**Hierarchical Compartmental Models**

Similar to a PK/PD model, Jake describes the state-space process of claims developments with a set of ordinary differential equations (ODE), using \(EX=\) Exposure, \(OS=\) Outstanding claims, \(PD=\) Paid claims (these are the compartments):

\[  
\begin{aligned}  
dEX/dt & = -k\_{er} \cdot EX \\  
dOS/dt & = k\_{er} \cdot RLR \cdot EX – k\_p \cdot OS \\  
dPD/dt & = k\_{p} \cdot RRF \cdot OS  
\end{aligned}  
\]

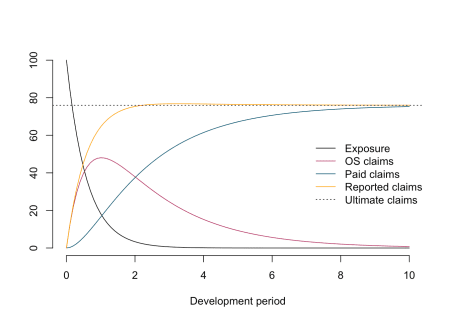
with initial conditions \(EX(0) = \Pi\) (ultimate earned premiums), \(OS(0)=0, PD(0)=0\).

The parameters describe:

* \(k\_{er}\) the rate at which claim events occur and are subsequently reported to the insurer
* \(RLR\) the reported loss ratio
* \(RRF\) the reserve robustness factor, the proportion of outstanding claims that eventually are paid
* \(k\_p\) the rate of payment, i.e. the rate at which outstanding claims are paid

Note, the product of \(RLR\) and \(RRF\) describes the ultimate loss ratio \(ULR =\) ultimate loss / premium, which is a key metric in insurance to assess performance and profitability.

Setting the parameters \(k\_{er}=1.7\), \(RLR=0.8\), \(k\_p=0.5\), \(RRF=0.95\) produces the following output.



The chart illustrates nicely the different compartments (processes) for a policy over time.

The autonomous system of ODEs above can be solved analytically:

\[  
\begin{aligned}  
EX(t) & = \Pi \cdot \exp(- k\_{er} t) \\  
OS(t) & = \frac{\Pi \cdot RLR \cdot k\_{er}}{k\_{er} – k\_p} \cdot  
\left(\exp(-k\_p t) – \exp(-k\_{er} t)\right)\\  
PD (t) & = \frac{\Pi \cdot RLR \cdot RRF}{k\_{er} – k\_p}  
\left( k\_{er} \cdot (1 – \exp(-k\_p t) – k\_p \cdot (1 – \exp(-k\_{er}t ) \right)  
\end{aligned}  
\]

In this post I will focus on the last two equations, as they describe the observable data of outstanding claims and cumulative paid claims over time. However, one could expand the model, e.g. to allow for different premium earning patterns.

**Get example data**

It uses data from the [CAS Reserving Database](http://www.casact.org/research/index.cfm?fa=loss_reserves_data), namely company 337 from the worker’s compensation file.

library(data.table)

CASdata <- fread("http://www.casact.org/research/reserve\_data/wkcomp\_pos.csv")

createLossData2 <- function(CASdata, company\_code){

compData <- CASdata[GRCODE==company\_code,

c("EarnedPremDIR\_D", "AccidentYear", "DevelopmentLag",

"IncurLoss\_D", "CumPaidLoss\_D", "BulkLoss\_D")]

setnames(compData, names(compData),

c("premium", "accident\_year", "dev",

"incurred\_loss", "paid\_loss", "bulk\_loss"))

compData <- compData[, `:=`(

origin = accident\_year - min(accident\_year) + 1)]

compData0 <- compData[dev==1]

compData0 <- compData0[, `:=`(dev = 0, incurred\_loss = 0,

paid\_loss = 0, bulk\_loss = 0)]

compData <- rbindlist(list(compData0, compData))

compData <- compData[, cal := origin + dev - 1][order(origin, dev)]

compData <- compData[, `:=`(

paid\_train = ifelse(cal <= max(origin), paid\_loss, NA),

paid\_test = ifelse(cal > max(origin), paid\_loss, NA),

os\_train = ifelse(cal <= max(origin), incurred\_loss - paid\_loss, NA),

os\_test = ifelse(cal > max(origin), incurred\_loss - paid\_loss, NA))]

traintest <- rbindlist(list(compData[cal <= max(origin)],

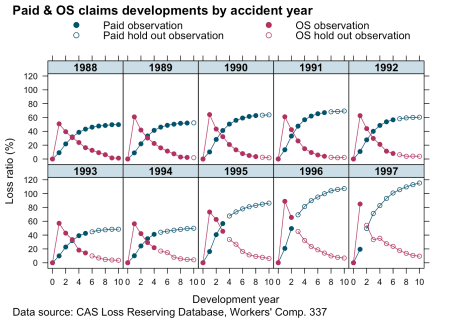
compData[cal > max(origin)]))

return(traintest)

}

lossData <- createLossData2(CASdata, company\_code = 337)

Let’s plot the data of outstanding and paid loss ratio against development years (\(t\)=dev) for the different accident years.



The data looks similar to the example output from our model above. That’s a good start.

**Re-organising data**

In order to fit the two curves simultaneously I use a little trick. I stack the paid and outstanding claims into a single column and add another indicator column (delta), which is either \(0\) for outstanding claims or \(1\) for paid claims. This turns the multivariate data back into a univariate set.

Additionally, I remove the artificial data for dev=0 again, add another column of the indicator variable as a factor (deltaf) and divide all measurement variables by the premium of the oldest accident year. The last step normalises the data and should make modelling a little easier as everything is in the ballpark of 1.

lossData0 <-

rbindlist(

list(lossData[, list(accident\_year, dev, loss\_train=os\_train,

loss\_test=os\_test, delta=0, premium=premium)],

lossData[,list(accident\_year, dev,loss\_train=paid\_train,

loss\_test=paid\_test, delta=1, premium=premium)]

))[order(accident\_year, dev)]

premind <- lossData0[accident\_year==min(accident\_year) & dev==min(dev) & delta==1, premium]

lossData0 <- lossData0[, `:=`(premium = premium/premind,

loss\_train = loss\_train/premind,

loss\_test = loss\_test/premind,

deltaf = factor(delta, labels = c("os", "paid")),

cal=accident\_year + dev - 1)][dev>0]

**Model fitting**

**Non-linear Least Squares**

Before I build a complex Bayesian model I start with a simple non-linear least squares model. Or in other words, I believe the data is generated from a Normal distribution, with the mean described by an analytical function \(\mu(t)=f(t,\dots)\) and constant variance \(\sigma^2\).

\[  
\begin{aligned}  
y(t) & \sim \mathcal{N}(\mu(t), \sigma^2) \\  
\mu(t) & = f(t, \Pi, k\_{er}, k\_p, RLR, RRF, \delta)\\  
& = \Pi \cdot [(1 – \delta) \frac{ RLR \cdot k\_{er}}{k\_{er} – k\_p} \cdot  
\left(\exp(-k\_p t) – \exp(-k\_{er} t) \right) + \\  
& \delta \frac{ RLR \cdot RRF}{k\_{er} – k\_p}  
\left( k\_{er} \cdot (1 – \exp(-k\_p t) – k\_p \cdot (1 – \exp(-k\_{er}t ) \right)]\\  
\delta & = \begin{cases}  
1 \mbox{ if } y \mbox{ is outstanding claim}\\  
0 \mbox{ if } y\mbox{ is paid claim}  
\end{cases}  
\end{aligned}  
\]

To ensure all parameters stay positive I will use the same approach as Jake does in this paper, that is I reparameterize using the logarithm of the parameters.

my.f <- function(t, premium, lk\_er, lk\_p, lRLR, lRRF, delta){

k\_er <- exp(lk\_er)

k\_p <- exp(lk\_p)

RLR <- exp(lRLR)

RRF <- exp(lRRF)

paid <- (RLR \* k\_er / (k\_er - k\_p) \* (exp(-k\_p \* t) - exp(-k\_er \* t)))

os <- (RLR \* RRF / (k\_er - k\_p) \* (k\_er \* (1 - exp(-k\_p \* t)) -

k\_p \* (1 - exp(-k\_er \* t))))

return(premium \* (paid \* (1 - delta) + os \* delta))

}

Using the nls function in R the model can be fitted quickly.

n1 <- nls(loss\_train ~ my.f(dev, premium,

lk\_er=lker, lk\_p=lkp,

lRLR=lRLR, lRRF=lRRF, delta=delta),

data=lossData0[cal<=max(accident\_year)],

start = c(lker = log(1.5), lRLR = log(1),

lkp = log(0.75), lRRF = log(0.75)))

n1

## Nonlinear regression model

## model: loss\_train ~ my.f(dev, premium, lk\_er = lker, lk\_p = lkp, lRLR = lRLR, lRRF = lRRF, delta = delta)

## data: lossData0[cal <= max(accident\_year)]

## lker lRLR lkp lRRF

## 0.8621 -0.1090 -0.8646 -0.4397

## residual sum-of-squares: 0.3505

##

## Number of iterations to convergence: 5

## Achieved convergence tolerance: 2.844e-06

Let’s bring the coefficients back to the original scale:

n1par <- data.table(summary(n1)$coefficients)[, exp(Estimate + 0.5\*`Std. Error`^2)]

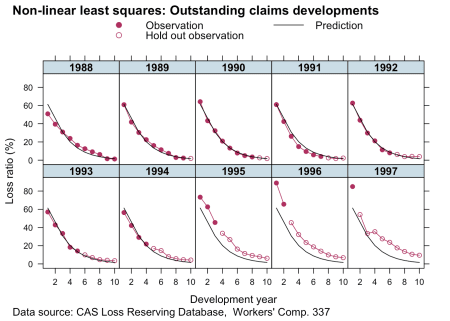
names(n1par) <- c("ker", "RLR", "kp", "RRF")

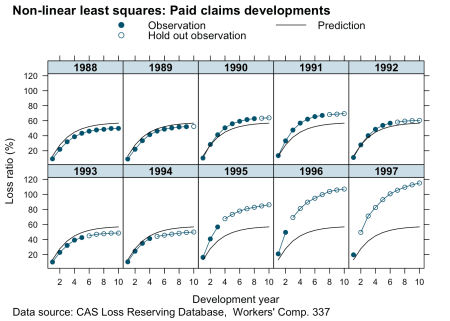
n1par

## ker RLR kp RRF

## 2.4375609 0.8985600 0.4231800 0.6466241

Assuming a constant variance across outstanding (delta=0) and paid claims (delta=1) doesn’t really make much sense, neither that the parameters RLR and RRF are the same across all accident years, which assumes the same ULR for all years: 57.8%.





**Hierarchical non-linear model**

To address the issues mentioned above, I allow for different variances depending on the compartment type (\(\sigma^2\_{y[\delta]}\)) and allow the parameters RLR and RRF to vary across accident years.

\[  
\begin{aligned}  
y(t) & \sim \mathcal{N}(f(t, \Pi, k\_{er}, k\_p, RLR\_{[i]}, RRF\_{[i]}), \sigma\_{y[\delta]}^2) \\  
\begin{pmatrix} RLR\_{[i]} \\ RRF\_{[i]}\end{pmatrix} & \sim  
\mathcal{N} \left(  
\begin{pmatrix}  
\mu\_{RLR} \\  
\mu\_{RRF}  
\end{pmatrix},  
\begin{pmatrix}  
\sigma\_{RLR}^2 & 0 \\  
0 & \sigma\_{RRF}^2  
\end{pmatrix}  
\right)  
\end{aligned}  
\]

The parameters \(\mu\_{RLR}, \mu\_{RRF}\) describe the underlying means across all years \(i\).

With the preparation done, I can fit the hierarchical non-linear model using the nlme package, assuming that the parameters lRLR and lRRF vary across accident years (random effects), while I treat the parameters lker and lkp as fixed effects.

library(nlme)

m1 <- nlme(loss\_train ~ my.f(dev, premium,

lk\_er=lker, lk\_p=lkp,

lRLR=lRLR, lRRF=lRRF, delta=delta),

data=lossData0[cal<=max(accident\_year)],

fixed = lker + lRLR + lkp + lRRF ~ 1,

random = pdDiag(lRLR + lRRF ~ 1),

groups = ~ accident\_year,

weights = varIdent(form = ~ 1 | deltaf),

start = c(lker = log(1.5), lRLR = log(1),

lkp = log(0.75), lRRF = log(0.75)),

control=list(msMaxIter=10000, pnlsMaxIter=10000,

pnlsTol=0.4))

summary(m1)

## Nonlinear mixed-effects model fit by maximum likelihood

## Model: loss\_train ~ my.f(dev, premium, lk\_er = lker, lk\_p = lkp, lRLR = lRLR, lRRF = lRRF, delta = delta)

## Data: lossData0[cal <= max(accident\_year)]

## AIC BIC logLik

## -524.4347 -502.8309 270.2174

##

## Random effects:

## Formula: list(lRLR ~ 1, lRRF ~ 1)

## Level: accident\_year

## Structure: Diagonal

## lRLR lRRF Residual

## StdDev: 0.186949 0.1318405 0.03337559

##

## Variance function:

## Structure: Different standard deviations per stratum

## Formula: ~1 | deltaf

## Parameter estimates:

## os paid

## 1.0000000 0.1805809

## Fixed effects: lker + lRLR + lkp + lRRF ~ 1

## Value Std.Error DF t-value p-value

## lker 0.4102733 0.05624434 97 7.294481 0.0000

## lRLR 0.0225969 0.06751438 97 0.334698 0.7386

## lkp -0.7946096 0.03483365 97 -22.811551 0.0000

## lRRF -0.4049580 0.05558069 97 -7.285948 0.0000

## Correlation:

## lker lRLR lkp

## lRLR -0.375

## lkp -0.928 0.388

## lRRF 0.522 -0.282 -0.572

##

## Standardized Within-Group Residuals:

## Min Q1 Med Q3 Max

## -2.63652139 -0.52065601 0.03990089 0.61348869 2.01628548

##

## Number of Observations: 110

## Number of Groups: 10

The parameters \(k\_{er}\), \(\mu\_{RLR}\), \(k\_p\), \(\mu\_{RRF}\) on the original scale are:

m1\_fe <- data.table(summary(m1)$tTable)[, exp(Value + 0.5\*`Std.Error`^2)]

names(m1\_fe) <- c("ker", "RLR", "kp", "RRF")

m1\_fe

## ker RLR kp RRF

## 1.5096155 1.0251880 0.4520317 0.6680359

If you look at the p-Value of lRLR, we might as well assume \(\mu\_{RLR}=1\).

The estimated ULR across accident years is 68.5%, and the median by accident year:

RLRRRF <- coef(m1)[,c(2,4)]

names(RLRRRF) <- c("RLR", "RRF")

round(exp(cbind(RLRRRF, ULR=apply(RLRRRF,1,sum))),3)

## RLR RRF ULR

## 1988 0.853 0.593 0.506

## 1989 0.925 0.577 0.534

## 1990 0.968 0.673 0.651

## 1991 0.910 0.798 0.727

## 1992 0.946 0.663 0.627

## 1993 0.899 0.562 0.505

## 1994 0.885 0.611 0.540

## 1995 1.232 0.724 0.892

## 1996 1.406 0.785 1.104

## 1997 1.382 0.734 1.014

The parameters \(\sigma\_{y[\delta]}\) are:

(sig\_delta <- sapply(split(resid(m1), lossData0[cal<=max(accident\_year)]$deltaf), sd))

## os paid

## 0.032065014 0.005522208

and \(\sigma\_{RLR}\) and \(\sigma\_{RRF}\) are:

VarCorr(m1)

## accident\_year = pdDiag(list(lRLR ~ 1,lRRF ~ 1))

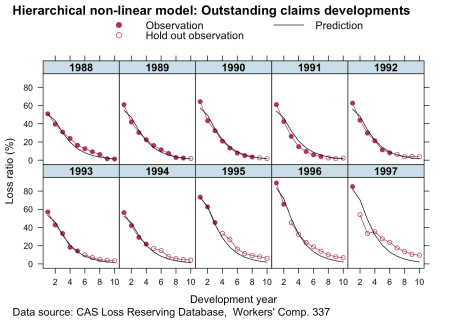
## Variance StdDev

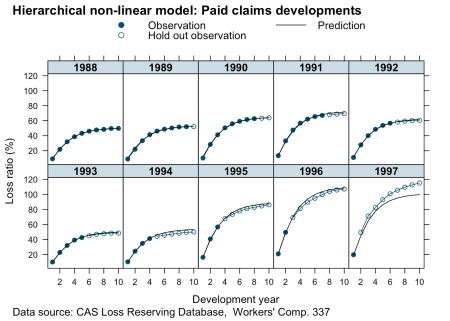
## lRLR 0.03494992 0.18694896

## lRRF 0.01738192 0.13184052

## Residual 0.00111393 0.03337559

Let’s looks at the predictions:





Looks good. This is a huge improvement to the simple non-linear least squares model. For all years but 1997 (here we had only one data point) the predictions look pretty reasonable. The more data we have the more credibility will be given to it and shift parameters RLR and RRF away from the population mean.

**Bayesian reserving models with Stan**

We can fit the same model with Stan, which allows us to make different assumptions on the data generating distribution and specify the priors more flexibly. For example, I’d like to assume Gamma priors over my parameters, this will ensure the parameters are all positive and I don’t have to reparameterise them as I did above.

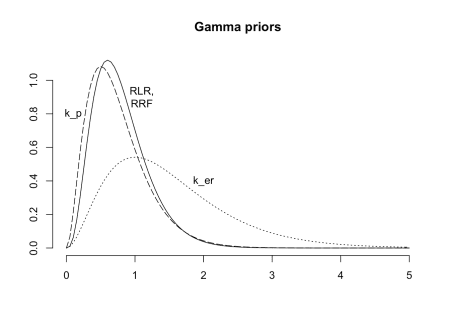
curve(dgamma(x, 4, 5), from = 0, 5, main="Gamma priors",

xlab="", ylab="", bty="n")

curve(dgamma(x, 3, 2), add = TRUE, lty=3)

curve(dgamma(x, 3, 4), add=TRUE, lty=5)

text(x = c(0.1, 1.1, 2), y = c(0.8, 0.9, 0.4), labels = c("k\_p", "RLR,\nRRF", "k\_er"))



Next I specify my model with brm, which will not only generate the Stan code, but also run the model and collect all the results. I continue to use a Gaussian distribution, as it will allow me to compare the output of brm with nlme.

library(rstan)

library(brms)

rstan\_options(auto\_write = TRUE)

options(mc.cores = parallel::detectCores())

fml <-

loss\_train ~ premium \* (

(RLR\*ker/(ker - kp) \* (exp(-kp\*dev) - exp(-ker\*dev))) \* (1 - delta) +

(RLR\*RRF/(ker - kp) \* (ker \*(1 - exp(-kp\*dev)) - kp\*(1 - exp(-ker\*dev)))) \* delta

)

b1 <- brm(bf(fml,

RLR ~ 1 + (1 | accident\_year),

RRF ~ 1 + (1 | accident\_year),

ker ~ 1,

kp ~ 1,

sigma ~ 0 + deltaf,

nl = TRUE),

data = lossData0[cal <= max(accident\_year)],

family = brmsfamily("gaussian", link\_sigma = "log"),

prior = c(prior(gamma(4, 5), nlpar = "RLR", lb=0),

prior(gamma(4, 5), nlpar = "RRF", lb=0),

prior(gamma(3, 2), nlpar = "ker", lb=0),

prior(gamma(3, 4), nlpar = "kp", lb=0)),

control = list(adapt\_delta = 0.999, max\_treedepth=15),

seed = 1234, iter = 1000)

b1

## Family: gaussian

## Links: mu = identity; sigma = log

## Formula: loss\_train ~ premium \* ((RLR \* ker/(ker - kp) \* (exp(-kp \* dev) - exp(-ker \* dev))) \* (1 - delta) + (RLR \* RRF/(ker - kp) \* (ker \* (1 - exp(-kp \* dev)) - kp \* (1 - exp(-ker \* dev)))) \* delta)

## RLR ~ 1 + (1 | accident\_year)

## RRF ~ 1 + (1 | accident\_year)

## ker ~ 1

## kp ~ 1

## sigma ~ 0 + deltaf

## Data: lossData0[cal <= max(accident\_year)] (Number of observations: 110)

## Samples: 4 chains, each with iter = 1000; warmup = 500; thin = 1;

## total post-warmup samples = 2000

## ICs: LOO = NA; WAIC = NA; R2 = NA

##

## Group-Level Effects:

## ~accident\_year (Number of levels: 10)

## Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat

## sd(RLR\_Intercept) 0.25 0.08 0.14 0.47 709 1.00

## sd(RRF\_Intercept) 0.11 0.04 0.06 0.20 838 1.01

##

## Population-Level Effects:

## Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat

## RLR\_Intercept 1.03 0.09 0.85 1.20 602 1.01

## RRF\_Intercept 0.67 0.05 0.58 0.76 701 1.00

## ker\_Intercept 1.55 0.11 1.37 1.77 1111 1.00

## kp\_Intercept 0.45 0.02 0.42 0.49 1181 1.00

## sigma\_deltafos -3.38 0.12 -3.60 -3.12 2000 1.00

## sigma\_deltafpaid -5.07 0.13 -5.31 -4.79 1326 1.00

##

## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample

## is a crude measure of effective sample size, and Rhat is the potential

## scale reduction factor on split chains (at convergence, Rhat = 1).

This looks similar to the output of nlme. Note, the link function for sigma was log, so to compare the output with nlme I have to transform it back to the original scale:

library(rstan)

X <- extract(b1$fit)

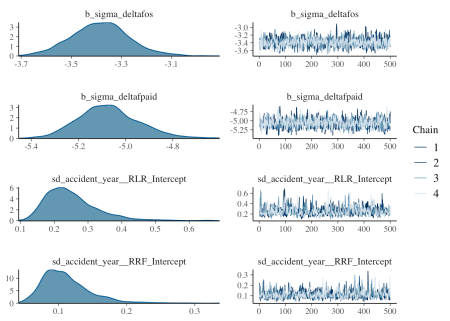
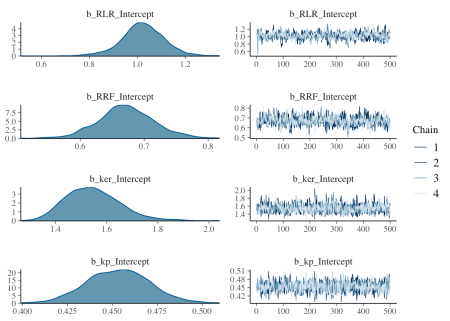
(sig\_delta\_brm <- apply(X$b\_sigma, 2, function(x) exp(mean(x)+0.5\*var(x))))

## [1] 0.034170216 0.006316905

Again, that’s pretty similar.

The simulations are well behaved as well:

plot(b1, N = 4, ask = FALSE)



Next I want to compare the estimated accident year ULRs with the ‘population’ level ULR.

RLR <- sweep(X$r\_1\_RLR\_1, 1, X$b\_RLR, "+")

RRF <- sweep(X$r\_2\_RRF\_1, 1, X$b\_RRF, "+")

ULR <- t(apply(RLR \* RRF, 2, quantile, c(0.025, 0.5, 0.975)))

matplot(unique(lossData0$accident\_year), ULR\*100,

t="l", ylim=c(0, 150), lty=1, col=c(1,2,4), bty="n",

ylab="Projected ULR (%)", xlab="Accident year")

legend("topleft", legend = c("2.5%ile","50%ile", "97.5%ile"),

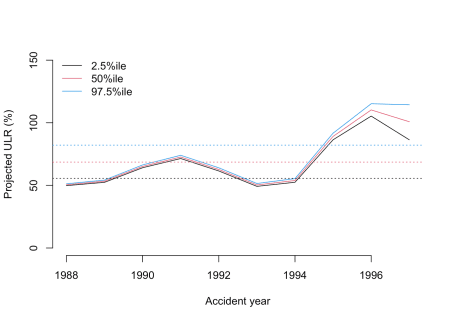
lty=1, col=c(1,2,4), bty="n")

baseULR <- X$b\_RLR \* X$b\_RRF

abline(h=quantile(baseULR, 0.025)\*100, col=1, lty=3)

abline(h=median(baseULR)\*100, col=2, lty=3)

abline(h=quantile(baseULR, 0.975)\*100, col=4, lty=3)



That’s interesting, as the softening casualty market (increasing loss ratio) of the late 1990’s is very visible. From this plot you would assume 1996 was the worst year and perhaps an outlier. However, as the data shows, 1997 was even less profitable than 1996.

Perhaps, the under-prediction for 1997 may arises because historically RRF<1 i.e. over-reserving. However, this switches to under-reserving in more recent accident years as the reserving cycle kicks in (for the latest accident year in particular). With only 2 data points from which to assess this, shrinkage drags the RRF parameter downwards. It would be an interesting exercise to try and incorporate a prior which captures the expectation of a deteriotating reserving cycle.

Let’s plot the predictions of the model, with the plotDevBananas function from an [earlier post](https://magesblog.com/post/correlated-lognormal-chain-ladder-model/).

predClaimsPred <- predict(b1, newdata = lossData0, method="predict")

plotDevBananas(`2.5%ile`/premium\*100 + `97.5%ile`/premium\*100 +

Estimate/premium\*100 + loss\_train/premium\*100 +

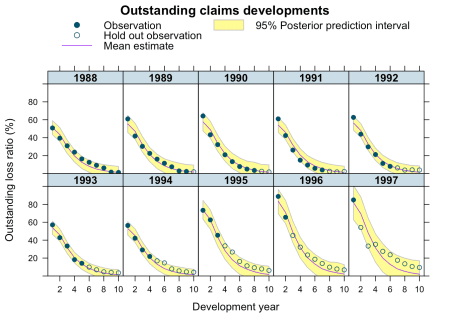
loss\_test/premium\*100 ~

dev | factor(accident\_year), ylim=c(0, 100),

data=cbind(lossData0, predClaimsPred)[delta==0],

main="Outstanding claims developments",

ylab="Outstanding loss ratio (%)")



plotDevBananas(`2.5%ile`/premium\*100 + `97.5%ile`/premium\*100 +

Estimate/premium\*100 + loss\_train/premium\*100 +

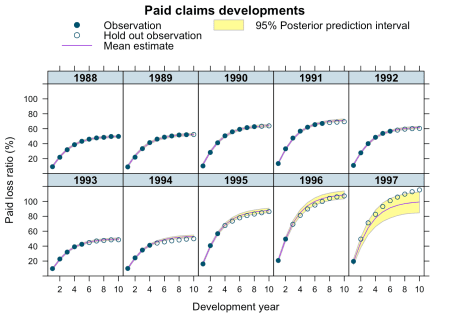
loss\_test/premium\*100 ~

dev | factor(accident\_year), ylim=c(0, 120),

data=cbind(lossData0, predClaimsPred)[delta==1],

main="Paid claims developments",

ylab="Paid loss ratio (%)")



The output looks very promising, apart form the 1997 accident year, which turned out even worse than 1996. Perhaps, I shouldn’t assume a Gaussian data generating distribution, but a distribution which is skewed to the right, such as Gamma, Weibull or Log-normal, all of which is doable with brm by changing the family argument.

Additionally, I could test for auto-correlation, calendar year effects, etc. Jake’s paper has further details on this, he also shows how the model can be extended to cover time-dependent parameters and covariate sub-models.

**Session Info**

sessionInfo()

## R version 3.4.3 (2017-11-30)

## Platform: x86\_64-apple-darwin15.6.0 (64-bit)

## Running under: macOS High Sierra 10.13.2

##

## Matrix products: default

## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib

## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib

##

## locale:

## [1] en\_GB.UTF-8/en\_GB.UTF-8/en\_GB.UTF-8/C/en\_GB.UTF-8/en\_GB.UTF-8

##

## attached base packages:

## [1] methods stats graphics grDevices utils datasets base

##

## other attached packages:

## [1] rstan\_2.17.2 StanHeaders\_2.17.1 ggplot2\_2.2.1

## [4] nlme\_3.1-131 latticeExtra\_0.6-28 RColorBrewer\_1.1-2

## [7] lattice\_0.20-35 data.table\_1.10.4-3 deSolve\_1.20

##

## loaded via a namespace (and not attached):

## [1] Rcpp\_0.12.14 mvtnorm\_1.0-6 gtools\_3.5.0

## [4] zoo\_1.8-1 assertthat\_0.2.0 rprojroot\_1.3-2

## [7] digest\_0.6.13 mime\_0.5 R6\_2.2.2

## [10] plyr\_1.8.4 backports\_1.1.2 stats4\_3.4.3

## [13] evaluate\_0.10.1 coda\_0.19-1 colourpicker\_1.0

## [16] blogdown\_0.4 pillar\_1.0.1 rlang\_0.1.6

## [19] lazyeval\_0.2.1 miniUI\_0.1.1 Matrix\_1.2-12

## [22] DT\_0.2 rmarkdown\_1.8 shinythemes\_1.1.1

## [25] shinyjs\_0.9.1 stringr\_1.2.0 htmlwidgets\_0.9

## [28] loo\_1.1.0 igraph\_1.1.2 munsell\_0.4.3

## [31] shiny\_1.0.5 compiler\_3.4.3 httpuv\_1.3.5

## [34] pkgconfig\_2.0.1 base64enc\_0.1-3 rstantools\_1.4.0

## [37] htmltools\_0.3.6 tibble\_1.4.1 gridExtra\_2.3

## [40] bookdown\_0.5 threejs\_0.3.1 matrixStats\_0.52.2

## [43] dplyr\_0.7.4 brms\_2.0.1 grid\_3.4.3

## [46] xtable\_1.8-2 gtable\_0.2.0 magrittr\_1.5

## [49] scales\_0.5.0 stringi\_1.1.6 reshape2\_1.4.3

## [52] bindrcpp\_0.2 dygraphs\_1.1.1.4 xts\_0.10-1

## [55] tools\_3.4.3 glue\_1.2.0 markdown\_0.8

## [58] shinystan\_2.4.0 crosstalk\_1.0.0 rsconnect\_0.8.5

## [61] abind\_1.4-5 parallel\_3.4.3 yaml\_2.1.16

## [64] inline\_0.3.14 colorspace\_1.3-2 bridgesampling\_0.4-0

## [67] bayesplot\_1.4.0 knitr\_1.18 bindr\_0.1

## [70] Brobdingnag\_1.2-4