This is a follow-up post on [hierarchical compartmental reserving models](https://magesblog.com/post/hierarchical-compartmental-reserving-models/) using PK/PD models. It will show how differential equations can be used with [Stan](https://magesblog.com/post/2018-01-30-pkpd-reserving-models/claimsprocess)/ [brms](https://cran.r-project.org/package=brms) and how correlation for the same group level terms can be modelled.

PK/ PD is usually short for pharmacokinetic/ pharmacodynamic models, but as [Eric Novik](https://www.linkedin.com/in/enovik/) of [Generable](https://www.generable.com/) pointed out to me, it could also be short for *Payment Kinetics/ Payment Dynamics Models* in the insurance context.

In this post I’d like to discuss an extension to Jake Morris’ [*hierarchical compartmental reserving model*](https://magesblog.com/post/hierarchical-compartmental-reserving-models/), as described in his original paper (Morris ([2016](https://magesblog.com/post/2018-01-30-pkpd-reserving-models/#ref-JakeMorris2016))) and my [previous post](https://magesblog.com/post/hierarchical-compartmental-reserving-models/), which allows for a time varying parameter \(k\_{er}(t)\) describing the changing rate of earning and reporting and allows for correlation between \(RLR\), the reported loss ratio, and \(RRF\), the reserve robustness factor. A positive correlation would give evidence of a reserving cycle, i.e. in years with higher initial reported loss ratios a higher proportion of reported losses are paid.

I am very grateful for the support Jake Morris and Paul-Christian Bürkner have given me over the last few weeks, answering questions around the model and [brms](https://cran.r-project.org/package=brms) (Bürkner ([2017](https://magesblog.com/post/2018-01-30-pkpd-reserving-models/#ref-brms))).

**PK/ PD model**

First of all let’s take a look at my new set of differential equations, describing the dynamics of exposure (\(EX\)), outstanding (\(OS\)) and paid claims (\(PD\)) over time. I replaced the constant parameter \(k\_{er}\) with the linear function \(\beta\_{er} \cdot t\).

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

The dynamical system is no longer autonomous and initially I can’t be bothered to solve it analytically. Hence, I use an ODE solver instead, but I will get back to integrating the differential equations later.

**Integrating ODEs with Stan**

Fortunately, an ODE solver is part of the Stan language. The following code demonstrates how I can integrate the differential equations in Stan. You will notice that the model section is empty, as all I want to do for now is run the solver.

functions{

real[] claimsprocess(real t, real [] y, real [] theta,

real [] x\_r, int[] x\_i){

real dydt[3];

dydt[1] = - theta[1] \* t \* y[1];

dydt[2] = theta[1] \* t \* theta[3] \* y[1] - theta[2] \* y[2];

dydt[3] = theta[2] \* theta[4] \* y[2];

return dydt;

}

}

data {

real theta[4];

int N\_t;

real times[N\_t];

real C0[3];

}

parameters{

}

transformed parameters{

real C[N\_t, 3];

C = integrate\_ode\_rk45(claimsprocess, C0, 0, times, theta,

rep\_array(0.0, 0), rep\_array(1, 1));

}

model{

}

generated quantities{

real Exposure[N\_t];

real OS[N\_t];

real Paid[N\_t];

Exposure = C[, 1];

OS = C[, 2];

Paid = C[, 3];

}

To run the Stan code I provide as an input a list with the parameters, the number of data points I’d like to generate, the time line and initial starting values. Note also that I have to set the algorithm = "Fixed\_param" as I run a deterministic model and for that reason one iteration and one chain are sufficient.

library(rstan)

input\_data <- list(

theta=c(Ber=5, kp=0.5, RLR=0.8, RRF=0.9),

N\_t=100,

times=seq(from = 0.1, to = 10, length.out = 100),

C0=c(100, 0, 0))

samples <- sampling(ODEmodel, data=input\_data,

algorithm = "Fixed\_param",

iter = 1, chain = 1)

Let’s take a look at the output.

X <- extract(samples)

out <- data.frame(

Time = input\_data$times,

Exposure = c(X[["Exposure"]]),

Outstandings = c(X[["OS"]]),

Paid = c(X[["Paid"]])

)

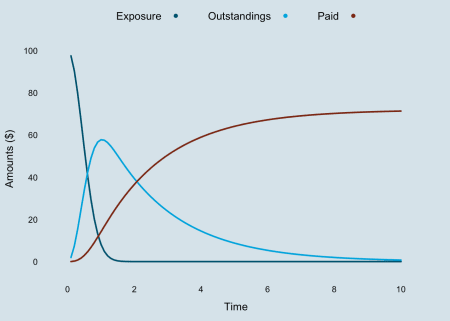
library(latticeExtra)

xyplot(Exposure + Outstandings + Paid ~ Time,

data = out, as.table=TRUE, t="l",

ylab="Amounts ($)", auto.key=list(space="top", columns=3),

par.settings = theEconomist.theme(with.bg = TRUE, box = "transparent"))



The exposure declines faster and the outstanding curve has a more pronounced peak as a result of replacing \(k\_{er}\) with \(\beta\_{er} \cdot t\).

**Updated model**

My updated model is not much different to the one presented in the earlier [post](https://magesblog.com/post/hierarchical-compartmental-reserving-models/), apart from the fact that I allow for the correlation between \(RLR\) and \(RRF\) and the mean function \(\tilde{f}\) is the integral of the ODEs above.

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

**Implementation with brms**

Let’s load the data back into R’s memory:

library(data.table)

lossData0 <- fread("https://raw.githubusercontent.com/mages/diesunddas/master/Data/WorkersComp337.csv")

Jake shows in the appendices of his [paper](https://www.casact.org/pubs/forum/16sforum/Morris.pdf) how to implement this model in R with the [nlmeODE](https://cran.r-project.org/package=nlmeODE) (Tornoe ([2012](https://magesblog.com/post/2018-01-30-pkpd-reserving-models/#ref-nlmeODE))) package, together with more flexible models in OpenBUGS (Lunn et al. ([2000](https://magesblog.com/post/2018-01-30-pkpd-reserving-models/#ref-lunn2000winbugs))). However, I will continue with brms and Stan.

Using the ODEs with brms requires a little extra coding, as I have to provide the integration function as an additional input, just like I did above, where I defined them as user defined functions for Stan.

To model the group-level terms (\(RLR\), \(RRF\)) of the same grouping factor (accident year) as correlated I add a unique identifier to the | operator, here I use p (looks similar to \(\rho\)), i.e. ~ 1 + (1 | p | accident\_year).

Below is my code for the new updated model. I am using again Gamma distributions as my priors for the parameters and [LKJ(2)](http://stla.github.io/stlapblog/posts/StanLKJprior.html) as a prior for the correlation coefficient.

myFuns <- "

real[] ode\_claimsprocess(real t, real [] y, real [] theta,

real [] x\_r, int[] x\_i){

real dydt[3];

dydt[1] = - theta[1] \* t \* y[1]; // Exposure

dydt[2] = theta[1] \* t \* theta[2] \* y[1] - theta[3] \* y[2]; // OS

dydt[3] = theta[3] \* theta[4] \* y[2]; // Paid

return dydt;

}

real claimsprocess(real t, real premium, real Ber, real kp,

real RLR, real RRF, real delta){

real y0[3];

real y[1, 3];

real theta[4];

theta[1] = Ber;

theta[2] = RLR;

theta[3] = kp;

theta[4] = RRF;

y0[1] = premium;

y0[2] = 0;

y0[3] = 0;

y = integrate\_ode\_rk45(ode\_claimsprocess,

y0, 0, rep\_array(t, 1), theta,

rep\_array(0.0, 0), rep\_array(1, 1),

0.001, 0.001, 100); // tolerances, steps

return (y[1, 2] \* (1 - delta) + y[1, 3] \* delta);

}

"

library(brms)

rstan\_options(auto\_write = TRUE)

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

b4 <- brm(

bf(loss\_train ~ claimsprocess(dev, premium, Ber, kp, RLR, RRF, delta),

RLR ~ 1 + (1 | p | accident\_year), # 'p' allow for correlation with RRF

RRF ~ 1 + (1 | p | accident\_year), # 'p' allow for correlation with RLR

Ber ~ 1,

kp ~ 1,

sigma ~ 0 + deltaf, # different sigma for OS and paid

nl = TRUE),

stan\_funs = myFuns, # defintion of 'claimsprocess' function

data = lossData0[cal <= max(accident\_year) & dev > 0],

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(12, 3), nlpar = "Ber", lb=0),

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

set\_prior("lkj(2)", class = "cor")),

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

seed = 1234, iter = 500)

While the original model with the analytical solution ran in about 3 minutes, this code took about 9 hours per chain for 500 samples. Any suggestions how to speed this up will be much appreciated. The frequentist model using nlmeODE runs in seconds.

b4

## Warning: There were 235 divergent transitions after warmup. Increasing adapt\_delta above 0.999 may help.

## See http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Family: gaussian

## Links: mu = identity; sigma = log

## Formula: loss\_train ~ claimsprocess(dev, premium, Ber, kp, RLR, RRF, delta)

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

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

## Ber ~ 1

## kp ~ 1

## sigma ~ 0 + deltaf

## Data: lossData0[cal <= max(accident\_year) & dev > 0] (Number of observations: 110)

## Samples: 4 chains, each with iter = 500; warmup = 250; thin = 1;

## total post-warmup samples = 1000

## 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

## sd(RLR\_Intercept) 0.18 0.06 0.11 0.32

## sd(RRF\_Intercept) 0.15 0.05 0.09 0.27

## cor(RLR\_Intercept,RRF\_Intercept) 0.54 0.24 -0.04 0.88

## Eff.Sample Rhat

## sd(RLR\_Intercept) 197 1.01

## sd(RRF\_Intercept) 397 1.00

## cor(RLR\_Intercept,RRF\_Intercept) 326 1.01

##

## Population-Level Effects:

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

## RLR\_Intercept 0.86 0.06 0.74 0.99 161 1.01

## RRF\_Intercept 0.83 0.05 0.73 0.93 179 1.02

## Ber\_Intercept 5.66 0.32 4.97 6.21 359 1.01

## kp\_Intercept 0.40 0.01 0.38 0.41 446 1.00

## sigma\_deltafos -3.61 0.11 -3.81 -3.40 593 1.00

## sigma\_deltafpaid -4.99 0.11 -5.20 -4.79 587 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).

The output above looks very similar to the output of nlmeODE. That’s good, also that Rhat is close to 1 for all parameters. However, there is a warning message of 235 divergent transitions after warm-up, which I will ignore for the time being.

The correlation coefficient \(\rho\) between \(RLR\_{[i]}\) and \(RRF\_{[i]}\) is estimated as 0.54, but with a wide 95% credible interval from -0.04 to 0.88. Therefore there is moderate evidence of a correlation.

As Jake puts it, a positive correlation between the reported loss ratio and reserve robustness factor parameters by accident year is indicative of a case reserving cycle effect, i.e. more conservative case reserves (low \(RRF\_{[i]}\)) in a hard market (low \(RLR\_{[i]}\)) to create cushions for the future.

stanplot(b4,

pars= c("b\_RLR\_Intercept", "b\_RRF\_Intercept", "b\_Ber\_Intercept",

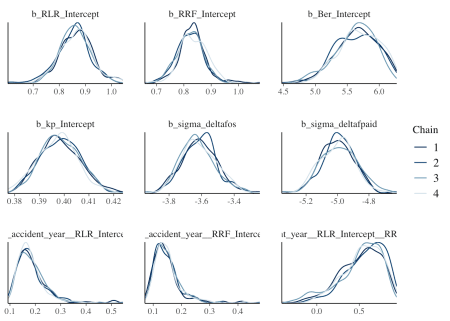
"b\_kp\_Intercept", "b\_sigma\_deltafos", "b\_sigma\_deltafpaid",

"sd\_accident\_year\_\_RLR\_Intercept",

"sd\_accident\_year\_\_RRF\_Intercept",

"cor\_accident\_year\_\_RLR\_Intercept\_\_RRF\_Intercept"),

type="dens\_overlay")



The density plots look OK, all chains seem to have behaved similarly. The last chart in the bottom right shows the distribution of the correlation parameter \(\rho\). Moderate evidence of a correlation might be an understatement. Note also that \(\sigma\_{\delta[OS]}\) and \(\sigma\_{\delta[PD]}\) are on a log-scale.

**Predict future claims developments**

To validate my model against the test data set I can use the predict function in brms. Before I can apply the method I have to expose the Stan functions I wrote above, namely claimsprocess to R.

expose\_functions(b4, vectorize = TRUE) # requires brms >= 2.1.0

With the claimsprocess function now in the R’s memory I can predict and plot the model in the same way as I did in the [previous post](https://magesblog.com/post/hierarchical-compartmental-reserving-models/).

predClaimsPred <- predict(b4, newdata = lossData0, method="predict")

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

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

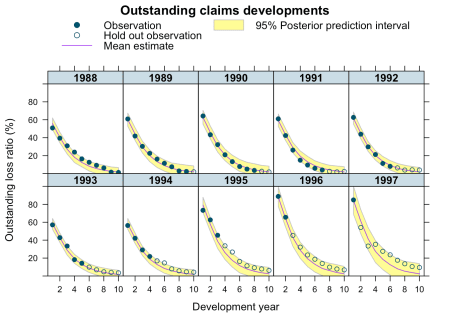
loss\_test/premium\*100 ~

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 +

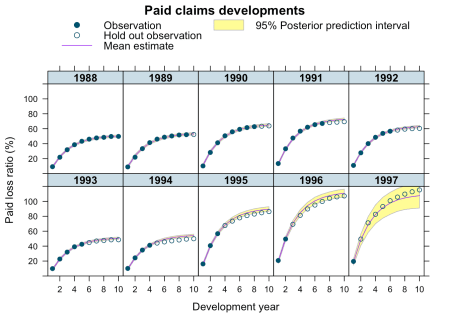
loss\_test/premium\*100 ~

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

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

main="Paid claims developments",

ylab="Paid loss ratio (%)")



The graphs look very similar to the output from the [previous model](https://magesblog.com/post/hierarchical-compartmental-reserving-models/). Yet, the prediction for 1997 improved significantly, perhaps because of the correlation between RLR and RRF no longer being assumed to be zero. But this model has tested my patience, as I used a remote 2-core machine on [Digital Ocean](https://m.do.co/c/3a02238dca6d) and waited 18 hours for the results.

**Analytical solution**

The long run time did in the end motivate me to look for an analytical solution of the ODEs, with initial values \(\mbox{EX}(0) = \Pi\) (premium), \(\mbox{OS}(0) = 0\), \(\mbox{PD}(0) = 0\).

According to [Wolfram Alpha the analytical solution](https://www.wolframalpha.com/input/?i=solve+%7Bf%27(x)+%3D+-x++b++f(x),+g%27(x)+%3D+b++x+++L++f(x)+-+k++g(x),+h%27(x)+%3D+k++R++g(x),+f(0)%3DP,+g(0)%3D0,+h(0)%3D0%7D) is:

\[  
\begin{aligned}  
\mbox{EX}(t) & = \Pi \exp \left(-\frac{\beta\_{er} t^2}{2} \right) \\  
\mbox{OS}(t) & = – \frac{\Pi \cdot RLR}{2 \sqrt{\beta\_{er}}}  
\exp\left(-\frac{\beta\_{er} t^2}{2} – k\_p t \right)  
\left[ \sqrt{2 \pi} k\_p \mbox{erf}\left(\frac{k\_p}{\sqrt{2 \beta\_{er}}}\right)  
\left(-\exp\left(\frac{k\_p^2}{2 \beta\_{er}} + \frac{\beta\_{er} t^2}{2} \right) \right) \right. – \\  
& \left. \qquad \sqrt{2 \pi} k\_p \exp\left(\frac{k\_p^2}{2 \beta\_{er}} + \frac{\beta\_{er} t^2}{2}\right)  
\mbox{erf}\left(\frac{\beta\_{er} t – k\_p}{\sqrt{2 \beta\_{er}}}\right) + 2 \sqrt{\beta\_{er}}  
\exp\left(k\_p t\right) – 2 \sqrt{\beta\_{er}}  
\exp\left(\frac{\beta\_{er} t^2}{2}\right)  
\right] \\  
\mbox{PD}(t) & = \frac{\Pi \cdot RLR \cdot RRF}{2 \sqrt{\beta\_{er}}}  
\exp\left(-k\_p t\right) \left[-\sqrt{2 \pi} k\_p  
\exp\left(\frac{k\_p^2}{2 \beta\_{er}}\right)  
\mbox{erf}\left(\frac{\beta\_{er} t – k\_p}{\sqrt{2 \beta\_{er}}}\right) \right. + \\  
& \left. \qquad \sqrt{2 \pi} k\_p  
\left(-\exp\left(\frac{k^2}{2 \beta\_{er}}\right)\right)  
\mbox{erf}\left(\frac{k}{\sqrt{2 \beta\_{er}}} \right) +  
2 \sqrt{\beta\_{er}} \exp\left(k\_p t\right) –  
2 \sqrt{\beta\_{er}}  
\right]  
\end{aligned}  
\]

Now I can write down \(\tilde{f}(t)\) in a closed-form: \[  
\tilde{f}(t, \delta,\dots) = (1 – \delta) \cdot \mbox{OS}(t, \dots) + \delta \cdot \mbox{PD}(t, \dots)  
\]

The [error function](https://en.wikipedia.org/wiki/Error_function) is part of the Stan language (not directly in R, although it is just erf <- function(x) 2 \* pnorm(x \* sqrt(2)) - 1). Thus, I create a Stan function for the analytical claims process again and feed this into brm.

library(brms)

library(rstan)

rstan\_options(auto\_write = TRUE)

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

myFun <- "

real anaclaimsprocess(real t, real premium, real Ber, real kp,

real RLR, real RRF, real delta){

real os;

real paid;

os = -(RLR \* exp(-(Ber \* t^2)/2.0 - kp \* t) \*

(sqrt(2.0 \* pi()) \* kp \* erf(kp/(sqrt(2.0) \* sqrt(Ber))) \*

(-exp(kp^2/(2.0 \* Ber) + (Ber \* t^2)/2.0)) -

sqrt(2.0 \* pi()) \* kp \* exp(kp^2/(2.0 \* Ber) + (Ber \*t^2)/2.0) \*

erf((Ber\*t - kp)/(sqrt(2.0) \* sqrt(Ber))) +

2.0 \* sqrt(Ber) \* exp(kp \* t) - 2.0 \* sqrt(Ber) \*

exp((Ber\*t^2)/2.0)))/(2.0 \* sqrt(Ber));

paid = (RLR \* RRF \* exp(-kp \* t) \*

(-sqrt(2.0 \* pi()) \* kp \* exp(kp^2/(2.0 \* Ber)) \*

erf((Ber\*t - kp)/(sqrt(2.0) \* sqrt(Ber))) +

sqrt(2.0 \* pi()) \* kp \* (-exp(kp^2/(2.0 \* Ber))) \*

erf(kp/(sqrt(2.0) \* sqrt(Ber))) + 2.0 \* sqrt(Ber) \* exp(kp \* t) -

2.0 \* sqrt(Ber)))/(2.0 \* sqrt(Ber));

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

}

"

b5 <- brm(bf(loss\_train ~ anaclaimsprocess(dev, premium, Ber, kp,

RLR, RRF, delta),

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

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

Ber ~ 1,

kp ~ 1,

sigma ~ 0 + deltaf,

nl = TRUE),

stan\_funs = myFun, # defintion of 'anaclaimsprocess' function

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(12, 3), nlpar = "Ber", lb=0),

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

set\_prior("lkj(2)", class = "cor")),

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

seed = 1234, iter = 1000)

Wonderful, the code ran in about 5 minutes, with twice the number of samples and without any warnings messages. Let’s take a look at the output:

b5

## Family: gaussian

## Links: mu = identity; sigma = log

## Formula: loss\_train ~ anaclaimsprocess(dev, premium, Ber, kp, RLR, RRF, delta)

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

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

## Ber ~ 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

## sd(RLR\_Intercept) 0.18 0.06 0.11 0.32

## sd(RRF\_Intercept) 0.15 0.04 0.09 0.26

## cor(RLR\_Intercept,RRF\_Intercept) 0.52 0.26 -0.08 0.89

## Eff.Sample Rhat

## sd(RLR\_Intercept) 852 1.01

## sd(RRF\_Intercept) 860 1.00

## cor(RLR\_Intercept,RRF\_Intercept) 885 1.00

##

## Population-Level Effects:

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

## RLR\_Intercept 0.86 0.06 0.73 0.97 480 1.00

## RRF\_Intercept 0.83 0.05 0.72 0.93 723 1.00

## Ber\_Intercept 5.69 0.35 5.03 6.42 1968 1.00

## kp\_Intercept 0.40 0.01 0.38 0.41 1701 1.00

## sigma\_deltafos -3.61 0.11 -3.80 -3.40 1337 1.00

## sigma\_deltafpaid -4.99 0.11 -5.19 -4.75 1547 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).

Perfect, the estimates are very much the same as from the ODE model and the plots haven’t changed much either.

stanplot(b5,

pars= c("b\_RLR\_Intercept", "b\_RRF\_Intercept", "b\_Ber\_Intercept",

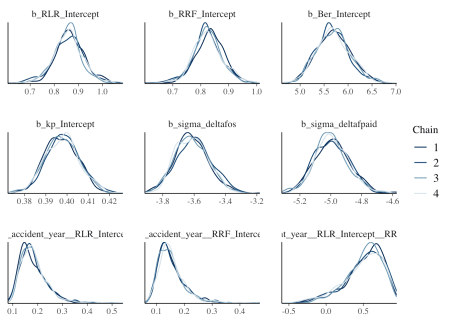
"b\_kp\_Intercept", "b\_sigma\_deltafos", "b\_sigma\_deltafpaid",

"sd\_accident\_year\_\_RLR\_Intercept",

"sd\_accident\_year\_\_RRF\_Intercept",

"cor\_accident\_year\_\_RLR\_Intercept\_\_RRF\_Intercept"),

type="dens\_overlay")



The density plots look smoother and more consistent across chains with twice the number of samples compared to the previous model.

expose\_functions(b5, vectorize = TRUE) # requires brms >= 2.1.0

predClaimsPred2 <- predict(b5, newdata = lossData0, method="predict")

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

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

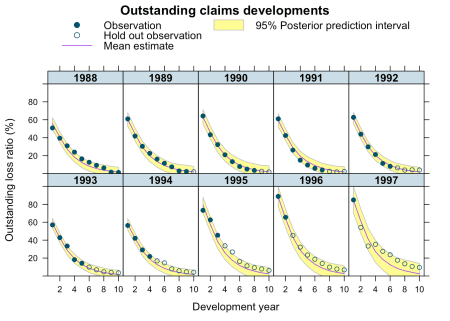
loss\_test/premium\*100 ~

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

data=cbind(lossData0, predClaimsPred2)[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 +

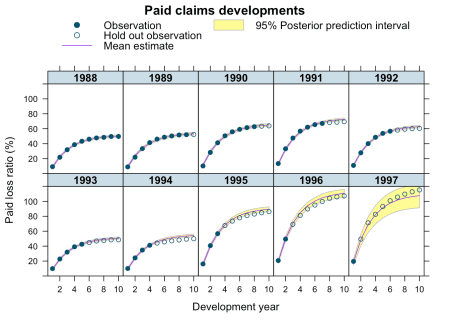
loss\_test/premium\*100 ~

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

data=cbind(lossData0, predClaimsPred2)[delta==1],

main="Paid claims developments",

ylab="Paid loss ratio (%)")



I am much happier now. The model runs in an acceptable time, allowing me to play around with my assumptions further. I have yet to understand why the integration routine in Stan took so long.

**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] brms\_2.1.0 Rcpp\_0.12.15 data.table\_1.10.4-3

## [4] latticeExtra\_0.6-28 RColorBrewer\_1.1-2 lattice\_0.20-35

## [7] rstan\_2.17.3 StanHeaders\_2.17.2 ggplot2\_2.2.1

##

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

## [1] mvtnorm\_1.0-7 gtools\_3.5.0 zoo\_1.8-1

## [4] assertthat\_0.2.0 rprojroot\_1.3-2 digest\_0.6.15

## [7] mime\_0.5 R6\_2.2.2 plyr\_1.8.4

## [10] backports\_1.1.2 stats4\_3.4.3 evaluate\_0.10.1

## [13] coda\_0.19-1 colourpicker\_1.0 blogdown\_0.5

## [16] pillar\_1.1.0 rlang\_0.1.6 lazyeval\_0.2.1

## [19] curl\_3.1 miniUI\_0.1.1 DT\_0.3.3

## [22] Matrix\_1.2-12 rmarkdown\_1.8 labeling\_0.3

## [25] shinythemes\_1.1.1 shinyjs\_1.0 stringr\_1.2.0

## [28] htmlwidgets\_1.0 loo\_1.1.0 igraph\_1.1.2

## [31] munsell\_0.4.3 shiny\_1.0.5 compiler\_3.4.3

## [34] httpuv\_1.3.5 xfun\_0.1 pkgconfig\_2.0.1

## [37] base64enc\_0.1-3 rstantools\_1.4.0 htmltools\_0.3.6

## [40] tibble\_1.4.2 gridExtra\_2.3 bookdown\_0.6

## [43] codetools\_0.2-15 threejs\_0.3.1 matrixStats\_0.53.0

## [46] dplyr\_0.7.4 grid\_3.4.3 nlme\_3.1-131

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

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

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

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

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

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

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

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

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

**References**

Bürkner, Paul-Christian. 2017. “brms: An R Package for Bayesian Multilevel Models Using Stan.” *Journal of Statistical Software* 80 (1): 1–28. doi:[10.18637/jss.v080.i01](https://doi.org/10.18637/jss.v080.i01).

Lunn, David J, Andrew Thomas, Nicky Best, and David Spiegelhalter. 2000. “WinBUGS-a Bayesian Modelling Framework: Concepts, Structure, and Extensibility.” *Statistics and Computing* 10 (4). <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.471.604&rep=rep1&type=pdf>; Springer: 325–37.

Morris, Jake. 2016. “Hierarchical Compartmental Models for Loss Reserving.” In. Casualty Actuarial Society Summer E-Forum; <https://www.casact.org/pubs/forum/16sforum/Morris.pdf>.

Tornoe, Christoffer W. 2012. *NlmeODE: Non-Linear Mixed-Effects Modelling in Nlme Using Differential Equations*. [https://CRAN.R-project.org/package=nlmeODE](https://cran.r-project.org/package=nlmeODE).