Experiences with using R in credit risk

Hong Ooi
Introduction

Not very sexy....

- LGD haircut modelling
- Through-the-cycle calibration
- Stress testing simulation app
- SAS and R
- Closing comments
Mortgage haircut model

• When a mortgage defaults, the bank can take possession of the property and sell it to recoup the loss
• We have some idea of the market value of the property
• Actual sale price tends to be lower on average than the market value (the *haircut*)
• If sale price > exposure at default, we don’t make a loss (excess is passed on to customer); otherwise, we make a loss

Expected loss = \( P(\text{default}) \times EAD \times P(\text{possess}) \times \text{exp.shortfall} \)

Notes:
1. For ANZ, <10% of defaults actually result in possession
2. Meaning of “haircut” depends on context; very different when talking about, say, US mortgages
Valuation

Expected shortfall

Haircut

Exposure at default
Stat modelling

- Modelling part is in finding parameters for the sale price distribution
  - Assumed distributional shape, eg Gaussian
  - Mean haircut relates average sale price to valuation
  - Spread (volatility) of sale price around haircut
- Once model is found, calculating expected shortfall is just (complicated) arithmetic
Volatility

Volatility of haircut (=sale price/valuation) appears to vary systematically:

<table>
<thead>
<tr>
<th>Property type</th>
<th>SD(haircut)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>11.6%</td>
</tr>
<tr>
<td>B</td>
<td>9.3%</td>
</tr>
<tr>
<td>C</td>
<td>31.2%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State/territory</th>
<th>SD(haircut)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>13.3%</td>
</tr>
<tr>
<td>3</td>
<td>7.7%</td>
</tr>
<tr>
<td>4</td>
<td>9.2%</td>
</tr>
<tr>
<td>5</td>
<td>15.6%</td>
</tr>
<tr>
<td>6</td>
<td>18.4%</td>
</tr>
<tr>
<td>7</td>
<td>14.8%</td>
</tr>
</tbody>
</table>

* valued after possession
Volatility modelling

- Use regression to estimate haircut as a function of loan characteristics
- Hierarchy of models, by complexity:
  - Constant volatility
  - Volatility varying by property type
  - Volatility varying by property type and state
- Use log-linear structure for volatility to ensure +ve variance estimates

- Constant volatility model is ordinary linear regression
- Varying-volatility models can be fit by generalised least squares, using gls in the nlme package
  - Simpler and faster to directly maximise the Gaussian likelihood with optim/nlminb (latter will reproduce gls fit)
Expected shortfall under constant volatility

Expected shortfall with volatility varying by property type

Estimated mean sale price fairly stable

Sale price
Volatility: T-regression

- Still assumes Gaussian distribution for volatility
- Data can be more heavy-tailed than the Gaussian, even after deleting outliers
  - Inflates estimates of variance

- Solution: replace Gaussian distribution with $t$ distribution on small df
  - df acts as a shape parameter, controls how much influence outlying observations have
  - Coding this is a straightforward extension of the Gaussian: simply change all *norm functions to *t
    - Can still obtain Gaussian model by setting df=Inf
  - cf Andrew Robinson’s previous MelbURN talk on robust regression and ML fitting
Example impact

Property type = C, state = 7, valuation $250k, EAD $240k

<table>
<thead>
<tr>
<th>Gaussian model</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean formula</td>
<td>Volatility formula</td>
<td>Expected shortfall ($)</td>
</tr>
<tr>
<td>~1</td>
<td>~1</td>
<td>7,610</td>
</tr>
<tr>
<td>~1</td>
<td>~proptype</td>
<td>23,686</td>
</tr>
<tr>
<td>~1</td>
<td>~proptype + state</td>
<td>29,931</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ts model</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean formula</td>
<td>Volatility formula</td>
<td>Expected shortfall ($)</td>
</tr>
<tr>
<td>~1</td>
<td>~1</td>
<td>4,493</td>
</tr>
<tr>
<td>~1</td>
<td>~proptype</td>
<td>10,190</td>
</tr>
<tr>
<td>~1</td>
<td>~proptype + state</td>
<td>5,896</td>
</tr>
</tbody>
</table>
Model fitting function (simplified)

tmod <- function(formula, formula.s, data, df, ...) {
  mf <- mf.s <- match.call(expand.dots = FALSE)
  mf.s[["formula"]]<- mf.s$formula.s
  mf$df <- mf$s$df <- mf$s$df <- mf.s$df <- mf.s$df <- NULL
  mf[[1]] <- mf.[[1]] <- as.name("model.frame")
  mf <- eval(mf, parent.frame())
  mf.s <- eval(mf.s, parent.frame())
  mm <- model.matrix(attr(mf, "terms"), mf)
  mm.s <- model.matrix(attr(mf.s, "terms"), mf.s)
  y <- model.response(mf)
  p <- ncol(mm)
  p.s <- ncol(mm.s)
  t.nll <- function(par, y, Xm, Xs)
  {
    m <- Xm %*% par[1:p]
    logs <- Xs %*% par[-(1:p)]
    -sum(dt((y - m)/exp(logs), df = df, log = TRUE) - logs)
  }
  lmf <- lm.fit(mm, y) # use ordinary least-squares fit as starting point
  par <- c(lmf$coefficients, log(sd(lmf$residuals)), rep(0, p.s - 1))
  names(par) <- c(colnames(mm), colnames(mm.s))
  nlminb(par, t.nll, y = y, Xm = mm, Xs = mm.s, ...)
}

  tmod(salepr/valuation ~ 1, ~ proptype + state, data = haircut, df = Inf)
tmod(salepr/valuation ~ proptype, ~ proptype, data = haircut, df = 5)
Because it downweights outliers, \( t \) distribution is more concentrated in the center.

Expected shortfall with \( t \)-distributed volatility.
Normal model residuals

$t_5$-model residuals
Notes on model behaviour

- Why does using a heavy-tailed error distribution reduce the expected shortfall?
  - With normal distrib, volatility is overestimated
    - Likelihood of low sale price is also inflated
  - $t$ distrib corrects this
  - Extreme tails of the $t$ less important
    - At lower end, sale price cannot go below 0
    - At higher end, sale price $> EAD$ is gravy
  - This is not a monotonic relationship! At low enough thresholds, eventually heavier tail of the $t$ will make itself felt

- In most regression situations, assuming sufficient data, distributional assumptions (ie, normality, homoskedasticity) are not so critical: CLT comes into play
- Here, they are important: changing the distributional assumptions can change expected shortfall by big amounts
In SAS

- SAS has PROC MIXED for modelling variances, but only allows one grouping variable and assumes a normal distribution
- PROC NLIN does general nonlinear optimisation
- Also possible in PROC IML

- None of these are as flexible or powerful as R
- The R modelling function returns an object, which can be used to generate predictions, compute summaries, etc
- SAS 9.2 now has PROC PLM that does something similar, but requires the modelling proc to execute a STORE statement first
  - Only a few procs support this currently
  - If you’re fitting a custom model (like this one), you’re out of luck
**Through-the-cycle calibration**

- For capital purposes, we would like an estimate of default probability that doesn’t depend on the current state of the economy
- This is called a *through-the-cycle* or *long-run* PD
  - Contrast with a *point-in-time* or *spot* PD, which is what most models will give you (data is inherently point-in-time)
- Exactly what long-run means can be the subject of philosophical debate; I’ll define it as a customer’s average risk, given their characteristics, across the different economic conditions that might arise
  - This is *not* a lifetime estimate: eg their age/time on books doesn’t change
  - Which variables are considered to be cyclical can be a tricky decision to make (many behavioural variables eg credit card balance are probably correlated with the economy)
- During bad economic times, the long-run PD will be below the spot, and *vice-versa* during good times
  - You don’t want to have to raise capital during a crisis
TTC approach

• Start with the spot estimate:

\[ PD(x, e) = f(x, e) \]

• \( x \) = individual customer’s characteristics
• \( e \) = economic variables (constant for all customers at any point in time)
• Average over the possible values of \( e \) to get a TTC estimate
**TTC approach**

- This is complicated numerically, can be done in various ways eg Monte Carlo
- Use backcasting for simplicity: take historical values of $e$, substitute into prediction equation, average the results
  - As we are interested in means rather than quantiles, this shouldn’t affect accuracy much (other practical issues will have much more impact)
- R used to handle backcasting, combining multiple spot PDs into one output value
TTC calculation

- Input from spot model is a prediction equation, along with sample of historical economic data

```r
spot_predict <- function(data)
{
  # code copied from SAS; with preprocessing, can be arbitrarily complex
  xb <- with(data, b0 + x1 * b1 + x2 * b2 + ...)
  plogis(xb)
}

ttc_predict <- function(data, ecodata, from = "2000-01-01", to = "2010-12-01")
{
  dates <- seq(as.Date(from), as.Date(to), by = "months")
  evars <- names(ecodata)
  pd <- matrix(nrow(data), length(dates))
  for(i in seq_along(dates))
  {
    data[evars] <- subset(ecodata, date == dates[i], evars)
    pd[, i] <- spot_predict(data)
  }
  apply(pd, 1, mean)
}
```
Binning/cohorting

- Raw TTC estimate is a combination of many spot PDs, each of which is from a logistic regression
  → TTC estimate is a complicated function of customer attributes
- Need to simplify for communication, implementation purposes

- Turn into bins or cohorts based on customer attributes: estimate for each cohort is the average for customers within the cohort
- Take pragmatic approach to defining cohorts
  - Create tiers based on small selection of variables that will split out riskiest customers
  - Within each tier, create contingency table using attributes deemed most interesting/important to the business
  - Number of cohorts limited by need for simplicity/manageability, <1000 desirable
  - Not a data-driven approach, although selection of variables informed by data exploration/analysis
Binning/cohorting

Example from nameless portfolio:

**Raw TTC PD**

**Cohorted TTC PD**
Binning input

```r
varlist <- list(
  low_doc2=list(name="low_doc",
    breaks=c("N", "Y"),
    midp=c("N", "Y"),
    na.val="N"),
  enq=list(name="wrst_nbr_enq",
    breaks=c(-Inf, 0, 5, 15, Inf),
    midp=c(0, 3, 10, 25),
    na.val=0),
  lvr=list(name="new_lvr_basel",
    breaks=c(-Inf, 60, 70, 80, 90, Inf),
    midp=c(50, 60, 70, 80, 95),
    na.val=70),
...
)

# by application of
# expand.grid()
```

---

```
low_doc wrst_nbr_enq new_lvr_basel ... tier1 tier2
1   N    0      50 ... 1 1
2   N    0      60 ... 1 2
3   N    0      70 ... 1 3
4   N    0      80 ... 1 4
5   N    0      95 ... 1 5
6   N    3      50 ... 1 6
7   N    3      60 ... 1 7
8   N    3      70 ... 1 8
9   N    3      80 ... 1 9
10  N    3      95 ... 1 10
```
Binning output

```plaintext
if low_doc = ' ' then low_doc2 = 1;
else if low_doc = 'Y' then low_doc2 = 1;
else low_doc2 = 2;

if wrst_nbr_enq = . then enq = 0;
else if wrst_nbr_enq <= 0 then enq = 0;
else if wrst_nbr_enq <= 5 then enq = 3;
else if wrst_nbr_enq <= 15 then enq = 10;
else enq = 25;

if new_lvr_basel = . then lvr = 70;
else if new_lvr_basel <= 60 then lvr = 50;
else if new_lvr_basel <= 70 then lvr = 60;
else if new_lvr_basel <= 80 then lvr = 70;
else if new_lvr_basel <= 90 then lvr = 80;
else lvr = 95;
...

if lvr = 50 and enq = 0 and low_doc = 'N' then do; tier2 = 1; ttc_pd = ________; end;
else if lvr = 60 and enq = 0 and low_doc = 'N' then do; tier2 = 2; ttc_pd = ________; end;
else if lvr = 70 and enq = 0 and low_doc = 'N' then do; tier2 = 3; ttc_pd = ________; end;
else if lvr = 80 and enq = 0 and low_doc = 'N' then do; tier2 = 4; ttc_pd = ________; end;
else if lvr = 95 and enq = 0 and low_doc = 'N' then do; tier2 = 5; ttc_pd = ________; end;
else if lvr = 50 and enq = 3 and low_doc = 'N' then do; tier2 = 6; ttc_pd = ________; end;
else if lvr = 60 and enq = 3 and low_doc = 'N' then do; tier2 = 7; ttc_pd = ________; end;
...
```
Binning/cohorting

R code to generate SAS code for scoring a dataset:

```r
sas_all <- character()
for(i in seq_along(tierdefs))
{
  tvars <- tierdefs[[i]]
  varnames <- lapply(varlist[tvars], `[[`, "name")
  this_tier <- which(celltable$tier == i)
  sas <- "if"
  for(j in seq_along(tvars))
  {
    sas <- paste(sas, tvars[j], "=", as.numeric(celltable[this_tier, varnames[j]]))
    if(j < length(tvars))
      sas <- paste(sas, "and")
  }
  sas <- paste(sas, sprintf("then do; tier2 = %d; ttc_pd = %s;", celltable$tier2[this_tier], celltable$ttc_pd[this_tier]))
  sas[-1] <- paste("else", sas[-1])
  sas_all <- c(sas_all, sas,
               sprintf("else put 'ERROR: unhandled case, tier = %d n = ' _n_;", i))
}
writeLines(sas_all, sasfile)
```
Stress testing simulation

- Banks run stress tests on their loan portfolios, to see what a downturn would do to their financial health
- Mathematical framework is similar to the “Vasicek model”:
  - Represent the economy by a parameter $X$
  - Each loan has a transition matrix that is shifted based on $X$, determining its risk grade in year $t$ given its grade in year $t-1$
    - Defaults if bottom grade reached
- Take a scenario/simulation-based approach: set $X$ to a stressed value, run $N$ times, take the average
  - Contrast to VaR: “average result for a stressed economy”, as opposed to “stressed result for an average economy”
- Example data: portfolio of 100,000 commercial loans along with current risk grade, split by subportfolio
- Simulation horizon: ~3 years
Application outline

• Front end in Excel (because the business world lives in Excel)
  • Calls SAS to setup datasets
    • Calls R to do the actual computations

• Previous version was an ad-hoc script written entirely in SAS, took ~4 hours to run, often crashed due to lack of disk space
  • Series of DATA steps (disk-bound)
  • Transition matrices represented by unrolled if-then-else statements (25x25 matrix becomes 625 lines of code)
• Reduced to 2 minutes with R, 1 megabyte of code cut to 10k
  • No rocket science involved: simply due to using a better tool
  • Similar times achievable with PROC IML, of which more later
Application outline

- For each subportfolio and year, get the *median result* and store it
- Next year’s simulation uses this year’s median portfolio
- To avoid having to store multiple transited copies of the portfolio, we manipulate random seeds

```r
for(p in 1:nPortfolios) # varies by project
{
  for(y in 1:nYears)  # usually 2-5
  {
    seed <- .GlobalEnv$.Random.seed
    for(i in 1:nIters)  # around 1,000, but could be less
      result[i] = summary(doTransit(portfolio[p, y], T[p, y]))

    med = which(result == median(result))
    portfolio[p, y + 1] = doTransit(portfolio[p, y], T[p, y], seed, med)
  }
}
```
Data structures

• For business reasons, we want to split the simulation by subportfolio
• And also present results for each year separately
  → Naturally have 2-dimensional (matrix) structure for output: \([i, j]\)th entry is the result for the \(i\)th subportfolio, \(j\)th year

• But desired output for each \([i, j]\) might be a bunch of summary statistics, diagnostics, etc
  → Output needs to be a list

• Similarly, we have a separate input transition matrix for each subportfolio and year
  → Input should be a matrix of matrices
Data structures

R allows matrices whose elements are lists:

```r
T <- matrix(list(), nPortfolios, nYears)
for(i in 1:nPortfolios) for(j in 1:nYears)
  T[[i, j]] <- getMatrix(i, j, ...)

M <- matrix(list(), nPortfolios, nYears)
M[[i, j]]$result <- doTransit(i, j, ...)
M[[i, j]]$sumstat <- summary(M[[i, j]]$result)
```
Data structures

• Better than the alternatives:
  • List of lists: \( L[[i]][[j]] \) contains data that would be in \( M[[i, \ j]] \)
    • Lose ability to operate on/extract all values for a given \( i \) or \( j \) via matrix indexing
  • Separate matrices for each statistic of interest (ie, doing it Fortran-style)
    • Many more variables to manage, coding becomes a chore
    • No structure, so loops may involve munging of variable names
  • Multidimensional arrays conflate data and metadata

• But can lead to rather cryptic code:

```r
getComponent <- function(x, component)
{
  x[] <- lapply(x, `[[`, component)
  lapply(apply(x, 1, I), function(xi) do.call("rbind", xi))
}
getComponent(M, "sumstat")
```
PROC IML: a gateway to R

• As of SAS 9.2, you can use IML to execute R code, and transfer datasets to and from R:

PROC IML;
   call ExportDataSetToR('portfol', 'portfol'); /* creates a data frame */
   call ExportMatrixToR("&Rfuncs", "rfuns");
   call ExportMatrixToR("&Rscript", "rscript");
   call ExportMatrixToR("&nIters", "nIters");
   ...
   submit /R;
      source(rfuncs)
      source(rscript)
   endsubmit;
   call ImportDataSetFromR('result', 'result');
QUIT;

• No messing around with import/export via CSV, transport files, etc
• Half the code in an earlier version was for import/export
**IML: a side-rant**

- IML lacks:
  - Logical vectors: everything has to be numeric or character
  - Support for zero-length vectors (you don’t realise how useful they are until they’re gone)
  - Unoriented vectors: everything is either a row or column vector (technically, everything is a matrix)
- So something like \( x = x + y[z < 0] \); fails in three ways

- IML also lacks anything like a dataset/data frame: everything is a matrix
  \( \rightarrow \) It’s easier to transfer a SAS dataset to and from R, than IML

- *Everything is a matrix*: no lists, let alone lists of lists, or matrices of lists
  - Not even multi-way arrays

- Which puts the occasional online grouching about R into perspective
Other SAS/R interfaces

- SAS has a proprietary dataset format (or, many proprietary dataset formats)
  - R’s foreign package includes `read.ssd` and `read.xport` for importing, and `write.foreign(*, package="SAS")` for exporting
  - Package Hmisc has `sas.get`
  - Package sas7bdat has an experimental reader for this format
  - Revolution R can read SAS datasets
  - All have glitches, are not widely available, or not fully functional
    - First 2 also need SAS installed
  - SAS 9.2 and IML make these issues moot
    - You just have to pay for it
    - Caveat: only works with R <= 2.11.1 (2.12 changed the locations of binaries)
      - SAS 9.3 will support R 2.12+
R and SAS rundown

- Advantages of R
  - Free! (base distribution, anyway)
  - Very powerful statistical programming environment: SAS takes 3 languages to do what R does with 1
  - Flexible and extensible
  - Lots of features (if you can find them)
    - User-contributed packages are a blessing and a curse
  - Ability to handle large datasets is improving
- Advantages of SAS
  - Pervasive presence in large firms
    - “Nobody got fired for buying IBM-SAS”
    - Compatibility with existing processes/metadata
    - Long-term support
  - Tremendous data processing/data warehousing capability
  - Lots of features (if you can afford them)
  - Sometimes cleaner than R, especially for data manipulation
R and SAS rundown

Example: get weighted summary statistics by groups

```
proc summary data = indat nway;
  class a b;
  var x y;
  weight w;
  output sum(w)=sumwt mean(x)=xmean mean(y)=ymean var(y)=yvar
     out = outdat;
run;

outdat <- local({
  res <- t(sapply(split(indat, indat[c("a", "b")]), function(subset) {
    c(xmean = weighted.mean(subset$x, subset$w),
      ymean = weighted.mean(subset$y, subset$w),
      yvar = cov.wt(subset["y"], subset$w)$cov)
  })),
  levs <- aggregate(w ~ a + b, data=indat, sum)
  cbind(levs, as.data.frame(res))
})
```

Thank god for plyr
Challenges for deploying R

• Quality assurance, or perception thereof
  • Core is excellent, but much of R’s attraction is in extensibility, ie contributed packages
  • Can I be sure that the package I just downloaded does what it says? Is it doing more than it says?
  • Backward compatibility
    • Telling people to use the latest version is not always helpful
• No single point of contact
  • Who do I yell at if things go wrong?
  • How can I be sure everyone is using the same version?
• Unix roots make package development clunky on Windows
  • Process is more fragile because it assumes Unix conventions
  • Why must I download a set of third-party tools to compile code?
  • Difficult to integrate with Visual C++/Visual Studio
• Interfacing with languages other than Fortran, C, C++ not yet integrated into core
Commercial R: an aside

- Many of these issues are fundamental in nature
- Third parties like Revolution Analytics can address them, without having to dilute R’s focus (I am not a Revo R user)
  - Other R commercialisations existed but seem to have disappeared
  - Anyone remember S-Plus?
- Challenge is not to negate R’s drawcards
  - Low cost: important even for big companies
  - Community and ecosystem
    - Can I use Revo R with that package I got from CRAN?
    - If I use Revo R, can I/should I participate on R-Help, StackOverflow.com?
- Also why SAS, SPSS, etc can include interfaces to R without risking too much
Good problems to have

- Sign of R’s movement into the mainstream
- S-Plus now touts R compatibility, rather than R touting S compatibility
- Nobody cares that SHAZAM, GLIM, XLisp-Stat etc don’t support XML, C# or Java
Other resources

- **SAS and R blog** by Ken Kleinman and Nick Horton
  - Online support for their book, *SAS and R: Data Management, Statistical Analysis, and Graphics*
- **Hadley Wickham’s** site
  - **STAT480** covers Excel, R and SAS (the links to UCLA ATS files are broken, use /stat/ instead of /STAT/)
- **The DO Loop** is the official SAS/IML blog
- **inside-R**, the Revolution Analytics community site
- **R-bloggers**: aggregated blogroll for R
- **AnnMaria De Mars’ blog** on SAS and social science (contains pictures of naked mole rats)