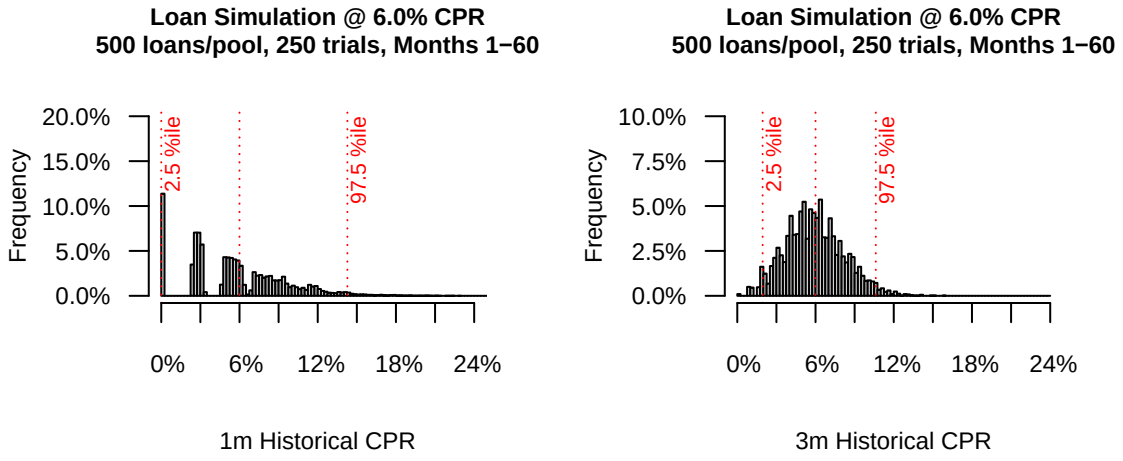

Figure 24: Medium/Large Pool 1m, 3m CPR

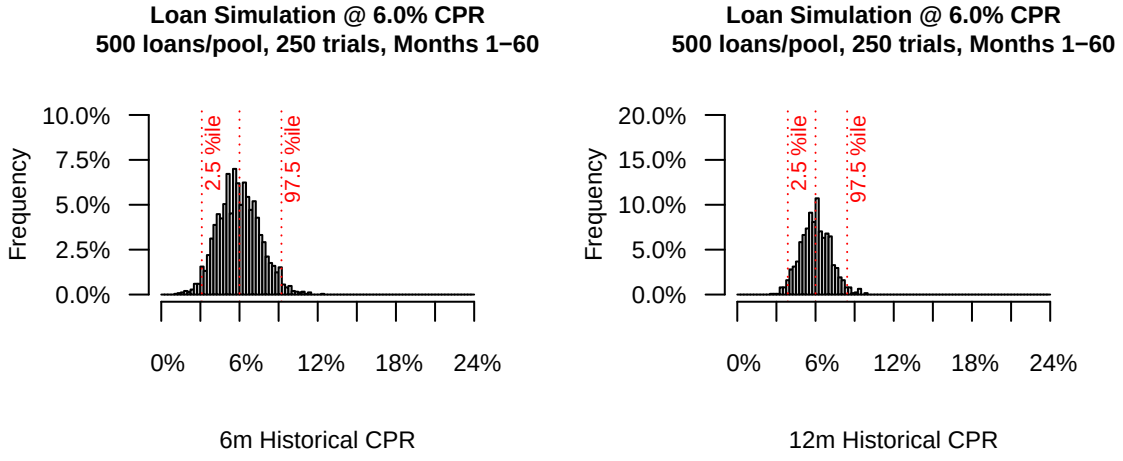

Figure 25: Medium/Large Pool 6m, 12m CPR

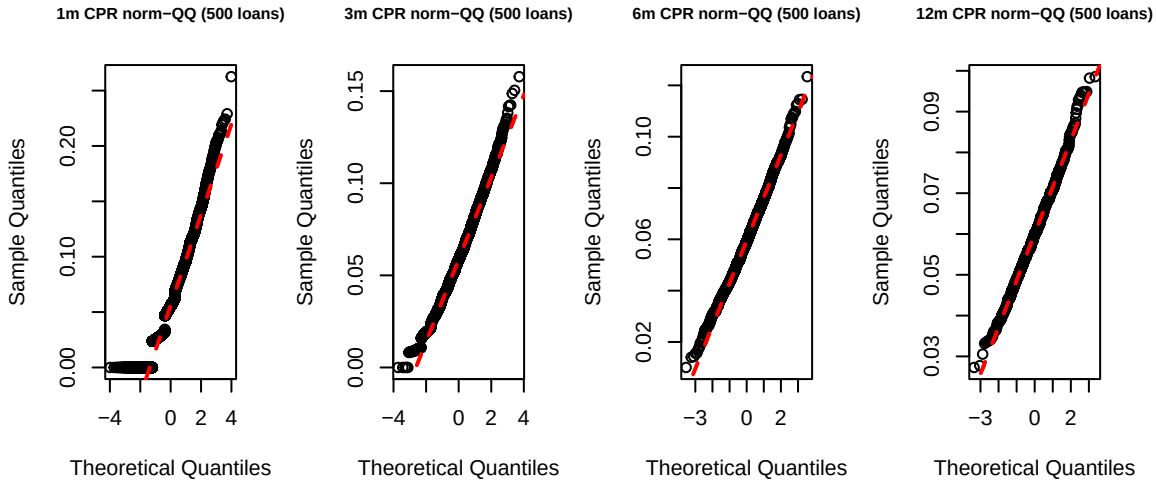

Figure 26: Medium/Large Pool CPR Q-Q

Table 9: 500 Loans/Pool Shapiro-Wilks (S-W), Kolmogorov-Smirnov (K-S), Anderson-Darling (A-D) tests

	S-W (W)	S-W p-value	K-S (D)	K-S p-value	A-D (A)	A-D p-value
1m CPR	0.9602	< 2e-16	0.1016	< 2e-16	46.4018	< 2e-16
3m CPR	0.9931	8.20e-15	0.0313	0.000109	6.4129	9.81e-16
6m CPR	0.9966	2.08e-05	0.0245	0.100618	1.9578	5.46e-05
12m CPR	0.9964	0.00554	0.0243	0.450969	0.6052	0.116

Large Pool Statistics (1000 Loans/Pool)

Table 10: 1000 Loan Pool Simulation, 250 Trials

User Time (sec)	System Time (ec)	Elapsed Time (sec)
439.294	0.711	440.545

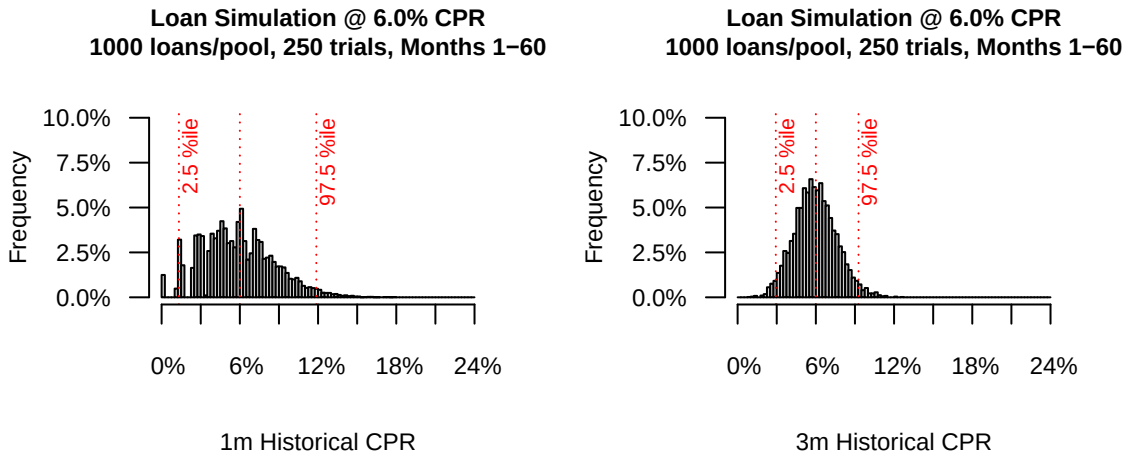

Figure 27: Large Pool 1m, 3m CPR

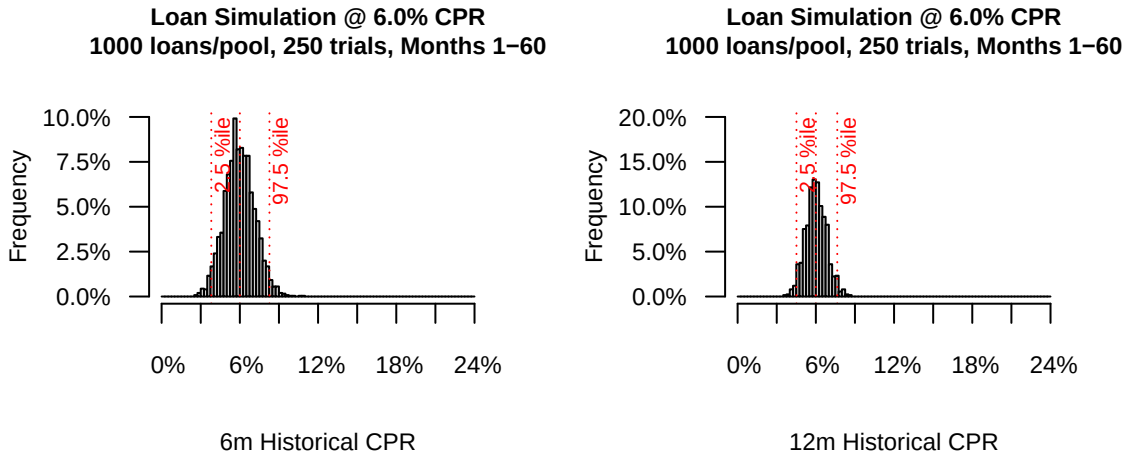

Figure 28: Large Pool 6m, 12m CPR

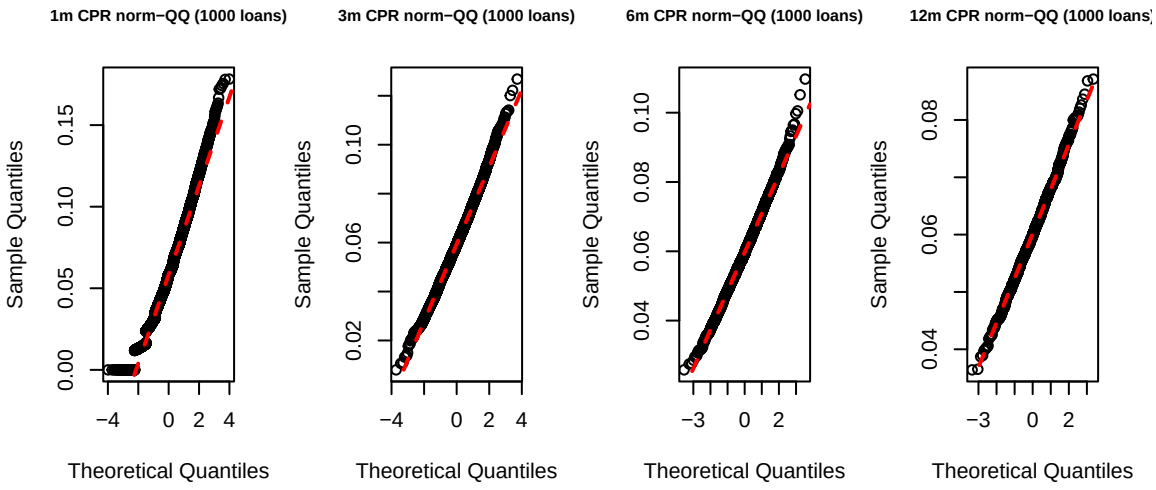

Figure 29: Large Pool CPR Q-Q

Table 11: 1000 Loans/Pool Shapiro-Wilks (S-W), Kolmogorov-Smirnov (K-S), Anderson-Darling (A-D) tests

	S-W (W)	S-W p-value	K-S (D)	K-S p-value	A-D (A)	A-D p-value
1m CPR	0.9885	< 2e-16	0.0349	9.91e-06	9.7944	< 2e-16
3m CPR	0.9970	2.08e-08	0.0174	0.0969	2.5319	2.16e-06
6m CPR	0.9979	0.00185	0.0224	0.1626	0.7944	0.0395
12m CPR	0.9982	0.22445	0.0185	0.7854	0.4472	0.2799

Appendix: Listing of R Code used in this Document

Amortization Functions

A. Single Loan Amortization

```
# level.pay.calc() ================================================================
# returns level (scheduled) payment given term to maturity and note.rate normalized to
# orig balance == 1.0
# rate of 0.0 is a special case, each month pays 1/rem.term of balance
#
level.pay.calc<-function(rem.term,monthly.rate)
{
  if(monthly.rate==0.0) { # special case of rate==0.0
    monthly.payment<-1/rem.term
  }
  else {
    monthly.payment<-(monthly.rate*(1+monthly.rate)^rem.term)/((1+monthly.rate)^rem.term-1)
  }
  return(monthly.payment)
}# level.pay.calc()
```

```
# amort.loan.cf(orig.term,note.rate,orig.bal,cpr=c(NA),mo.borrower.pay=NA) ===============
#
# amortize single loan incorporating prepayments
# orig.term = mortgage term in months
# note rate = borrower interest, e.g., 0.06 == 6% annual
# orig.bal = borrower original balance
# cpr = CPR vector of size 1:orig.term
#   CPR == 1 signifies paid-in-full
#   0 < CPR < 1 are curtailments
# mo.borrower.pay = borrower sets his own monthly level payment higher
#   than the scheduled payment, e.g., rounding to the next $100 or $1000. when
#   mo.borrower.pay is passed in, then cpr is not used
#
amort.loan.cf<-function(orig.term,note.rate,orig.bal,cpr=c(NA),mo.borrower.pay=NA)
{
  monthly.rate<-note.rate/12
  level.pay<-level.pay.calc(orig.term,monthly.rate)*orig.bal    # derive sched borrower pay
  if(!is.na(mo.borrower.pay) & mo.borrower.pay<level.pay){      # error-check borrower pay
    print(sprintf("ERROR $%s mo.borrower.pay < $%s level.pay, ignoring",
          prettyNum(round(mo.borrower.pay,2),big.mark=",", scientific=FALSE),
          prettyNum(round(level.pay,2),big.mark=",", scientific=FALSE)
          ))
```

```
    mo.borrower.pay<-NA
  }

  if(is.na(cpr[1]) | !is.na(mo.borrower.pay)){
    smm<-replicate(orig.term,0)
  } else {
    smm<-1-(1-cpr)^(1/12)
  }
  # initialize data.table for output
  cf<-data.table(mon=1:orig.term)
  cf[,bbal:=0]
  cf[1,bbal:=orig.bal]
  cf[,int:=0]
  cf[,sched.prin:=0]
  cf[,unsched.prin:=0]
  cf[,prin:=0]
  cf[,eschedbal:=0]
  cf[,ebal:=0]
  # begin amortization
  mo.bbal<-orig.bal
  mo.unsched.prin<-0
  for(m in 1:orig.term){  # unfortunately we must use loop as each month depends on results
                          # from prior month (can't vectorize)
    mo.int<-mo.bbal * monthly.rate        # pay interest first
    mo.sched.prin<-min(mo.bbal,level.pay - mo.int)     # remainder of payment is sched prin
    mo.eschedbal<-mo.bbal - mo.sched.prin # balance after sched prin is sched bal
    if (!is.na(mo.borrower.pay)){
      mo.unsched.prin<-mo.borrower.pay - level.pay  # unsched prin is % of sched bal
    } else {
      mo.unsched.prin<-mo.eschedbal*smm[m]  # unsched prin is % of sched bal
    }
    mo.prin<-mo.sched.prin + mo.unsched.prin # total prin is sched + unsched prin
    mo.ebal<-mo.bbal - mo.prin            # ending actual balance
    # copy results to data.matrix as actual borrower cashflows, rounding to $0.01
    cf[m,bbal:=round(mo.bbal,2)]
    cf[m,int:=round(mo.int,2)]
    cf[m,sched.prin:=round(mo.sched.prin,2)]
    cf[m,unsched.prin:=round(mo.unsched.prin,2)]
    cf[m,prin:=round(mo.prin,2)]
    cf[m,eschedbal:=round(mo.eschedbal,2)]
    cf[m,ebal:=round(mo.ebal,2)]
    if(mo.ebal<=0.005) break
    mo.bbal<-mo.ebal                    # for next month
  }
  return (cf)
}# amort.loan.cf()
```

B. MBS Convention Pool Amortization

```
# amort.pool.cf() ================================================================
#
# Amortize loan pool using MBS convention which models the pool balance and prepayments as
# continuous. The pool would theoretically consisting of many loans of 0.01 each
```

```
# CPR in a given month represents the % of loans that PAY IN FULL that month
# there is no provision for curtailments
#
amort.pool.cf<-function(orig.term,note.rate,orig.bal,cpr=c(NA))
{
  monthly.rate<-note.rate/12
  if(is.na(cpr[1])){
    smm<-replicate(orig.term,0)
  } else {
    smm<-1-(1-cpr)^(1/12)
  }
  # initialize data.table for output
  cf<-data.table(mon=1:orig.term)
  cf[1,bbal:=orig.bal]
  cf[,int:=0]
  cf[,sched.prin:=0]
  cf[,unsched.prin:=0]
  cf[,prin:=0]
  cf[,eschedbal:=0]
  cf[,ebal:=0]
  mo.bbal<-orig.bal
  mo.unsched.prin<-0
  for(m in 1:orig.term){ # must loop as each month depends on prior month
    level.pay<-level.pay.calc(orig.term+1-m,monthly.rate)*mo.bbal  # recalc level pay based
                                          # on act bal, implying SMM paid in full
    mo.int<-mo.bbal * monthly.rate        # pay interest first
    mo.sched.prin<-level.pay - mo.int     # remainder is sched prin
    mo.eschedbal<-mo.bbal - mo.sched.prin # bal after sched prin is sched bal
    mo.unsched.prin<-mo.eschedbal*smm[m]  # unsched prin is % of sched bal
    mo.prin<-mo.sched.prin + mo.unsched.prin # total prin = sched + unsched prin
    mo.ebal<-mo.bbal - mo.prin            # actual bal
    # copy results to data.matrix as remitted pool cashflows, rounding to $0.01
    cf[m,bbal:=round(mo.bbal,2)]
    cf[m,int:=round(mo.int,2)]
    cf[m,sched.prin:=round(mo.sched.prin,2)]
    cf[m,unsched.prin:=round(mo.unsched.prin,2)]
    cf[m,prin:=round(mo.prin,2)]
    cf[m,eschedbal:=round(mo.eschedbal,2)]
    cf[m,ebal:=round(mo.ebal,2)]
    if(mo.ebal<0.005) break             # half-cent tolerance
    mo.bbal<-mo.ebal # for next month
  }
  return (cf)
} # amort.pool.cf
```

C. Loan Simulation Pool Amortization

```
# simulate.loan.pool(orig.term,note.rate,orig.bal,scalar.cpr,loan.count) ----------------
#
# NOTE: This is a "slow" version of simulate.loan.pool() see NOTE below
# convert prepayments to probabilities, CPR is probability a loan will prepay in full in
# the year, SMM the probabilty a loan will prepay in the month. For each loan, randomly
# simulate a projected time series of binary payment outcomes, 1.0 for paid in full
```

```r
# and 0.0 for amortize. This series translates directly to CPR (or SMM, as the 0.0 and 1.0
# are equivalent CPR and SMM) vectors which are passed into the amort.loan.cf() function.
# Each loan cashflow is aggregated into a pool cashflow. The resulting pool cashflow of
# discrete loan prepayments more closely resembles observable pool cashflows than the
# continuous cashflows using the MBS pool convention, but converges with theMBS pool
# convention as loan count exceeds 250 or more.  This simulation offers is no provision for
# curtailment.
#
# NOTE: This is a "slow" version of simulate.loan.pool() which
# amortizes every loan brute-force, which can theoretically accommodate loans with unique
# characteristics
#
simulate.loan.pool<-function(orig.term,note.rate,orig.bal,scalar.cpr,loan.count){
  cpr<-replicate(orig.term,scalar.cpr)
  smm<-1-(1-cpr)^(1/12)
  #schedcf<-amort.loan.cf(orig.term,note.rate,orig.bal,cpr=c(NA)) # sched
  for(loan.no in 1:loan.count){
    pifcpr<-runif(n=orig.term)  # uniformly random for each month 1:orig.term
    pifcpr<-(pifcpr<smm)*1      # interpret smm as a probability of monthly PIF
    cf1<-amort.loan.cf(orig.term,note.rate,orig.bal,pifcpr)

    # accumulate individual loan cashflows to the pool level
    if(loan.no==1){
      cfsim<-cf1
    } else {
      cfsim<-cfsim+cf1
    }
  }
  cfsim$mon<-cfsim$mon/loan.count
  return(cfsim)
}# simulate.loan.pool()
```

```r
# simulate.loan.pool.f(orig.term,note.rate,orig.bal,scalar.cpr,loan.count) ===============
#
# NOTE: This is a "fast cheater" version of simulate.loan.pool.f() see NOTE below
# convert prepayments to probabilities, CPR is probability a loan will prepay in full in the
# year, SMM the probabilty a loan will prepay in the month. For each loan, randomly simulate
# a projected time series of binary payment outcomes, 1.0 for paid in full and 0.0 for
# amortize. This series translates directly to CPR (or SMM, as the 0.0 and 1.0 are equivalent
# CPR and SMM) vectors whichare passed into the amort.loan.cf() function. Each loan cashflow
# is aggregated into a pool cashflow.
# The resulting pool cashflow of discrete loan prepayments more closely resembles observable
# pool cashflows than the continuous cashflows using the MBS pool convention, but converges
# with the MBS pool convention as loan count exceeds 250 or more.
# This simulation offers is no provision for curtailment.
#
# NOTE: This is a "fast cheater" version of simulate.loan.pool.f() which assumes every loan
#       is identical (term,note rate,orig.face). At this time (May 2025) the "fast" version
#       return the same output as the "slow" version, but about 50 times faster.
# The fast cheater:
#   amortizes loan only once per call (to data.table schedcf)
#   for each loan generates random binary smm vector simulating scalar.cpr
#   derives PFI month (idx) from smm vec
#   deep-copies schedcf to cf1
```

```
#   and manually sets unsched.prin to ebal at month idx
#   zeroes out following months of cf1
#   accumulates pool cfs same as in the slow version
#
simulate.loan.pool.f<-function(orig.term,note.rate,orig.bal,scalar.cpr,loan.count){
  cpr<-replicate(orig.term,scalar.cpr)
  smm<-1-(1-cpr)^(1/12)
  schedcf<-amort.loan.cf(orig.term,note.rate,orig.bal,cpr=c(NA)) # scheduled cashflows ONCE
  for(loan.no in 1:loan.count){
    pifcpr<-runif(n=orig.term)  # uniformly random for each month 1:orig.term
    pifcpr<-(pifcpr<smm)*1      # interpret smm as a probability loan will PFI this month
    idx<-match(1.0,pifcpr) # PIF month (only first occurrence)
    if(!is.na(idx)&idx<orig.term){ # idx not NA and not last CF
      cf1<-copy(schedcf) # make a deep copy
      cf1[idx,unsched.prin:=ebal] # PIF
      cf1[idx,ebal:=0.0]          # PIF
      # zero out following CF attributes in subsequent periods to term
      cf1[(idx+1):orig.term,c("bbal","int","sched.prin","unsched.prin",
                              "prin","eschedbal","ebal"):=0.0]
    } else {
      cf1<-schedcf # if no prepay then use sched bal; no need to deep copy
    }
    # accumulate individual loan cashflows to the pool level
    if(loan.no==1){
      cfsim<-cf1
    } else {
      cfsim<-cfsim+cf1
    }
  }
  cfsim$mon<-cfsim$mon/loan.count
  return(cfsim)
}# simulate.loan.pool()
```

Observed (historical) CPR Computation

```
# calc.hist.cpr(cf) ==============================================================================
#
# given a pool (or loan) cashflow calculate 1-,3-,6- and 12-month CPR
# cashflow cf is a data.table minimally containg the following 4 attributes for each month
#
# mon   bbal      ebal      unsched.prin
# 1     5000000.0 4960467.5  25643.63
# 2     4960467.5 4921209.3  25440.68
# ...
# mon          = timeseries month (index):   cf[1,] represents the first cashflow
# bbal         = beginning (actual) balance: cf[1,bbal] the original balance of loan or pool
# ebal         = ending (actual) balance:    cf[1,ebal] the remaining bal after 1st payment
# unsched.prin = unscheduled principal:      cf[1,unsched.prin] borrower payment excess of
#                                             sched pay incorporates payments of full
#                                             repayments and curtailments
# implementation note: data.table() allows this to be vectorized nicely
#
calc.hist.cpr<-function(cf){
```

```
  orig.term<-nrow(cf)
  hist.cpr<-data.table(mon=1:orig.term)
  hist.cpr$bbal<-cf$bbal
  hist.cpr$ebal<-cf$ebal
  hist.cpr$unsched.prin<-cf$unsched.prin
  hist.cpr[,wam:=orig.term-mon]
  # collect trailing 1,3,6,12-month unscheduled principal
  # this is the only section requiring shift(1..11)
  hist.cpr[bbal>0,uprin01:=unsched.prin]
  hist.cpr[bbal>0,uprin03:=uprin01+shift(unsched.prin,1)+shift(unsched.prin,2)]
  hist.cpr[bbal>0,uprin06:=uprin03+shift(unsched.prin,3)+shift(unsched.prin,4)+
             shift(unsched.prin,5)]
  hist.cpr[bbal>0,uprin12:=uprin06+shift(unsched.prin,6)+shift(unsched.prin,7)+
             shift(unsched.prin,8)+shift(unsched.prin,9)+
             shift(unsched.prin,10)+shift(unsched.prin,11)]
  # determine 1-,3-,6-,12-mo scheduled ending balances for month m
  # e.g., hist.cpr[m,sched.ebal01] is the sched bal projected from hist.cpr[m-1,ebal]
  #       sched.ebal01 for month m is m's ending balance plus m's unsched prin
  #       sched.ebal03 for month m is m's ending balance plus unsched prin in month m,
  #                                       m-1, m-2 (3 mos)
  #       etc.
  hist.cpr[,sched.ebal01:=ebal+uprin01]
  hist.cpr[,sched.ebal03:=ebal+uprin03]
  hist.cpr[,sched.ebal06:=ebal+uprin06]
  hist.cpr[,sched.ebal12:=ebal+uprin12]
  # SMM single monthly mortality
  # SMM is defined as unscheduled principal divided by scheduled balance ([SF1999] p. SF-5).
  # when calculating 3-,6- & 12-mo SMM, decompound total unsched over period to 1m equivalents
  # such that, e.g. smm03[m] ~= 1-(1-smm01[m])(1-smm01[m-1])(1-smm01[m-2])
  # we signify ~= (approximate) because a constant 1m SMM for example is applied to balances
  # which are amortizing as well as prepaying
  hist.cpr[,smm01:=uprin01/sched.ebal01]
  hist.cpr[,smm03:=1-(1-uprin03/sched.ebal03)^(1/3)]
  hist.cpr[,smm06:=1-(1-uprin06/sched.ebal06)^(1/6)]
  hist.cpr[,smm12:=1-(1-uprin12/sched.ebal12)^(1/12)]
  # annualize SMMs to CPRs
  hist.cpr[,cpr01:=1-(1-smm01)^12]
  hist.cpr[,cpr03:=1-(1-smm03)^12]
  hist.cpr[,cpr06:=1-(1-smm06)^12]
  hist.cpr[,cpr12:=1-(1-smm12)^12]
  #
  return(hist.cpr)
}# calc.hist.cpr()
```

Loan Parameters used in this Document

```
loan.count<-250
small.loan.count<-100
big.loan.count<-1000
```

Basic Loan Amortization Mechanics

Loan Scheduled Payments

```
cpr.eg<-0
cprsingle<-replicate(orig.term,cpr.eg)
cfloan<-amort.loan.cf(orig.term,note.rate,orig.bal,cprsingle)
title<-sprintf("%3.3f%% Loan Scheduled Amortization (%3.1f%% CPR)",note.rate*100,cpr.eg*100)
plot.cfs(cfloan,title)
#knitr::knit_exit()
```

Loan Repayment (Payment in Full)

```
pif.mon<-354
cprsingle<-replicate(orig.term,0)
cprsingle[pif.mon]<-1.0
cfloan<-amort.loan.cf(orig.term,note.rate,orig.bal,cprsingle)
title<-sprintf("%3.3f%% Loan Payment in Full at month %3d",note.rate*100, pif.mon)
plot.cfs(cfloan,title)
```

Loan Curtailment

```
cpr.eg<-0.01
cprsingle<-replicate(orig.term,cpr.eg)
cfloan<-amort.loan.cf(orig.term,note.rate,orig.bal,cprsingle)
title<-sprintf("%3.3f%% Loan Curtailments (at %3.1f%% CPR)",note.rate*100,cpr.eg*100)
plot.cfs(cfloan,title)
```

Run Cashflows

A. Loan

```
cfloan<-amort.loan.cf(orig.term,note.rate,orig.bal,cpr)
title<-sprintf("Single %3.3f%% Loan Cashflow %3.1f%% CPR Curtailments",
               note.rate*100,scalar.cpr*100)
plot.cfs(cfloan,title)
```

B. Pool (MBS Convention)

```
pool.orig.bal<-orig.bal*loan.count
cfpool<-amort.pool.cf(orig.term,note.rate,pool.orig.bal,cpr)

title<-sprintf("MBS Convention %3.3f%% Pool Cashflows, %3.1f%% CPR",
               note.rate*100,scalar.cpr*100)
plot.cfs(cfpool,title)
```

C. Pool (Simulation)

```
# sim.trials(min.cutoff,max.cutoff,loan.count) ============================================
#
# run the pool simulation, extract historical CPR, return in list, suitable to be saved
# in data.table
sim.trials<-function(min.cutoff=1,max.cutoff,loan.count){
```

```
  cfsim<-simulate.loan.pool.f(orig.term,note.rate,orig.bal,scalar.cpr,loan.count)
  hist.cpr<-calc.hist.cpr(cfsim)
  months<-nrow(hist.cpr)
  return(list(
        list(hist.cpr[seq(pmax(min.cutoff,1),pmin(max.cutoff,months),by=1),cpr01]),
        list(hist.cpr[seq(pmax(min.cutoff,3),pmin(max.cutoff,months),by=3),cpr03]),
        list(hist.cpr[seq(pmax(min.cutoff,6),pmin(max.cutoff,months),by=6),cpr06]),
        list(hist.cpr[seq(pmax(min.cutoff,12),pmin(max.cutoff,months),by=12),cpr12]))
        #list(hist.cpr[min.cutoff:max.cutoff,cpr03]),
        #list(hist.cpr[min.cutoff:max.cutoff,cpr06]),
        #list(hist.cpr[min.cutoff:max.cutoff,cpr12]))
  )
}# sim.trials()
```

```
set.seed(666) # replicable in isolation
time<-system.time(
  cfsim<-simulate.loan.pool.f(orig.term,note.rate,orig.bal,scalar.cpr,loan.count)
  )
title<-sprintf("%d Loan Pool Simulation",loan.count)
print.time.stats(time,title=title)
```

```
title<-sprintf("%3.3f%% Pool Loan Simulation Cashflows, %d Loans, %3.1f%% CPR",
                note.rate*100,loan.count,scalar.cpr*100)
plot.cfs(cfsim,title)
```

```
title<-sprintf("%3.3f%% Pool Loan Simulation Cashflows, %d Loans, %3.1f%% CPR",
               note.rate*100,loan.count,scalar.cpr*100)
plot.hist.cpr(cfsim,title,baseline.cpr=scalar.cpr)
```

```
title<-sprintf("%3.3f%% Pool Loan Simulation Cashflows, %d Loans, %3.1f%% CPR",
               note.rate*100,small.loan.count,scalar.cpr*100)
plot.hist.cpr(small.cfsim,title,baseline.cpr=scalar.cpr)

cat('\n\n')
title<-sprintf("%3.3f%% Pool Loan Simulation Cashflows, %d Loans, %3.1f%% CPR",
               note.rate*100,loan.count,scalar.cpr*100)
plot.hist.cpr(cfsim,title,baseline.cpr=scalar.cpr)

cat('\n\n')
title<-sprintf("%3.3f%% Pool Loan Simulation Cashflows, %d Loans, %3.1f%% CPR",
               note.rate*100,big.loan.count,scalar.cpr*100)
plot.hist.cpr(big.cfsim,title,baseline.cpr=scalar.cpr)
```

CPR Statistics

```
plot.cpr.dist<-function(cprs,title=NA,desc=NA,lo=NA,hi=NA){
  binwidth=0.0025
  show.max.cpr<-scalar.cpr*4
  breaks<-seq(from=0.0,to=pmax(show.max.cpr,max(cprs,na.rm=TRUE)+binwidth),by=binwidth)
  show.breaks<-seq(from=0.0,to=show.max.cpr,by=0.03)
  h<-hist(cprs,breaks=breaks,plot=FALSE)
  h$counts<-h$counts/sum(h$counts)
  ymax<-ceiling(max(h$counts)*10)/10
```

```
  nudgex=show.max.cpr*1/100  # 1% of x axis
  nudgey=ymax*1/100

  plot(h,xlim=c(0,show.max.cpr),xaxt="n",xlab="",yaxt="n",ylim=c(0,ymax),
       main=title,cex.main=cex.size,cex.lab=cex.size)
  abline(v=scalar.cpr,lty=3,lwd=1,col=baseline.cpr.col)
  if(!is.na(lo)){
    abline(v=lo,lty=3,lwd=1,col=baseline.cpr.col)
    text(x=lo+nudgex,y=(1-nudgey)*ymax,"2.5 %ile",cex=cex.size*0.9,srt=90,
         adj=c(1,1),col=baseline.cpr.col)
  }
  if(!is.na(hi)){
    abline(v=hi,lty=3,lwd=1,col=baseline.cpr.col)
    text(x=hi+nudgex,y=(1-nudgey)*ymax,"97.5 %ile",cex=cex.size*0.9,srt=90,
         adj=c(1,1),col=baseline.cpr.col)
  }

  axis(1, at = show.breaks,cex.axis=cex.size,
              sprintf("%3.0f%%",show.breaks*100),cex.axis=cex.size,las=1)
  axis(2, at = seq(0,ymax,length.out=5),cex.axis=cex.size,
              sprintf("%3.1f%%",seq(0,ymax,length.out=5)*100),cex.axis=cex.size,las=1)
  mtext(desc,side=1,line=3.0,cex=cex.size)
  cat('\n\n')
}# plot.cpr.dist()
```

```
pool.simulation<-function(trials,min.cutoff=1,max.cutoff,loan.count){
  results<-data.table(trial=1:trials)
  time<-system.time(
    results[,c("cpr01","cpr03","cpr06","cpr12"):=
              sim.trials(min.cutoff,max.cutoff,loan.count),by="trial"]
  )

  # organize all CPR output for all trials
  cpr01<-do.call(c,results$cpr01)
  cpr03<-do.call(c,results$cpr03)
  cpr06<-do.call(c,results$cpr06)
  cpr12<-do.call(c,results$cpr12)
  # gather distribution statistics
  quants=c(0.005,0.025,0.05,0.50,0.95,0.975,0.995)
  return(list(
    trials=trials,
    loan.count=loan.count,
    min.cutoff=min.cutoff,
    max.cutoff=max.cutoff,
    cpr01=cpr01[!is.na(cpr01)],
    cpr03=cpr03[!is.na(cpr03)],
    cpr06=cpr06[!is.na(cpr06)],
    cpr12=cpr12[!is.na(cpr12)],
    cpr01.q=quantile(cpr01  ,quants,na.rm=TRUE),
    cpr03.q=quantile(cpr03  ,quants,na.rm=TRUE),
    cpr06.q=quantile(cpr06  ,quants,na.rm=TRUE),
    cpr12.q=quantile(cpr12  ,quants,na.rm=TRUE),
    time=time
  ))
```

```
}# pool.simulation()
```

```
show.pool.sim.results<-function(sim){
  trials<-sim$trials
  loan.count<-sim$loan.count
  min.cutoff<-sim$min.cutoff
  max.cutoff<-sim$max.cutoff
  time<-sim$time
  cpr01<-sim$cpr01
  cpr03<-sim$cpr03
  cpr06<-sim$cpr06
  cpr12<-sim$cpr12
  cpr01.q<-sim$cpr01.q
  cpr03.q<-sim$cpr03.q
  cpr06.q<-sim$cpr06.q
  cpr12.q<-sim$cpr12.q

  #title<-sprintf("%d Loan Pool Simulation, %d Trials",loan.count,trials)
  #print(print.time.stats(time=time,title=title))

  title<-sprintf("Loan Simulation @ %3.1f%% CPR\n%d loans/pool, %d trials, Months %d-%d",
                 scalar.cpr*100,loan.count,trials,min.cutoff,max.cutoff)
  par(mfrow=c(1,2))
  plot.cpr.dist(cpr01,title=title,desc="1m Historical CPR",
                lo=cpr01.q["2.5%"],hi=cpr01.q["97.5%"])
  plot.cpr.dist(cpr03,title=title,desc="3m Historical CPR",
                lo=cpr03.q["2.5%"],hi=cpr03.q["97.5%"])
  par(mfrow=c(1,2))
  plot.cpr.dist(cpr06,title=title,desc="6m Historical CPR",
                lo=cpr06.q["2.5%"],hi=cpr06.q["97.5%"])
  plot.cpr.dist(cpr12,title=title,desc="12m Historical CPR",
                lo=cpr12.q["2.5%"],hi=cpr12.q["97.5%"])
}# show.pool.sim.results()
```

```
norm.test<-function(dat,title){
  if(length(dat)>5000){
    dat<-sample(dat,5000,replace=FALSE)
  }
  print(sprintf("%s: length=%d",title,length(dat)))
  shapiro<-shapiro.test(dat)
  ks<-ks.test(dat,"pnorm")
  ad<-ad.test(dat)
  tab<-data.frame(
    method=c(shapiro$method,ad$method,)
  )
}

show.norm.stat<-function(sim,title){
  par(mfrow=c(1,4))
  qqnorm(sim$cpr01,main=sprintf(title,"1m CPR",sim$loan.count),cex.main=cex.size)
  qqline(sim$cpr01,col = "red",lwd=2,lty=2)
  cat('\n\n')
  qqnorm(sim$cpr03,main=sprintf(title,"3m CPR",sim$loan.count),cex.main=cex.size)
  qqline(sim$cpr03,col = "red",lwd=2,lty=2)
```

```
cat('\n\n')
#par(mfrow=c(1,2))
qqnorm(sim$cpr06,main=sprintf(title,"6m CPR",sim$loan.count),cex.main=cex.size)
qqline(sim$cpr06,col = "red",lwd=2,lty=2)
cat('\n\n')
qqnorm(sim$cpr12,main=sprintf(title,"12m CPR",sim$loan.count),cex.main=cex.size)
qqline(sim$cpr12,col = "red",lwd=2,lty=2)
cat('\n\n')

 dat<-sim$cpr01
if(length(dat)>5000){ # Shapiro-Wilks does not run if n>5000
  dat<-sample(dat,5000,replace=FALSE)
}
shapiro01<-shapiro.test(dat)
ks01<-ks.test(dat,"pnorm",mean=mean(dat),sd=sd(dat))
ad01<-ad.test(dat)

dat<-sim$cpr03
  if(length(dat)>5000){
  dat<-sample(dat,5000,replace=FALSE)
}
shapiro03<-shapiro.test(dat)
ks03<-ks.test(dat,"pnorm",mean=mean(dat),sd=sd(dat))
ad03<-ad.test(dat)

dat<-sim$cpr06
  if(length(dat)>5000){
  dat<-sample(dat,5000,replace=FALSE)
}
shapiro06<-shapiro.test(dat)
ks06<-ks.test(dat,"pnorm",mean=mean(dat),sd=sd(dat))
ad06<-ad.test(dat)

dat<-sim$cpr12
  if(length(dat)>5000){
  dat<-sample(dat,5000,replace=FALSE)
}

shapiro12<-shapiro.test(dat)
ks12<-ks.test(dat,"pnorm",mean=mean(dat),sd=sd(dat))
ad12<-ad.test(dat)

tab<-data.frame(
  cpr01.stats=c(shapiro01$statistic,shapiro01$p.value,
                ks01$statistic,ks01$p.value,
                ad01$statistic,ad01$p.value),
  cpr03.stats=c(shapiro03$statistic,shapiro03$p.value,
                ks03$statistic,ks03$p.value,
                ad03$statistic,ad03$p.value),
  cpr06.stats=c(shapiro06$statistic,shapiro06$p.value,
                ks06$statistic,ks06$p.value,
                ad06$statistic,ad06$p.value),
  cpr12.stats=c(shapiro12$statistic,shapiro12$p.value,
```

```
                ks12$statistic,ks12$p.value,
                ad12$statistic,ad12$p.value)

 )

  tab<-t(tab)
  row.names(tab)=c("1m CPR","3m CPR","6m CPR","12m CPR")
  colnames(tab)<-c("sw.stat","sw.pvalue","ks.stat","ks.pvalue","ad.stat","ad.pvalue")
  data.frame(tab) %>%
    mutate(sw.pvalue = cell_spec(format.pval(sw.pvalue,3), color = ifelse(sw.pvalue > 0.05, "red","blac
    mutate(ks.pvalue = cell_spec(format.pval(ks.pvalue,3), color = ifelse(ks.pvalue > 0.05, "red","blac
    mutate(ad.pvalue = cell_spec(format.pval(ad.pvalue,3), color = ifelse(ad.pvalue > 0.05, "red","blac
  kbl(escape=FALSE,booktabs = T, longtable=T,
              caption=sprintf(
                paste("%d Loans/Pool",
                      "Shapiro-Wilks (S-W),",
                      "Kolmogorov-Smirnov (K-S),",
                      "Anderson-Darling (A-D)",
                      "tests"),
                              sim$loan.count),
        col.name=c("S-W (W)","S-W p-value",
                   "K-S (D)","K-S p-value",
                   "A-D (A)","A-D p-value"
                   ),
        digits=4
        ) %>% kable_paper("striped", full_width = F) # kable_styling()

}# show.norm.stat()
```

```
min.cutoff<-1
max.cutoff<-60 # only include prepay from month 1:max.cutoff
trials<-250
#
lc<-100
cat(sprintf("\n\n## Small Pool Statistics (%d Loans/Pool)\n",lc))
set.seed(666) # for unit replicability
sim<-pool.simulation(trials,min.cutoff,max.cutoff,loan.count=lc)
title<-sprintf("%d Loan Pool Simulation, %d Trials",sim$loan.count,sim$trials)
print.time.stats(time=sim$time,title=title)
show.pool.sim.results(sim=sim)
show.norm.stat(sim,"%s norm-QQ (%d loans)")
```

```
lc<-250
cat(sprintf("\n\n## Medium Pool Statistics (%d Loans/Pool)\n",lc))
set.seed(666) # for unit replicability
sim<-pool.simulation(trials,min.cutoff,max.cutoff,loan.count=lc)
title<-sprintf("%d Loan Pool Simulation, %d Trials",sim$loan.count,sim$trials)
print.time.stats(time=sim$time,title=title)
show.pool.sim.results(sim=sim)
show.norm.stat(sim,"%s norm-QQ (%d loans)")
```

```
lc<-500
cat(sprintf("\n\n## Medium-Large Pool Statistics (%d Loans/Pool)\n",lc))
set.seed(666) # for unit replicability
```

```
sim<-pool.simulation(trials,min.cutoff,max.cutoff,loan.count=lc)
title<-sprintf("%d Loan Pool Simulation, %d Trials",sim$loan.count,sim$trials)
print.time.stats(time=sim$time,title=title)
show.pool.sim.results(sim=sim)
show.norm.stat(sim,"%s norm-QQ (%d loans)")
```

```
lc<-1000
cat(sprintf("\n\n## Large Pool Statistics (%d Loans/Pool)\n",lc))
set.seed(666) # for unit replicability
sim<-pool.simulation(trials,min.cutoff,max.cutoff,loan.count=lc)
title<-sprintf("%d Loan Pool Simulation, %d Trials",sim$loan.count,sim$trials)
print.time.stats(time=sim$time,title=title)
show.pool.sim.results(sim=sim)
show.norm.stat(sim,"%s norm-QQ (%d loans)")
```

Plot/Print Utility Functions

```
# plot.cfs(cf,title) ==========================================================================
#
# plot cashflows
#
plot.cfs<-function(cf,title){
  par(mar=c(bottom=5, left=5, top=3, right=5) + 0.1) # bot, left, top, right in lines
  maxy<-scale.axis.max(max(cf$int+cf$sched.prin+cf$unsched.prin))
  displ.y<-maxy
  suff.y<-"($)"
  if(maxy>1000*1000){
    displ.y<-maxy/1000
    suff.y<-"($1,000s)"
  }
  cfm<-as.matrix(cf[,c("int","sched.prin","unsched.prin")])
  rownames(cfm)<-cf$mon
  cfmt<-t(cfm) # transpose
  barplot(cfmt,border=NA,col=c(int.col,sched.prin.col,unsched.prin.col),space=c(0,0),
          ylim=c(0,maxy),
          cex.main=cex.size,
          xaxt="n",yaxt="n",xlab="",ylab="",main=title)
  axis(1, at = seq(0, orig.term, 12*5),cex.axis=cex.size)
  axis(2, at = seq(0, maxy, maxy/4),
       prettyNum(seq(0, displ.y, displ.y/4),big.mark=",",
                 scientific=FALSE),las=1,cex.axis=cex.size)
  maxy2<-max(cf[!is.na(cf$bbal)]$bbal)

  displ.y2<-maxy2
  suff.y2<-"($)"
  if(maxy2>1000*1000){
    displ.y2<-maxy2/1000
    suff.y2<-"($1,000s)"
  }

  par(new=TRUE)      # new plot on same canvas, for ending balance
  plot(cf$mon,cf$ebal,type="l",col=ebal.col,xlab="", lty=2, ylab="",xaxt="n",yaxt="n",
       axes=FALSE,cex.main=cex.size ## cex.axis=cex.size,cex.lab=cex.size
```

```
  )
  axis(4, at = seq(0, maxy2, maxy2/4),
     prettyNum(seq(0, displ.y2, displ.y2/4),big.mark=",", scientific=FALSE),las=1,
     cex.axis=cex.size)

  mtext("Age (Mo.)",side=1,line=2.0,cex=cex.size)
  mtext(paste("Monthly Cashflow",suff.y),side=2,line=4.0,cex=cex.size)
  mtext(paste("Ending Bal.",suff.y2),side=4,line=4.0,cex=cex.size)
  legend(x="bottom", xpd=TRUE,inset = c(0, -0.75),horiz=TRUE,cex=cex.size,
         legend=c("Interest","Sched.Prin.","Unsched.Prin.","Ending Balance"),
         lty=c(1,1,1,2),
         col=c(NA,NA,NA,ebal.col),
         fill=c(int.col,sched.prin.col,unsched.prin.col,NA),border=NA,
         bty="n"
         )
  cat('\n\n')
} # plot.cfs()
```

```
plot.comp.bals<-function(cf,cf2,title){
  par(mar=c(bottom=5, left=5, top=3, right=5) + 0.1) # bot, left, top, right in lines

  maxy2<-max(cf[!is.na(cf$bbal)]$bbal)

  maxy<-max(cf[!is.na(cf$bbal)]$bbal,cf2[!is.na(cf2$bbal)]$bbal)
  displ.y<-maxy
  suff.y<-"($)"
  if(maxy>1000*1000){
    displ.y<-maxy/1000
    suff.y<-"($1,000s)"
  }

  plot(cf$mon,cf$ebal,type="l",col=ebal.col,main=title,
       xlab="", ylab="",xaxt="n",yaxt="n",axes=FALSE,
       cex.main=cex.size)
  lines(cf2$mon,cf2$ebal,lty=2,col=ebal2.col) # dashed
  axis(1, at = seq(0, orig.term, 12*5),cex.axis=cex.size)
  axis(2, at = seq(0, maxy, maxy/4),
       prettyNum(seq(0, displ.y, displ.y/4),big.mark=",", scientific=FALSE),
       las=1,cex.axis=cex.size)

  mtext("Age (Mo.)",side=1,line=2.0,cex=cex.size)
  mtext(paste("Ending Bal.",suff.y),side=2,line=4.0,cex=cex.size)
  legend(x="bottom", xpd=TRUE,inset = c(0, -0.75),horiz=TRUE,cex=cex.size,
         legend=c("Loan Simulation","MBS Convention"),
         lty=c(1,2),
         col=c(ebal.col,ebal2.col),
         bty="n"
         )
  cat('\n\n')
} # plot.comp.bals()
```

```
plot.hist.cpr<-function(cf,title=NA,baseline.cpr=NA){
  par(mar=c(bottom=5, left=5, top=3, right=5) + 0.1) # bot, left, top, right in lines
  if(is.na(baseline.cpr)){ baseline.cpr=0.10 }
```

```
  maxy<-ceiling(baseline.cpr*10)/5
  ## 0.20 ## scale.axis.max(max(cf$int+cf$sched.prin+cf$unsched.prin))
  hist.cpr<-calc.hist.cpr(cf)
  plot(hist.cpr$mon,hist.cpr$cpr01,col=cpr01.col,type="l",lty=2,xlab="",
       main=title,cex.main=cex.size,
       ylab="",xaxt="n",yaxt="n",ylim=c(0,maxy)) # type="l"
  lines(hist.cpr$cpr03,lty=1,col=cpr03.col) #
  lines(hist.cpr$cpr06,lty=1,col=cpr06.col) #
  lines(hist.cpr$cpr12,lty=1,lwd=2,col=cpr12.col) #
  abline(h=baseline.cpr,lty=2,col=baseline.cpr.col)
  axis(1, at = seq(0, orig.term, 12*5),cex.axis=cex.size)
  axis(2, at = seq(0, maxy, maxy/4),
       sprintf("%3.0f%%",seq(0, maxy, maxy/4)*100),cex.axis=cex.size,las=1)
  par(new=TRUE)      # new plot on same canvas, for ending balance
  plot(hist.cpr$mon,hist.cpr$ebal,type="l",col=ebal.col,lty=2,
       xlab="", xaxt="n",ylab="",yaxt="n",axes=FALSE,
          cex.main=cex.size
  )
  maxy2<-max(cf[!is.na(cf$bbal)]$bbal)
    displ.y2<-maxy2
  suff.y2<-"($)"
  if(maxy2>1000*1000){
    displ.y2<-maxy2/1000
    suff.y2<-"($1,000s)"
  }

  axis(4, at = seq(0, maxy2, maxy2/4),
     prettyNum(seq(0, displ.y2, displ.y2/4),big.mark=",", scientific=FALSE),
     las=1,cex.axis=cex.size)
  mtext("Age (Mo.)",side=1,line=2.0,cex=cex.size)
  mtext(paste("Prepayment (CPR)"),side=2,line=4.0,cex=cex.size)
  mtext(paste("Ending Bal.",suff.y2),side=4,line=4.0,cex=cex.size)
  legend(x="bottom", xpd=TRUE,inset = c(0, -0.75),horiz=TRUE,cex=cex.size*0.9,
         legend=c("1m CPR (LHS)","3m CPR (LHS)", "6m CPR (LHS)", "12m CPR (LHS)",
                  "End.Bal.(RHS)"),
         lty=c(1,1,1,1,2),
         lwd=c(1,1,1,2,1),
         col=c(cpr01.col,cpr03.col,cpr06.col,cpr12.col,ebal.col),
         bty="n"
         )
} # plot.hist.cpr()
```

```
# print.time.stats() ================================================================
# helper function to pretty print proc_time object from system.time() call
print.time.stats<-function(time,title=NA){
  cat("\n\n")
  #knitr::kable(t(time[1:3]),
   kbl(t(time[1:3]),
              caption=title, booktabs=T,longtable=T,
               col.names=c("User Time (sec)","System Time (ec)","Elapsed Time (sec)"),
               format.args=list(width=20)) %>% ## kable_styling()
            kable_paper("striped", full_width = F) # kable_styling()
}# print.time.stats()
```

Colophon

```
print(sessionInfo())
```

```
R version 4.4.3 (2025-02-28)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 20.04.6 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3;  LAPACK version 3.9.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] nortest_1.0-4     kableExtra_1.4.0  data.table_1.17.0 dplyr_1.1.4

loaded via a namespace (and not attached):
 [1] vctrs_0.6.5        svglite_2.2.1      cli_3.6.5          knitr_1.50
 [5] rlang_1.1.6        xfun_0.52          stringi_1.8.7      generics_0.1.3
 [9] textshaping_1.0.1  glue_1.8.0         htmltools_0.5.8.1  tinytex_0.57
[13] scales_1.4.0       rmarkdown_2.29     evaluate_1.0.3     tibble_3.2.1
[17] fastmap_1.2.0      yaml_2.3.10        lifecycle_1.0.4    stringr_1.5.1
[21] compiler_4.4.3     RColorBrewer_1.1-3 pkgconfig_2.0.3    rstudioapi_0.17.1
[25] systemfonts_1.2.3  farver_2.1.2       digest_0.6.37      viridisLite_0.4.2
[29] R6_2.6.1           tidyselect_1.2.1   pillar_1.10.2      magrittr_2.0.3
[33] tools_4.4.3        xml2_1.3.8
```

```
packinfo <- installed.packages(fields = c("Package", "Version"))
print(packinfo[,  "Version", drop=F])
```

```
            Version
base64enc   "0.1-3"
bslib       "0.9.0"
cachem      "1.1.0"
cli         "3.6.5"
commonmark  "1.9.5"
cpp11       "0.5.2"
data.table  "1.17.0"
digest      "0.6.37"
dplyr       "1.1.4"
evaluate    "1.0.3"
fansi       "1.0.6"
```

```
farver       "2.1.2"
fastmap      "1.2.0"
fontawesome  "0.5.3"
formattable  "0.2.1"
fs           "1.6.6"
generics     "0.1.3"
glue         "1.8.0"
highr        "0.11"
htmltools    "0.5.8.1"
htmlwidgets  "1.6.4"
jquerylib    "0.1.4"
jsonlite     "2.0.0"
kableExtra   "1.4.0"
knitr        "1.50"
labeling     "0.4.3"
lifecycle    "1.0.4"
litedown     "0.7"
magrittr     "2.0.3"
markdown     "2.0"
memoise      "2.0.1"
mime         "0.13"
nortest      "1.0-4"
pillar       "1.10.2"
pkgconfig    "2.0.3"
R6           "2.6.1"
rappdirs     "0.3.3"
RColorBrewer "1.1-3"
rlang        "1.1.6"
rmarkdown    "2.29"
rstudioapi   "0.17.1"
sass         "0.4.10"
scales       "1.4.0"
stringi      "1.8.7"
stringr      "1.5.1"
svglite      "2.2.1"
systemfonts  "1.2.3"
textshaping  "1.0.1"
tibble       "3.2.1"
tidyselect   "1.2.1"
tinytex      "0.57"
utf8         "1.2.4"
vctrs        "0.6.5"
viridisLite  "0.4.2"
withr        "3.0.2"
xfun         "0.52"
xml2         "1.3.8"
yaml         "2.3.10"
txtplot      "1.0-4"
base         "4.4.3"
boot         "1.3-31"
class        "7.3-23"
cluster      "2.1.8"
codetools    "0.2-19"
compiler     "4.4.3"
```

```
datasets     "4.4.3"
foreign      "0.8-88"
graphics     "4.4.3"
grDevices    "4.4.3"
grid         "4.4.3"
KernSmooth   "2.23-26"
lattice      "0.22-5"
MASS         "7.3-65"
Matrix       "1.7-2"
methods      "4.4.3"
mgcv         "1.9-1"
nlme         "3.1-167"
nnet         "7.3-20"
parallel     "4.4.3"
rpart        "4.1.24"
spatial      "7.3-11"
splines      "4.4.3"
stats        "4.4.3"
stats4       "4.4.3"
survival     "3.8-3"
tcltk        "4.4.3"
tools        "4.4.3"
utils        "4.4.3"
```