Mortgage Loan & Pool Amortization & Prepay Mechanics

Rei Shinozuka rei@reishinozuka.com

2025-05-11

Amortization and Prepayment Calculations 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*.

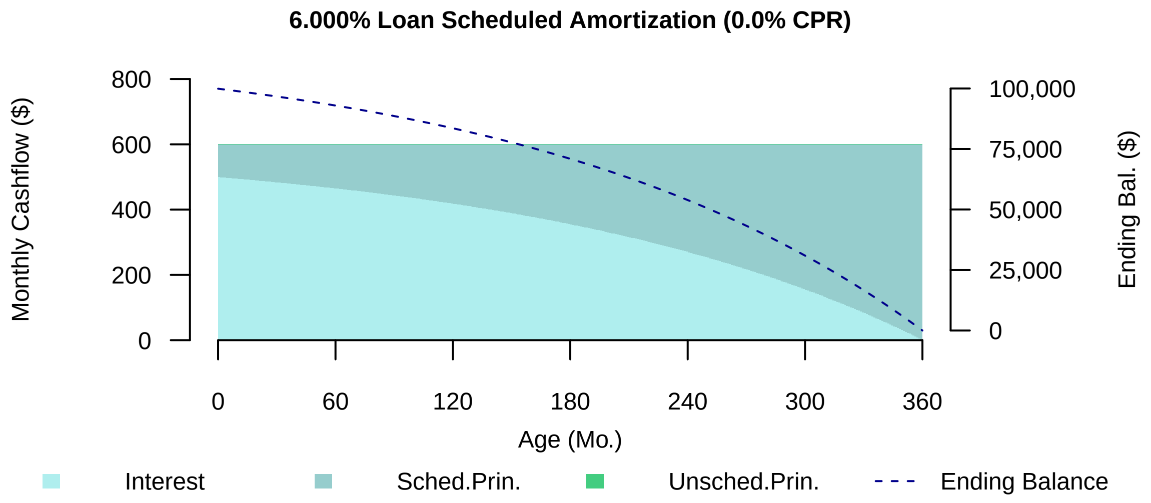
Most of the numerical conventions which follow are based on the reference *Standard Formulas for the Analysis of Mortgage-Backed Securities and Other Related Securities* (1999) published by the **Bond Market Association** (today known as **SIFMA**) available at the URL:

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

and denoted *SF1999* in any references below.

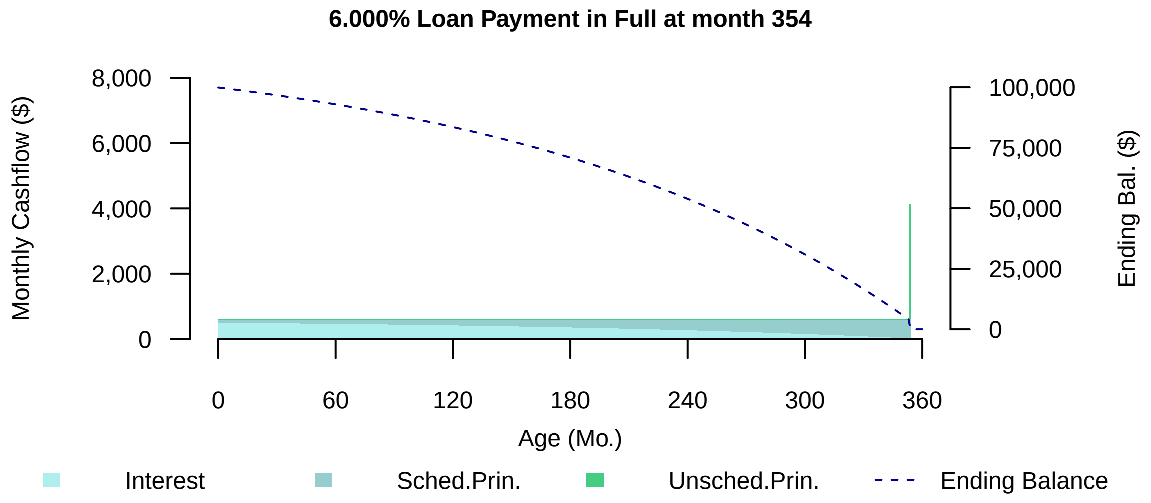
Basic Loan Amortization Mechanics

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.



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*.

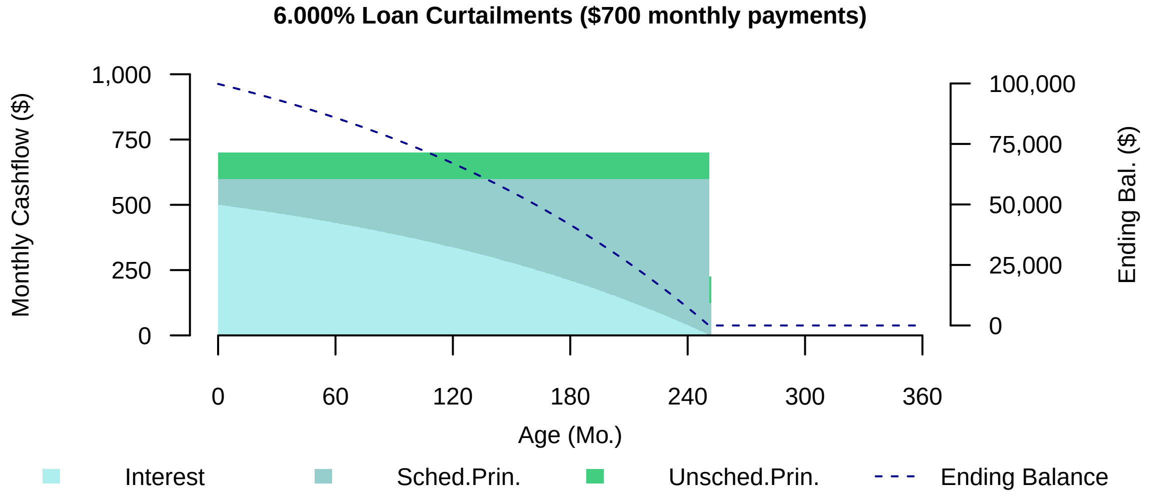
The below example shows a loan paying in full prior to loan maturity.



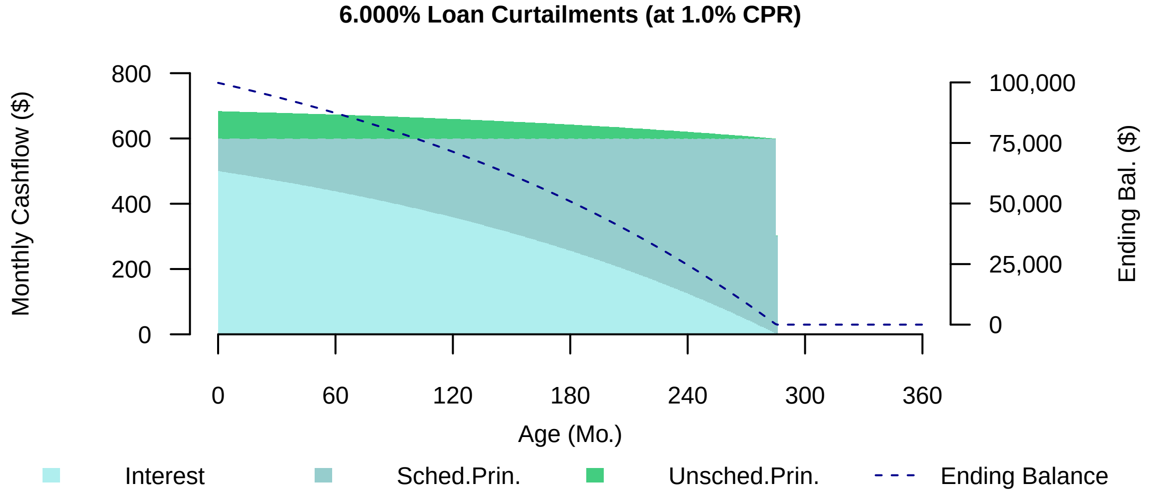
Loan Curtailments The borrower may also partially prepay the remaining balance of the loan resulting in a *curtailment*.
Curtailments are typically small relative to the monthly payment, often produced when borrowers round up the monthly payment to a multiple of $100 to simplify book keeping or to shorten 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 payment $100.45. is the monthly curtailment, which reduces the remaining balance and brings the final payment to an earlier date.



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, the curtailments here decrease with 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.*

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. Single Loan Amortization and Prepayment

B. Pool Amortization and Prepayment under MBS Convention

C. Pool Amortization and Prepayment simulating discrete loans

Parameters

- 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. Single Loan Amortization and Prepayment

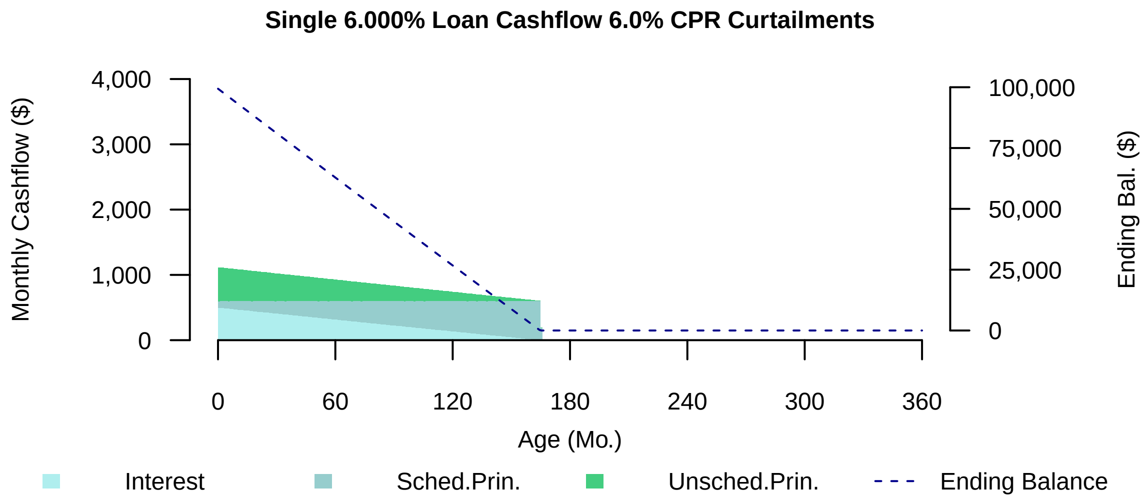
The single loan is modeled following its contractual terms.

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



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:

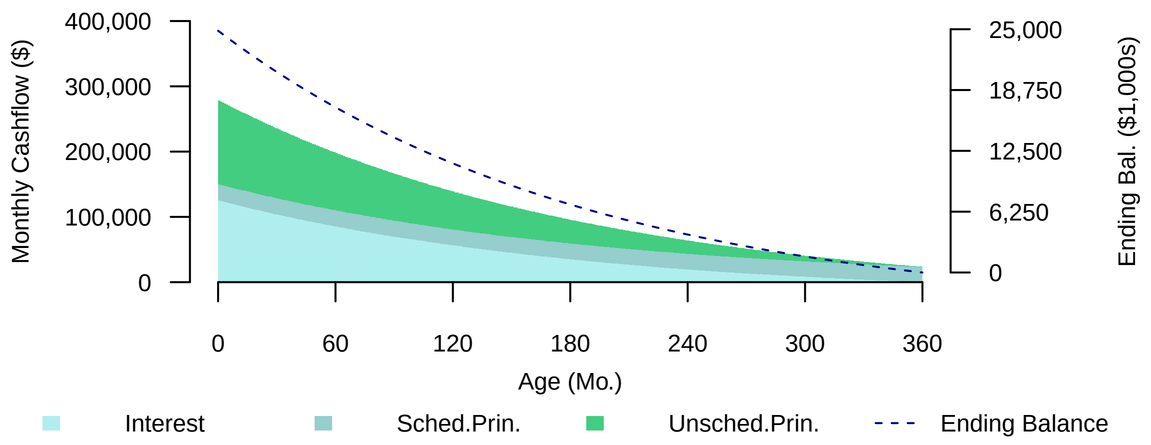
1. paid in full for a proportion of loans up to 100% (CPR > 0%)
2. scheduled amortization (CPR == 0%)
3. There is no provision for curtailment.

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


MBS Convention 6.000% Pool Cashflows, 6.0% CPR

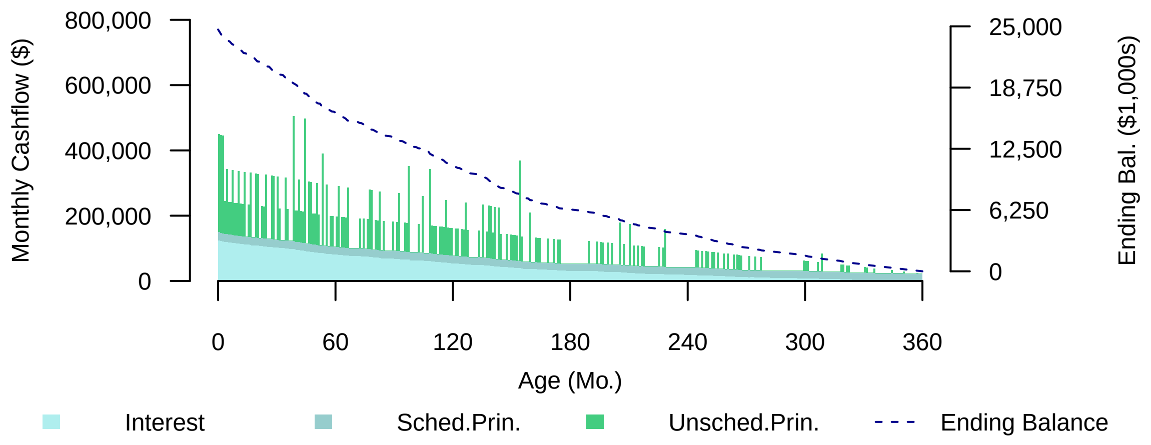
C. Pool Amortization and Prepayment simulating discrete loans

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

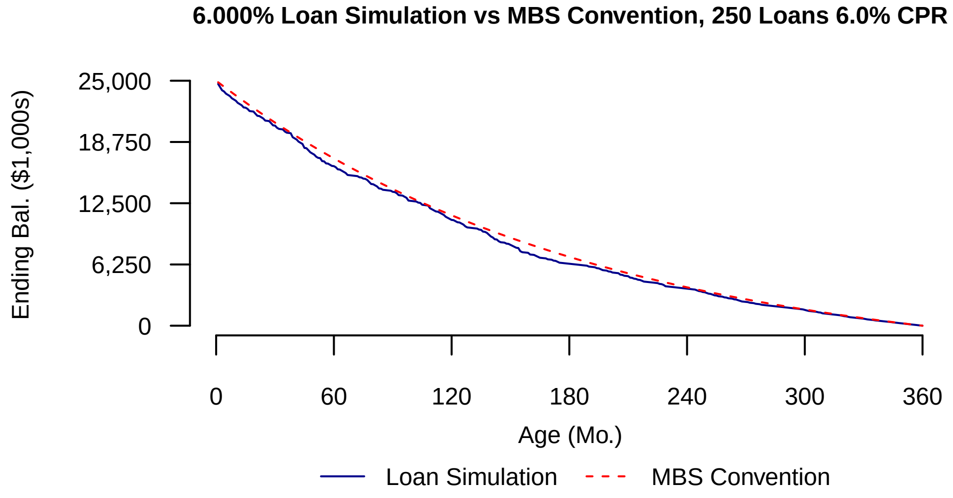
The simulation interprets an SMM as the probability for an individual loan being paid in full each month. There is no provision for curtailment.

Example:

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

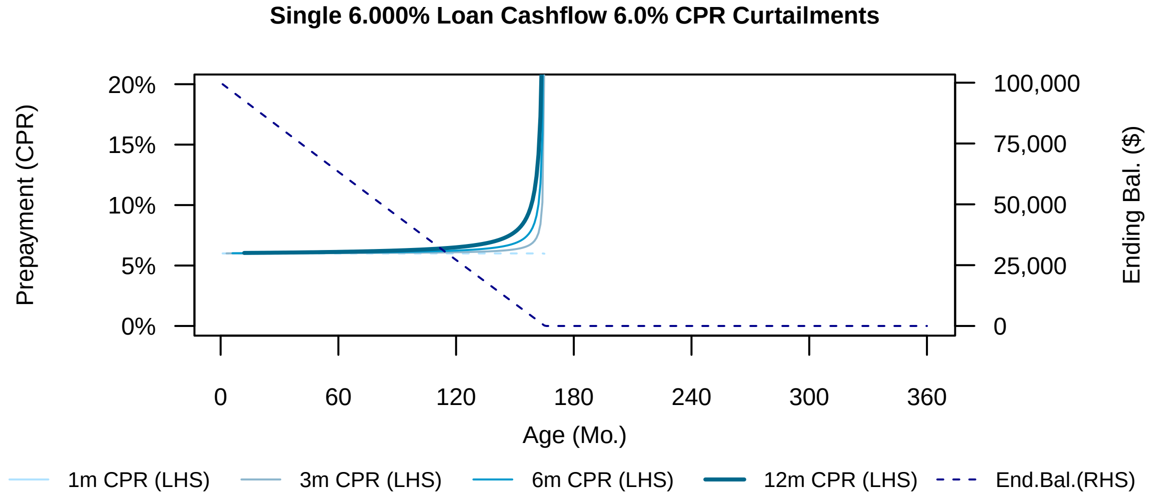

6.000% Pool Loan Simulation Cashflows, 250 Loans, 6.0% CPR

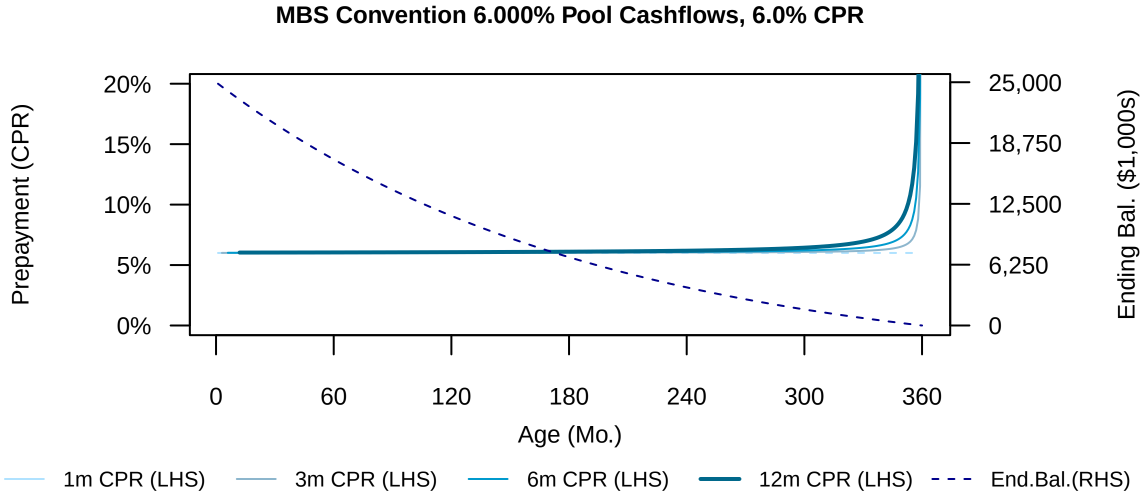
Comparing Simulation against MBS Convention With higher loan counts, the discrete simulation approaches results of the MBS convention.

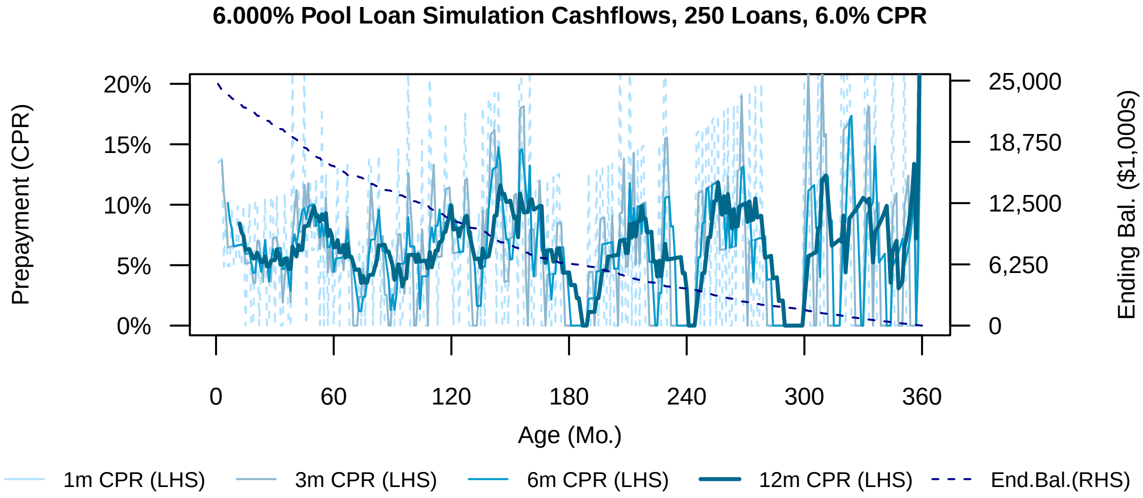


Determining Historical Prepayments from Loan and Pool Cashflows

From loan and pool cashflows, calculate CPRs from balances and unscheduled principal. The pool loan simulation example will most closely resemble measurement of historical prepayments.







Amortization Functions

```
# level.pay.calc()
# returns level (scheduled) payment given term to maturity and note.rate normalized to
# orig balance == 1.0
level.pay.calc<-function(rem.term,monthly.rate)
{
  if(monthly.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 =
#   CPR == 1 signifies paid-in-full
#   0 < CPR < 1 are curtailments
# mo.borrower.pay is used to model when the 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 set, 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 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){
    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()
```

Single Loan Amortization

```
# amort.pool.cf()
# Amortize loan pool using MBS convention
# which models prepayments as continuous
# and pool as 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){
    level.pay<-level.pay.calc(orig.term+1-m,monthly.rate)*mo.bbal  # recalc level pay based on act bal
    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
```

MBS Convention Pool Amortization

```
# simulate.loan.pool(orig.term,note.rate,orig.bal,scalar.cpr,loan.count)
# 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 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 the
# MBS pool convention as loan count exceeds 250 or more.
# This simulation offers is no provision for curtailment.
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)

  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()
```

Loan Simulation Pool Amortization

Parameters

```
loan.count<-250
```

Run Cashflows

```
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)
```

```
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)
```

```
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)
```

```
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)
```

```
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)
```

```
set.seed(666)
cfsim<-simulate.loan.pool(orig.term,note.rate,orig.bal,scalar.cpr,loan.count)
```

```
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)
```

```
# given a cashflow calculate 1-,3-,6- and 12-month CPR
# cashflow cf minimally contains the following 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 balance after first payment
# unsched.prin = unscheduled principal:      cf[1,unsched.prin] borrower payment excess of sched pay
#                                            incorporates payments of full repayments and curtailments
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
  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] w/ 0 prepays
  #       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* page SF-5).
  # when calculating 3-,6- and 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
```

Calculating Historical CPRs

Plot Functions

```
# plot.cfs() 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, -0.65),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"
         )
} # 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, -0.65),horiz=TRUE,cex=cex.size,
         legend=c("Loan Simulation","MBS Convention"),
         lty=c(1,2),
         col=c(ebal.col,ebal2.col),
         bty="n"
         )
} # plot.comp.bals()
```

```
plot.hist.cpr<-function(cf,title=NA){
  par(mar=c(bottom=5, left=5, top=3, right=5) + 0.1) # bot, left, top, right in lines
  maxy<-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) #
  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,xlab="", lty=2,ylab="",xaxt="n",yaxt="n",
          axes=FALSE,
          cex.main=cex.size ## cex.axis=cex.size,cex.lab=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, -0.65),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()
```

Colophon

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

```
R version 4.3.2 (2023-10-31)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 22.04.3 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

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] data.table_1.15.0 dplyr_1.1.4

loaded via a namespace (and not attached):
 [1] digest_0.6.34     utf8_1.2.4        R6_2.5.1          fastmap_1.1.1
 [5] tidyselect_1.2.0  xfun_0.42         magrittr_2.0.3    glue_1.7.0
 [9] tibble_3.2.1      knitr_1.45        pkgconfig_2.0.3   htmltools_0.5.7
[13] rmarkdown_2.25    generics_0.1.3    lifecycle_1.0.4   cli_3.6.2
[17] fansi_1.0.6       vctrs_0.6.5       compiler_4.3.2    highr_0.10
[21] rstudioapi_0.15.0 tools_4.3.2       pillar_1.9.0      evaluate_0.23
[25] yaml_2.3.8        rlang_1.1.3
```

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

```
                Version
acs             "2.1.4"
bitops          "1.0-7"
brew            "1.0-10"
brio            "1.1.4"
bslib           "0.6.1"
cachem          "1.0.8"
checkmate       "2.3.1"
choroplethr     "3.7.2"
choroplethrMaps "1.0.1"
choroplethrZip  "1.5.0"
chron           "2.3-61"
classInt        "0.4-10"
cli             "3.6.2"
```

```
common          "1.1.1"
conflicted      "1.2.0"
credentials     "2.0.1"
curl            "5.2.0"
desc            "1.4.3"
devtools        "2.4.5"
diffobj         "0.3.5"
downlit         "0.4.3"
dplyr           "1.1.4"
e1071           "1.7-14"
fastmap         "1.1.1"
fmtr            "1.6.3"
fontawesome     "0.5.2"
formattable     "0.2.1"
Formula         "1.2-5"
gert            "2.0.1"
ggmap           "4.0.0"
gh              "1.4.0"
gitcreds        "0.1.2"
gridExtra       "2.3"
gsubfn          "0.7"
hexbin          "1.28.3"
Hmisc           "5.1-1"
htmlTable       "2.4.2"
htmltools       "0.5.7"
htmlwidgets     "1.6.4"
httr2           "1.0.0"
ini             "0.3.1"
jpeg            "0.1-10"
jquerylib       "0.1.4"
kableExtra      "1.4.0"
lifecycle       "1.0.4"
mapproj         "1.2.11"
maps            "3.4.2"
memoise         "2.0.1"
miniUI          "0.1.1.1"
patchwork       "1.2.0"
pillar          "1.9.0"
pkgbuild        "1.4.3"
pkgdown         "2.0.7"
pkgload         "1.3.4"
plogr           "0.2.0"
plyr            "1.8.9"
png             "0.1-8"
praise          "1.0.0"
profvis         "0.3.8"
proto           "1.0.0"
proxy           "0.4-27"
rcmdcheck       "1.4.0"
remotes         "2.4.2.1"
reprex          "2.1.0"
RgoogleMaps     "1.5.1"
rlang           "1.1.3"
rmarkdown       "2.25"
```

```
roxygen2        "7.3.1"
rprojroot       "2.0.4"
RSQLite         "2.3.6"
rversions       "2.1.2"
s2              "1.1.6"
sass            "0.4.8"
sessioninfo     "1.2.2"
sf              "1.0-15"
sqldf           "0.4-11"
svglite         "2.1.3"
testthat        "3.2.1"
tibble          "3.2.1"
tidycensus      "1.6"
tidyselect      "1.2.0"
tidyverse       "2.0.0"
tigris          "2.1"
units           "0.8-5"
urlchecker      "1.0.1"
usethis         "2.2.3"
vctrs           "0.6.5"
viridis         "0.6.5"
waldo           "0.5.2"
WDI             "2.7.8"
whisker         "0.4.1"
wk              "0.9.1"
xfun            "0.42"
XML             "3.99-0.16.1"
xopen           "1.0.0"
zip             "2.3.1"
askpass         "1.2.0"
assertthat      "0.2.1"
backports       "1.4.1"
base64enc       "0.1-3"
bit             "4.0.5"
bit64           "4.0.5"
blob            "1.2.4"
broom           "1.0.5"
bslib           "0.6.1"
cachem          "1.0.8"
callr           "3.7.3"
cellranger      "1.1.0"
cli             "3.6.2"
clipr           "0.8.0"
colorspace      "2.1-0"
commonmark      "1.9.1"
cpp11           "0.4.7"
crayon          "1.5.2"
curl            "4.3.2"
data.table      "1.15.0"
DBI             "1.2.2"
dbplyr          "2.4.0"
digest          "0.6.34"
dplyr           "1.1.4"
dtplyr          "1.3.1"
```

```
ellipsis        "0.3.2"
evaluate        "0.23"
fansi           "1.0.6"
farver          "2.1.1"
fastmap         "1.1.1"
fontawesome     "0.5.2"
forcats         "1.0.0"
fs              "1.6.3"
gargle          "1.5.2"
generics        "0.1.3"
ggplot2         "3.4.4"
glue            "1.7.0"
googledrive     "2.1.1"
googlesheets4   "1.1.1"
gtable          "0.3.4"
haven           "2.4.3"
highr           "0.10"
hms             "1.1.3"
htmltools       "0.5.7"
httpuv          "1.6.5"
httr            "1.4.7"
ids             "1.0.1"
isoband         "0.2.7"
jquerylib       "0.1.4"
jsonlite        "1.8.8"
knitr           "1.45"
labeling        "0.4.3"
later           "1.3.2"
lifecycle       "1.0.4"
littler         "0.3.19"
lubridate       "1.9.3"
magrittr        "2.0.3"
memoise         "2.0.1"
mime            "0.12"
modelr          "0.1.11"
munsell         "0.5.0"
openssl         "2.1.1"
pillar          "1.9.0"
pkgconfig       "2.0.3"
pkgKitten       "0.2.3"
prettyunits     "1.2.0"
processx        "3.8.3"
progress        "1.2.3"
promises        "1.2.1"
ps              "1.7.6"
purrr           "1.0.2"
R6              "2.5.1"
ragg            "1.2.5"
rappdirs        "0.3.3"
RColorBrewer    "1.1-3"
Rcpp            "1.0.12"
readr           "2.1.5"
readxl          "1.4.3"
rematch         "2.0.0"
```

```
rematch2        "2.1.2"
rlang           "1.1.3"
rstudioapi      "0.15.0"
rvest           "1.0.4"
sass            "0.4.8"
scales          "1.3.0"
selectr         "0.4-2"
shiny           "1.8.0"
sourcetools     "0.1.7-1"
stringi         "1.8.3"
stringr         "1.5.1"
sys             "3.4.2"
systemfonts     "1.0.5"
textshaping     "0.3.6"
tibble          "3.2.1"
tidyr           "1.3.1"
tidyselect      "1.2.0"
timechange      "0.3.0"
tinytex         "0.49"
tzdb            "0.4.0"
utf8            "1.2.4"
uuid            "1.2-0"
vctrs           "0.6.5"
viridisLite     "0.4.2"
vroom           "1.6.5"
withr           "3.0.0"
xfun            "0.42"
xml2            "1.3.3"
xtable          "1.8-4"
yaml            "2.3.8"
base            "4.3.2"
boot            "1.3-28"
class           "7.3-22"
cluster         "2.1.6"
codetools       "0.2-19"
compiler        "4.3.2"
datasets        "4.3.2"
foreign         "0.8-86"
graphics        "4.3.2"
grDevices       "4.3.2"
grid            "4.3.2"
KernSmooth      "2.23-22"
lattice         "0.22-5"
MASS            "7.3-60.0.1"
Matrix          "1.6-3"
methods         "4.3.2"
mgcv            "1.9-1"
nlme            "3.1-163"
nnet            "7.3-19"
parallel        "4.3.2"
rpart           "4.1.23"
spatial         "7.3-17"
splines         "4.3.2"
stats           "4.3.2"
```

```
stats4          "4.3.2"
survival        "3.5-7"
tcltk           "4.3.2"
tools           "4.3.2"
utils           "4.3.2"
```