Dynamic generalized linear model derivations using Polya-gamma data augmentation

1 Logistic regression

Consider the standard logistic regression model.

$$
\begin{aligned}
y_i &\sim \text{binomial}(n_i, \pi_i) \\
\pi_i &= \frac{\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}
\end{aligned}
$$

To sample the joint posterior distribution of $\boldsymbol{\beta}$, we place multivariate normal priors on $\boldsymbol{\beta}$ and implement the Polya-gamma data augmentation strategy described by (Polson, Scott and Windle, 2013), which allows for Gibbs draws. The details are provided below.

1.1 Derivations

The full conditional posterior distribution of the regression coefficients is proportional to the following:

$$
\begin{aligned}
p(\boldsymbol{\beta}|\boldsymbol{y}) &\propto p(\boldsymbol{\beta}) \prod_{i=1}^{n} p(y_i|\boldsymbol{\beta}) \\
&\propto p(\boldsymbol{\beta}) \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{n_i-y_i}, \qquad \pi_i = \frac{\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}
\end{aligned} \tag{1}
$$

This can be rewritten as the following:

$$
\begin{aligned}
p(\boldsymbol{\beta}|\boldsymbol{y}) &\propto p(\boldsymbol{\beta}) \prod_{i=1}^{n} \left(\frac{\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}\right)^{y_i} \left(1-\frac{\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}\right)^{n_i-y_i} \\
&= p(\boldsymbol{\beta}) \prod_{i=1}^{n} \frac{\exp(\boldsymbol{x}_i'\boldsymbol{\beta})^{y_i}}{(1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta}))^{n_i}}
\end{aligned} \tag{2}
$$

Theorem one of (Polson et al., 2013) states that for $b > 0$,

$$
\frac{(e^{\psi})^a}{(1+e^{\psi})^b} = 2^{-b} e^{\kappa\psi} \int_0^{\infty} e^{-\omega\psi^2/2} p(\omega) d\omega,
$$

for $\kappa = a - b/2$ and $\omega \sim \text{PG}(b, 0)$, where PG denotes the Polya-gamma density. Therefore, revisiting (2), conditioning on the Polya-gamma latents, and letting $\psi_i = \boldsymbol{x}_i'\boldsymbol{\beta}$ we have:

$$\begin{aligned} p(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{\omega}) &\propto p(\boldsymbol{\beta}) \prod_{i=1}^{n} \frac{(e^{\psi_i})^{y_i}}{(1 + e^{\psi_i})^{n_i}} \\ &= p(\boldsymbol{\beta}) \prod_{i=1}^{n} \exp(\kappa_i \psi_i - \omega_i \psi_i^2/2) \\ &\propto p(\boldsymbol{\beta}) \prod_{i=1}^{n} \exp\left(-\frac{\omega_i}{2} (z_i - \psi_i)^2\right) \\ &= p(\boldsymbol{\beta}) \exp\left\{-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{\Omega}(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})\right\}, \end{aligned} \tag{3}$$

where $\kappa_i = y_i - \frac{n_i}{2}$, $z_i = \frac{\kappa_i}{\omega_i}$, and $\boldsymbol{\Omega} = \text{diag}(\omega_1, ..., \omega_n)$. From (3), $\boldsymbol{z}$ is conditionally Gaussian. That is,

$$\boldsymbol{z}|\boldsymbol{\beta}, \boldsymbol{\Omega} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{\Omega}^{-1}) \tag{4}$$

Therefore, placing a $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ prior on $\boldsymbol{\beta}$ results in the following full conditional distribution:

$$\begin{aligned} p(\boldsymbol{\beta}|\boldsymbol{z}, \boldsymbol{\Omega}) &\propto p(\boldsymbol{z}|\boldsymbol{\beta}, \boldsymbol{\Omega}) \cdot p(\boldsymbol{\beta}) \\ &\propto \exp\left\{-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})' \boldsymbol{\Omega} (\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})\right\} \exp\left\{-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\mu}_0)' \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{\beta} - \boldsymbol{\mu}_0)\right\} \\ &= \exp\left\{-\frac{1}{2}\left(\boldsymbol{z}'\boldsymbol{\Omega}\boldsymbol{z} - 2\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{z} + \boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{X}\boldsymbol{\beta}\right)\right\} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\beta}'\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\beta} - 2\boldsymbol{\beta}'\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 + \boldsymbol{\mu}_0'\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right)\right\} \\ &\propto \exp\left\{-\frac{1}{2}\left(-2\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{z} + \boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{X}\boldsymbol{\beta}\right)\right\} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\beta}'\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\beta} - 2\boldsymbol{\beta}'\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right)\right\} \\ &= \exp\left\{-\frac{1}{2}\left(-2\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{z} + \boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}'\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\beta} - 2\boldsymbol{\beta}'\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right)\right\} \\ &= \exp\left\{-\frac{1}{2}\left(-2\boldsymbol{\beta}'\left(\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{z} + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right) + \boldsymbol{\beta}'\left(\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{X} + \boldsymbol{\Sigma}_0^{-1}\right)\boldsymbol{\beta}\right)\right\} \end{aligned} \tag{5}$$

And now, a quick note on identifying kernels of a multivariate normal distribution. Suppose $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Then

$$\begin{aligned} p(\boldsymbol{\beta}) &\propto \exp\left\{-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu})\right\} \\ &\propto \exp\left\{-\frac{1}{2}\left(\boldsymbol{\beta}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\beta} - 2\boldsymbol{\beta}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\right\} \end{aligned} \tag{6}$$

Therefore, based on (6), (5) implies that

$$\boldsymbol{\beta}|\boldsymbol{z}, \boldsymbol{\Omega} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{V}), \tag{7}$$

where $\boldsymbol{V} = \left(\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{X} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1}$ and $\boldsymbol{m} = \boldsymbol{V}\left(\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{z} + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right)$. Finally, we note that $\boldsymbol{\Omega}\boldsymbol{z} = \boldsymbol{\kappa}$.

Finally, (Polson et al., 2013) note that the full conditional distribution of $\boldsymbol{\Omega}$ is also in the Polya-gamma family, and given by the following:

$$\omega_i|\boldsymbol{\beta} \sim \text{PG}(n_i, \psi_i) \tag{8}$$

where $\psi_i = \boldsymbol{x}_i'\boldsymbol{\beta}$. We omit the derivation here.

1.2 Implementation

As a motivating example, consider the famed Donner party dataset. The MLEs for an additive model with sex and age are given below.

```
data(case2001, package = 'Sleuth3')
summary(glm(Status ~ Sex + Age, family = 'binomial', data = case2001))
```

```
##
## Call:
## glm(formula = Status ~ Sex + Age, family = "binomial", data = case2001)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7445  -1.0441  -0.3029   0.8877   2.0472
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.23041    1.38686   2.329   0.0198 *
## SexMale     -1.59729    0.75547  -2.114   0.0345 *
## Age         -0.07820    0.03728  -2.097   0.0359 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 61.827  on 44  degrees of freedom
## Residual deviance: 51.256  on 42  degrees of freedom
## AIC: 57.256
##
## Number of Fisher Scoring iterations: 4
```

We now fit this model with a Gibbs sampler using the strategy described above.

```
# response, size, and design
y <- with(Sleuth3::case2001, as.numeric(Status) - 1) # died = 0, survived = 1
```

```r
size <- rep(1, nrow(Sleuth3::case2001))
n <- nrow(Sleuth3::case2001)
X <- model.matrix(~ Sex + Age, data = Sleuth3::case2001)

# precompute kappa
kappa <- y - size/2

# setup sampler and priors
num.mcmc <- 10000
p <- ncol(X)
beta.mcmc <- matrix(0, num.mcmc, p);colnames(beta.mcmc) <- colnames(X)

mu0 <- matrix(0, nrow = p, ncol = 1)
Sigma0.inv <- solve(16*diag(p))
prior.prod <- Sigma0.inv %*% mu0

# initialize
beta <- matrix(rnorm(p), ncol = 1)

# sampler
for(i in 2:num.mcmc){
  # update latent omegas
  eta <- c(X %*% beta)
  omega <- BayesLogit::rpg(n, size, eta)
  Omega <- diag(omega)

  # update beta
  V <- solve(t(X) %*% Omega %*% X + Sigma0.inv)
  m <- V %*% (t(X) %*% kappa + prior.prod)
  beta <- matrix(mvtnorm::rmvnorm(1, m, V), ncol = 1)

  # store
  beta.mcmc[i, ] <- c(beta)
}
est <- cbind(
  colMeans(beta.mcmc),
  apply(beta.mcmc, 2, sd)
); colnames(est) <- c('mean', 'sd')
est
```

```
##                    mean        sd
## (Intercept)  3.19539419 1.3008378
## SexMale     -1.57176687 0.7477836
## Age         -0.07852955 0.0357323
```

These estimates and standard deviations are consistent with those of the MLEs, which reflects our weakly informative priors.

2 Dynamic logistic regression

We now allow the regression coefficients to change over time.

$$
\begin{aligned}
y_t &\sim \text{binomial}(n_t, \pi_t), & \pi_t &= \frac{\exp(\boldsymbol{x}_t'\boldsymbol{\beta}_t)}{1+\exp(\boldsymbol{x}_t'\boldsymbol{\beta}_t)} \\
\boldsymbol{\beta}_t &= \boldsymbol{\beta}_{t-1} + \boldsymbol{V}_t, & \boldsymbol{V}_t &\sim \mathcal{N}(\mathbf{0}, \tau^2\boldsymbol{I})
\end{aligned}
$$

2.1 Derivations

Recall from Section 1.1 that if we let $z_t = \frac{1}{\omega_t}(y_t - \frac{n_t}{2})$, then $z_t|\boldsymbol{\beta}_t, \omega_t \sim N(x_t'\boldsymbol{\beta}_t, \frac{1}{\omega_t})$. Therefore, to sample from the joint posterior distribution of $\boldsymbol{\beta}_t$ and ω_t, we implement a FFBS algorithm treating z_t as working responses. The details of this algorithm are presented in (Petris, Petrone and Campagnoli, 2009), though we provide the general outline here. Our observation and evolution equations are as follows:

$$
\begin{aligned}
z_t &= x_t'\boldsymbol{\beta} + w_t, & w_t &\sim N\left(0, \frac{1}{\omega_t}\right) \\
\boldsymbol{\beta}_t &= \boldsymbol{\beta}_{t-1} + \boldsymbol{V}_t, & \boldsymbol{V}_t &\sim \mathcal{N}(\mathbf{0}, \tau^2\boldsymbol{I})
\end{aligned}
$$

To take fully Bayesian draws from the joint posterior distribution, we implement the following Gibbs sampler:

1) sample $\boldsymbol{\beta}_{1:T}|\cdot$ using a FFBS
2) sample $\omega_t|\cdot \sim \text{PG}(n_i, x_t'\boldsymbol{\beta}_t)$
3) sample $\tau|\cdot \sim \text{IG}(a_0 + \frac{T}{2}, b_0 + \frac{1}{2}\sum_{t=1}^{T}(x_t'\boldsymbol{\beta} - x_{t-1}\boldsymbol{\beta}_{t-1})^2)$

2.2 Implementation

We begin by simulating dynamic logistic data.

2.2.1 One observation per time point

We first consider the case where we observe a single binomial trial per time point.

```
set.seed(05192019)
time.pts <- 100
n <- rep(100, time.pts)
```

```
p <- 2
tau2 <- .01
beta <- matrix(0, nrow = time.pts, ncol = p)
for(t in 2:time.pts) beta[t,] <- beta[t-1,] + rnorm(p, 0, sqrt(tau2))
X <- matrix(c(
  rep(1, time.pts),
  runif(time.pts * (p-1), -1, 1)
), nrow = time.pts, ncol = p)

eta <- rowSums(X * beta)
pi <- exp(eta) / (1 + exp(eta))

df <- data.frame(
  time = 1:time.pts,
  n = n,
  y = rbinom(time.pts, n, pi),
  pi = pi
)

df <- cbind(df, X[,2:p]); names(df)[5:(3 + p)] <- paste0('x', 1:(p-1))
save(df, file = 'df.Rdata')

ggthemr::ggthemr('dust', layout = 'scientific')
ggplot(df) +
  geom_line(aes(x = time, y = pi)) +
  geom_point(aes(x = time, y = y/n), col = 'grey') +
  labs(title = "Simulated data",
       y = expression(pi)) +
  ylim(0, 1) +
  theme(axis.title.y = element_text(angle = 0, vjust = .5))
```

We now implement the sampler described in Section 2.1.

```
load('df.Rdata')

# setup sampler
time.pts <- nrow(df)
p <- 2
n <- df$n
num.mcmc <- 1000
X <- model.matrix(~ x1, df)

beta.mcmc <- array(0, dim = c(time.pts, p, num.mcmc))
dimnames(beta.mcmc)[[2]] <- c('b0', 'b1')
beta.mcmc[,,1] <- matrix(rnorm(p * time.pts), time.pts, p)
beta <- beta.mcmc[,,1]

tau2.mcmc <- rep(1, num.mcmc)
tau2 <- tau2.mcmc[1]

# priors
mu0 <- matrix(0, ncol = 1, nrow = p)
Sigma0 <- 100*diag(p)
a0 <- b0 <- .000000001
```

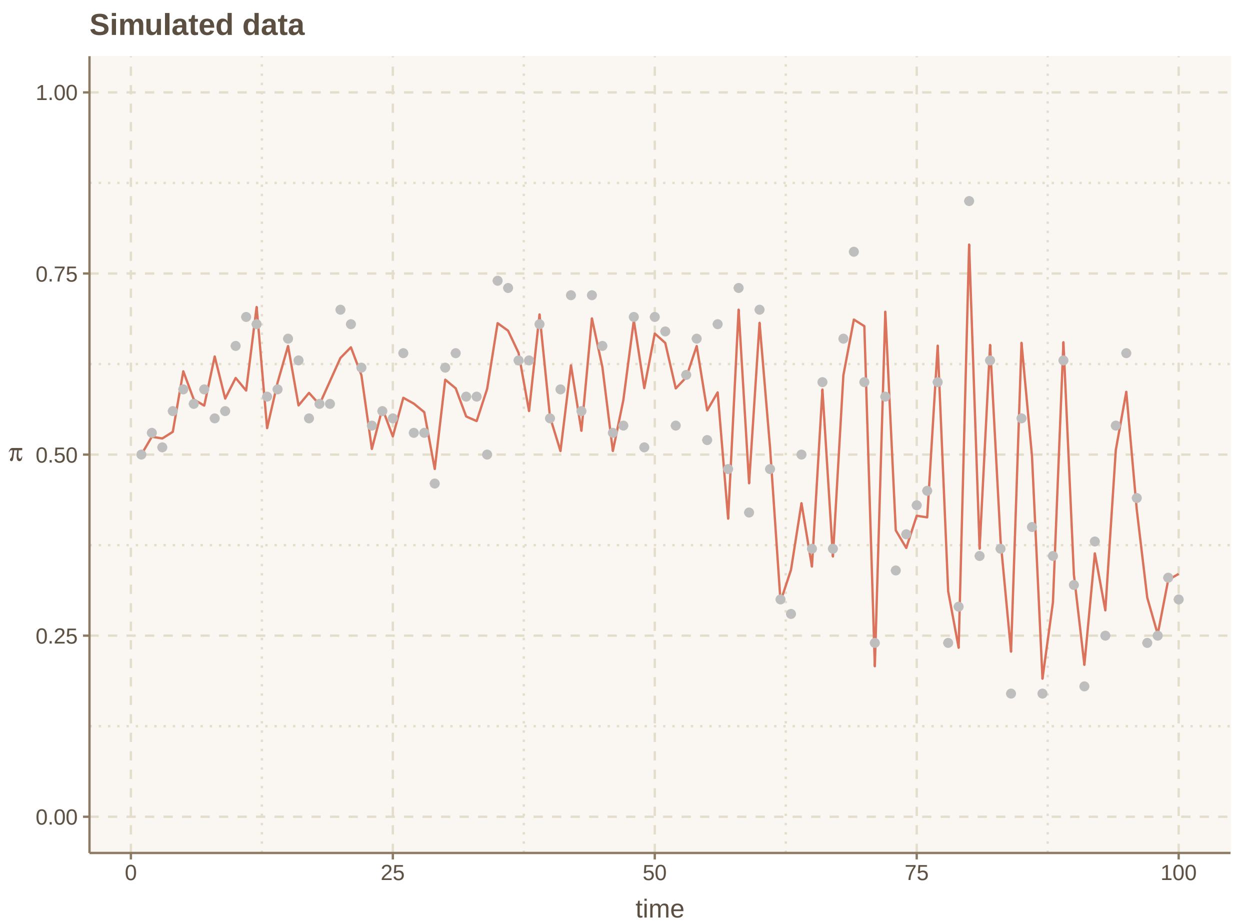

Figure 1: True probability of success in orange, observed sample proportions in grey.

```r
# precompute kappa
kappa <- with(df, y - n/2)
begin <- Sys.time()
for(i in 2:num.mcmc){
  # update omega
  eta <- rowSums(X * beta)
  omega <- BayesLogit::rpg(time.pts, n, eta)
  z <- (1/omega)*kappa

  # update betas
  beta <- ffbs(y = z, X = X, mu0 = mu0, phi0 = Sigma0, tau2 = tau2, sigma2 = 1/omega)$beta

  # update tau2
  # tau2 <- .01
  gamma.n <- a0 + .5*time.pts
  tau.n <- b0 + .5 * sum((eta[2:time.pts] - eta[1:(time.pts-1)])^2)
  tau2 <- LearnBayes::rigamma(1, gamma.n, tau.n)

  # store
  beta.mcmc[,,i] <- beta
  tau2.mcmc[i] <- tau2

  # progress
  if(F) my.prog(begin = begin, num.mcmc = num.mcmc, i = i)
}

pred.pi <- matrix(0, num.mcmc, time.pts)
for(i in 1:num.mcmc){
  eta <- rowSums(X * beta.mcmc[,,i])
  pred.pi[i,] <- exp(eta) / (1 + exp(eta))
}
df$pi.est <- colMeans(pred.pi)
df$pi.lwr <- apply(pred.pi, 2, quantile, probs = 0.025)
df$pi.upr <- apply(pred.pi, 2, quantile, probs = 0.975)

ggthemr::ggthemr('dust', layout = 'scientific')
wrapper <- function(x, ...) {
  paste(strwrap(x, ...), collapse = "\n")
}
ggplot(df) +
  geom_point(aes(x = time, y = y/n), col = 'grey', pch = 16) +
  geom_line(aes(x = time, y = pi.est)) +
  geom_line(aes(x = time, y = pi.lwr), linetype = 'dotdash') +
  geom_line(aes(x = time, y = pi.upr), linetype = 'dotdash') +
  labs(title = 'Posterior predictive summary',
       y = expression(hat(pi))) +
  ylim(0,1) +
  theme(axis.title.y = element_text(angle = 0, vjust = .5))
```

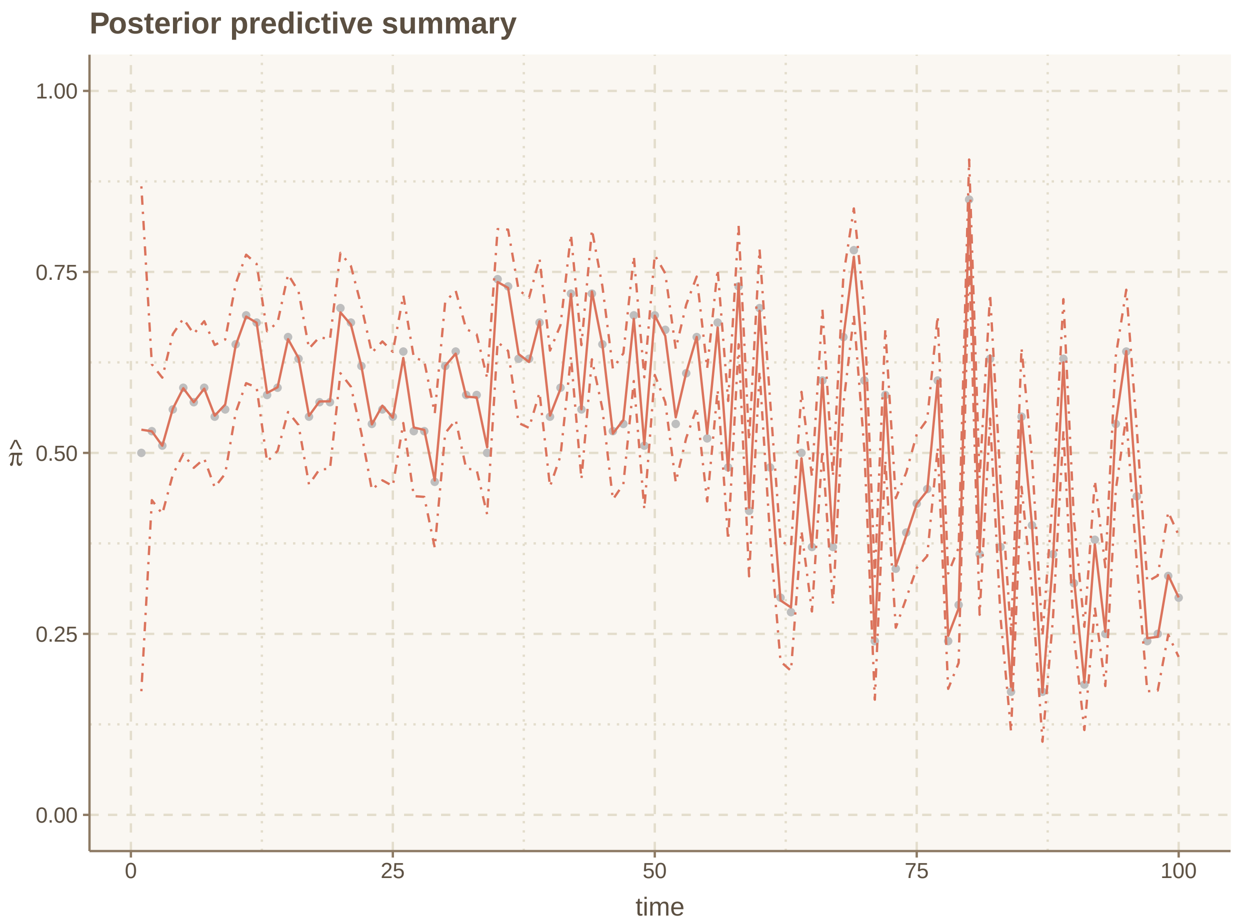

Figure 2: Posterior mean probability in solid orange, 95% credibility intervals in dotted orange, observed sample proportions in grey.

2.2.2 Multiple observations per time point

We now consider the case with multiple binomial trials per time point.

```r
set.seed(05192019)
time.pts <- 100
n <- rep(100, time.pts)
p <- 2
tau2 <- .01
beta <- matrix(0, nrow = time.pts, ncol = p)
for(t in 2:time.pts) beta[t,] <- beta[t-1,] + rnorm(p, 0, sqrt(tau2))
X <- matrix(c(
  rep(1, time.pts),
  runif(time.pts * (p-1), -1, 1)
), nrow = time.pts, ncol = p)

eta <- rowSums(X * beta)
pi <- rep(exp(eta) / (1 + exp(eta)), each = 10)

df <- data.frame(
  time = rep(1:time.pts, each = 10),
  n = rep(n, each = 10),
  y = rbinom(time.pts*10, rep(n, each = 10), pi),
  pi = pi
)

df <- cbind(df, X[,2:p]); names(df)[5:(3 + p)] <- paste0('x', 1:(p-1))
save(df, file = 'df.Rdata')

ggthemr::ggthemr('dust', layout = 'scientific')
ggplot(df) +
  geom_point(aes(x = time, y = y/n), col = 'grey') +
  geom_line(aes(x = time, y = pi)) +
  labs(title = "Simulated data",
       y = expression(pi)) +
  ylim(0, 1) +
  theme(axis.title.y = element_text(angle = 0, vjust = .5))
```

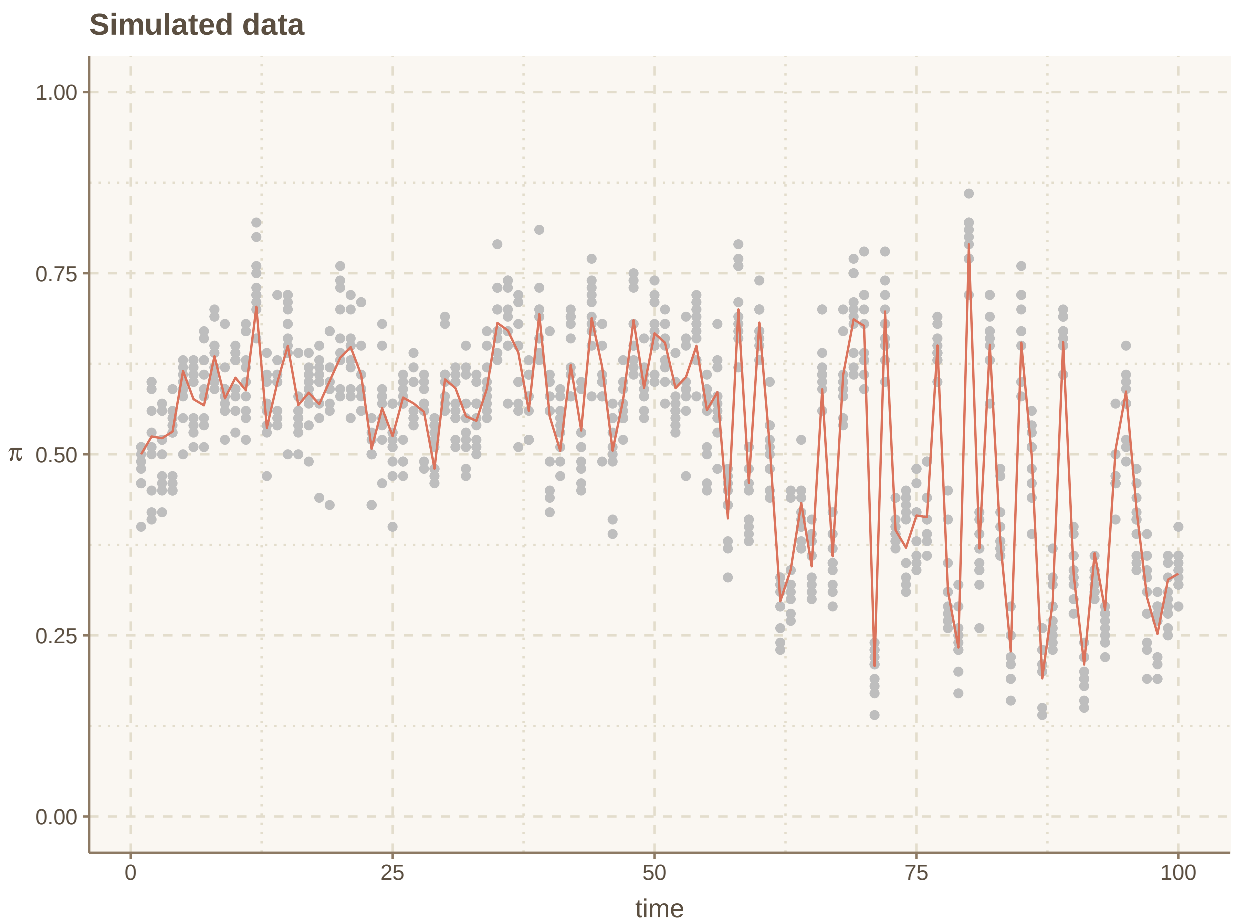

Figure 3: True probability in orange, observed sample proportions in grey.

3 Multinomial Regression

We now extend the standard logistic regression model to the multinomial case using the multinomial logit (softmax) link function.

$$\boldsymbol{y}_i|\boldsymbol{\pi}_i \sim \text{multinomial}\,(1, \boldsymbol{\pi}_i)$$
$$\pi_{ij} = \frac{\exp(\boldsymbol{x}_i'\boldsymbol{\beta}_j)}{\sum_{k=1}^{J}\exp(\boldsymbol{x}_i'\boldsymbol{\beta}_k)}$$

where $\boldsymbol{y}_i$ represents the vector of responses for the multinomial trial on observation i and $\boldsymbol{\pi}_i$ represents the vector of probabilities of success for each level of the multinomial trial, and π_{ij} represents the probability of success for level j on trial i.

3.1 Derivations

To sample the joint posterior distribution of $\boldsymbol{\beta}$, we again make use of the Polya-gamma data augmentation strategy described by (Polson et al., 2013). To do so, we require the likelihood contribution of the regression coefficients associated with one level of the response conditional on the others. (Holmes and Held, 2006) showed that this contribution is given by the following:

$$\ell(\boldsymbol{\beta}_j|\boldsymbol{\beta}_{-j}, \boldsymbol{y}) \propto \prod_{i=1}^{N}\left(\frac{e^{\eta_{ij}}}{1+e^{\eta_{ij}}}\right)^{y_{ij}}\left(\frac{1}{1+e^{\eta_{ij}}}\right)^{n_i - y_{ij}} = \prod_{i=1}^{N}\frac{(e^{\eta_{ij}})^{y_{ij}}}{(1+e^{\eta_{ij}})^{n_i}}$$

where $\eta_{ij} = \boldsymbol{x}_i'\boldsymbol{\beta}_j - c_{ij}$ and $c_{ij} = \log\left(\sum_{k\neq j}\exp(\boldsymbol{x}_i'\boldsymbol{\beta}_k)\right)$. Thus, it is clear that conditional on the regression coefficients associated with the other levels of the response, the likelihood contribution of $\boldsymbol{\beta}_j$ has the same form as that of the standard logistic regression model. Therefore, we can replicate the samplers described above, looping over $J-1$ (for identifiability) levels of the response.

If we let $z_{ij} = \frac{1}{\omega_{ij}}(y_{ij} - \frac{n_i}{2})$, then $z_{ij}|\boldsymbol{\beta}, \omega_{ij} \sim N(\eta_{ij}, \frac{1}{\omega_{ij}})$. We now derive the full conditional posterior distribution of $\boldsymbol{\beta}_j$, again assuming a $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ prior on $\boldsymbol{\beta}_j$.

$$
\begin{aligned}
p(\boldsymbol{\beta}_j|\boldsymbol{z}, \boldsymbol{\Omega}_j) &\propto p(\boldsymbol{z}|\boldsymbol{\beta}_j, \boldsymbol{\Omega}_j) \cdot p(\boldsymbol{\beta}_j) \\
&\propto \exp\left\{-\frac{1}{2}\left(\boldsymbol{z}_j - (\boldsymbol{X}\boldsymbol{\beta}_j - \boldsymbol{c}_j)\right)' \boldsymbol{\Omega}_j \left(\boldsymbol{z}_j - (\boldsymbol{X}\boldsymbol{\beta}_j - \boldsymbol{c}_j)\right)\right\} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\beta}_j - \boldsymbol{\mu}_0\right)' \boldsymbol{\Sigma}_0^{-1} \left(\boldsymbol{\beta}_j - \boldsymbol{\mu}_0\right)\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left(-2\beta_j' \boldsymbol{X}'\boldsymbol{\Omega}_j \boldsymbol{z}_j - 2\beta_j' \boldsymbol{X}'\boldsymbol{\Omega}_j \boldsymbol{c}_j + \beta_j' \boldsymbol{X}'\boldsymbol{\Omega}_j \boldsymbol{X}\beta_j\right)\right\} \exp\left\{-\frac{1}{2}\left(\beta_j' \boldsymbol{\Sigma}_0^{-1}\beta_j - 2\beta_j' \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right)\right\} \\
&= \exp\left\{-\frac{1}{2}\left(-2\beta_j' \boldsymbol{X}'\boldsymbol{\Omega}_j(\boldsymbol{z}_j + \boldsymbol{c}_j) + \beta_j' \boldsymbol{X}'\boldsymbol{\Omega}_j \boldsymbol{X}\beta_j\right)\right\} \exp\left\{-\frac{1}{2}\left(\beta_j' \boldsymbol{\Sigma}_0^{-1}\beta_j - 2\beta_j' \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right)\right\} \\
&= \exp\left\{-\frac{1}{2}\left(-2\beta_j' \left(\boldsymbol{X}'\boldsymbol{\Omega}_j(\boldsymbol{z}_j + \boldsymbol{c}_j) + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right) + \beta_j' \left(\boldsymbol{X}'\boldsymbol{\Omega}_j \boldsymbol{X} + \boldsymbol{\Sigma}_0^{-1}\right)\beta_j\right)\right\}
\end{aligned}
$$

Consequently, we have the following full conditional posterior distributions:

$$
\boldsymbol{\beta}_j|\boldsymbol{\beta}_{-j}, \boldsymbol{z}_j, \boldsymbol{\Omega}_j \sim \mathcal{N}(\boldsymbol{m}_j, \boldsymbol{V}_j)
$$

$$
\omega_{ij}|\boldsymbol{\beta}, \boldsymbol{Z} \sim \mathrm{PG}(n_i, \eta_{ij})
$$

where $\boldsymbol{V}_j = \left(\boldsymbol{X}'\boldsymbol{\Omega}_j\boldsymbol{X} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1}$, $\boldsymbol{m} = \boldsymbol{V}(\boldsymbol{X}'(\boldsymbol{\kappa_j} + \boldsymbol{\Omega_j c_j}) + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0)$, $\eta_{ij} = \boldsymbol{x}_i'\boldsymbol{\beta}_j - c_{ij}$, and $c_{ij} = \log\left(\sum_{k\neq j}\exp(\boldsymbol{x}_i'\beta_k)\right)$.

3.2 Implementation

We first generate some multinomial data.

```
N <- 100
num.trials <- 100
J <- 3
beta <- rbind(
  c(0, -2),
  c(0, 2),
  c(0, 0)
)
X <- cbind(
  rep(1, N),
  runif(N, -1, 1)
)

linpred <- X %*% t(beta)
pi <- exp(linpred) / rowSums(exp(linpred))
y <- matrix(0, N, J)
for(i in 1:N){
  y[i,] <- c(rmultinom(1, num.trials, pi[i,]))
}
df <- data.frame(cbind(
  X[,2],
```

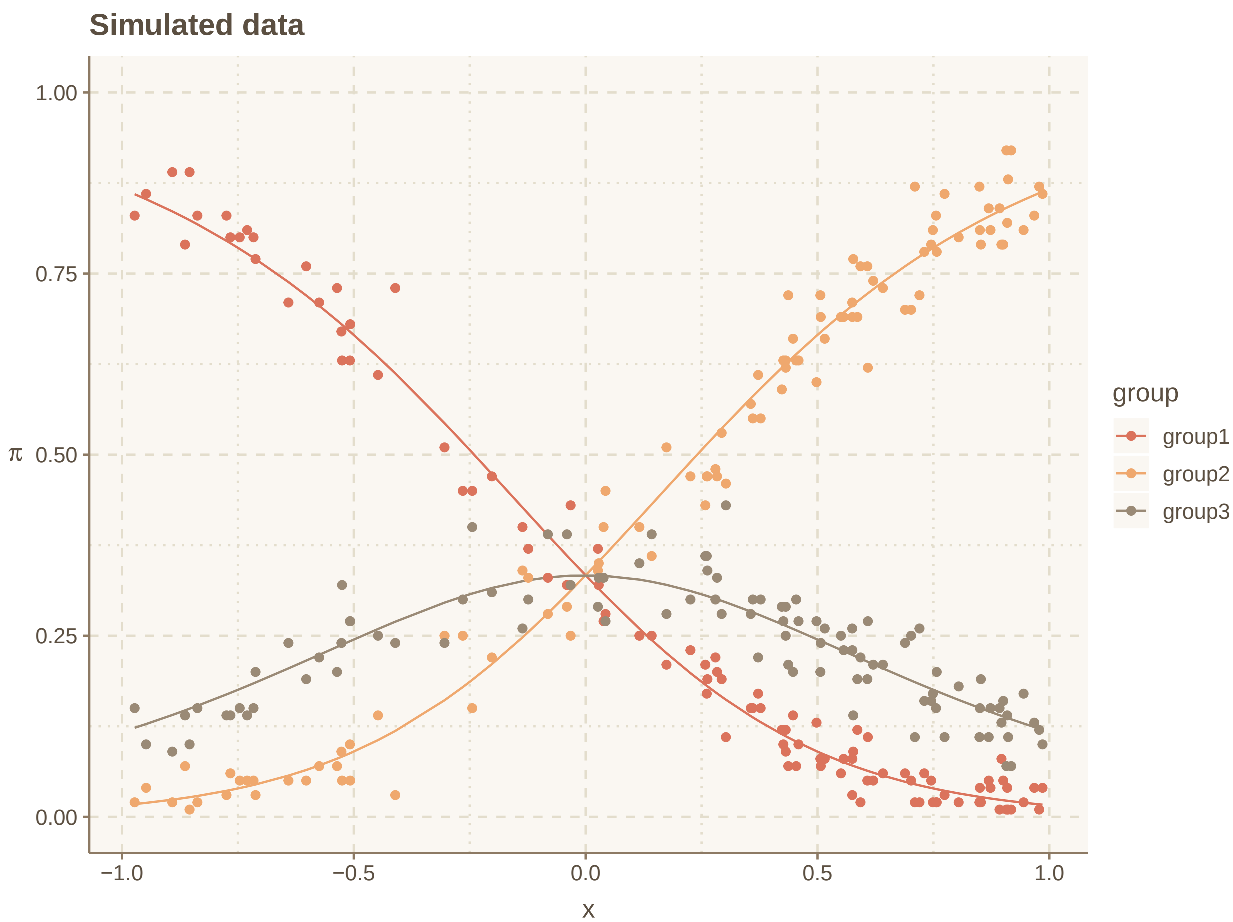

Figure 4: Simulated data. Each color represents a level of the response. The line represents the true probability, the points represent the observed proportions out of a multinomial trial with a size of 100.

```
  y,
  rep(num.trials, N)
))
names(df) <- c('x', 'group1', 'group2', 'group3', 'n')

ggthemr::ggthemr('dust', layout = 'scientific')
df %>% gather(group, count, group1:group3) %>% cbind(., pi = c(pi)) %>%
  ggplot() +
  geom_line(aes(x = x, y = pi, col = group)) +
  geom_point(aes(x = x, y = count / n, col = group)) +
  labs(title = 'Simulated data',
       y = expression(pi)) +
  ylim(0, 1) +
  theme(axis.title.y = element_text(angle = 0, vjust = .5))
```

```r
# response, size, and design
size <- rep(num.trials, N)
X <- model.matrix( ~ x, df)
J <- ncol(y)

# precompute kappa
kappa <- y - size/2

# setup sampler and priors
num.mcmc <- 1000
p <- ncol(X)
beta.mcmc <- array(0, dim = c(num.mcmc, p, J))
dimnames(beta.mcmc)[[2]] <- colnames(X)
dimnames(beta.mcmc)[[3]] <- c(paste0('y', 1:3))

mu0 <- matrix(0, nrow = p, ncol = 1)
Sigma0.inv <- solve(9*diag(p))
prior.prod <- Sigma0.inv %*% mu0

# initialize
beta.mcmc[1,,] <- matrix(rnorm(p*J), ncol = 1)

# sampler
for(i in 2:num.mcmc){
  for(j in 1:(J-1)){
    # calculate matrix of linear predictors
    linpred <- X %*% beta.mcmc[i-1,,]

    # update latent omegas
    C <- log(rowSums(exp(linpred[,-j])))
    eta <- linpred[,j] - C
    omega <- BayesLogit::rpg(N, size, eta)
    Omega <- diag(omega)

    # update beta
    V <- solve(t(X) %*% Omega %*% X + Sigma0.inv)
    m <- V %*% (t(X) %*% (kappa[,j] + Omega %*% C) + prior.prod)
    beta.mcmc[i,,j] <- matrix(mvtnorm::rmvnorm(1, m, V), ncol = 1)
  }
}
colMeans(beta.mcmc)
```

```
##                     y1         y2            y3
## (Intercept)  0.0154324 0.01381272 -0.0002284058
## x           -2.0183701 1.96109070 -0.0003076617
```

These estimates are consistent with the parameters that generated the data.

4 Dynamic multinomial regression

We now allow the regression coefficients to change over time, as we did in the logistic case.

$$\boldsymbol{y}_t \sim \text{multinomial}(n_t, \boldsymbol{\pi}_t), \qquad \boldsymbol{\pi}_t = \frac{\exp(\boldsymbol{x}_t'\boldsymbol{\beta}_{t,j})}{\sum_{k=1}^J \exp(\boldsymbol{x}_t'\boldsymbol{\beta}_{t,k})}$$

$$\boldsymbol{\beta}_{t,j} = \boldsymbol{\beta}_{t-1,j} + \boldsymbol{V}_{t,j}, \qquad \boldsymbol{V}_{t,j} \sim \mathcal{N}(\boldsymbol{0}, \tau_j^2 \boldsymbol{I})$$

4.1 Derivations

Recall from Section 3.1 that if we let $z_{t,j} = \frac{1}{\omega_{t,j}}(y_{t,j} - \frac{n_t}{2})$, then $z_{t,j}|\boldsymbol{\beta}_{t,j}, \omega_{t,j} \sim N(\eta_{t,j}, \frac{1}{\omega_{t,j}})$, where $\eta_{t,j} = \boldsymbol{x}_t'\boldsymbol{\beta}_{t,j} - c_{t,j}$ and $c_{t,j} = \log\left(\sum_{k \neq j} \exp(\boldsymbol{x}_t'\boldsymbol{\beta}_{t,k})\right)$. Therefore, to sample from the joint posterior distribution of $\boldsymbol{\beta}_{1:T,j}$ and $\omega_{t,j}$, we implement a FFBS algorithm treating $z_{t,j}$ as working responses, as we did in the logistic case. Our observation and evolution equations are as follows:

$$z_{t,j} = x_t'\boldsymbol{\beta}_{t,j} - c_{t,j} + w_{t,j}, \qquad w_{t,j} \sim N\left(0, \frac{1}{\omega_{t,j}}\right)$$

$$\boldsymbol{\beta}_{t,j} = \boldsymbol{\beta}_{t-1,j} + \boldsymbol{V}_{t,j}, \qquad \boldsymbol{V}_{t,j} \sim \mathcal{N}(\boldsymbol{0}, \tau_j^2 \boldsymbol{I})$$

To take fully Bayesian draws from the joint posterior distribution, we implement the following Gibbs sampler, looping over $J-1$ categories:

1) sample $\boldsymbol{\beta}_{1:T,j}|\cdot$ using a FFBS
2) sample $\omega_{t,j}|\cdot \sim \text{PG}(n_t, \eta_{t,j})$
3) sample $\tau_j|\cdot \sim \text{IG}(a_0 + \frac{T}{2}, b_0 + \frac{1}{2}\sum_{t=1}^T (\eta_{t,j} - \eta_{t-1,j})^2)$

4.2 Implementation

Again, we begin by generating dynamic multinomial regression data.

4.2.1 One observation per time point

We begin by consider a single multinomial trial per time point.

```
set.seed(05232019)
time.pts <- 100
size <- rep(500, time.pts)
```

```r
p <- 2
J <- 3
X <- cbind(
  rep(1, time.pts),
  runif(time.pts, -2, 2)
)

# generate dynamic data
beta <- array(0, dim = c(time.pts, p, J))
tau2 <- .01
for(t in 2:time.pts){
  for(j in 1:(J-1)){
    beta[t,, j] <- beta[t-1,, j] + rnorm(p, 0, sd = sqrt(tau2))
  }
}
linpred <- apply(beta * array(rep(X, J), dim = c(time.pts, p, J)), 3, rowSums)
pi <- exp(linpred) / (rowSums(exp(linpred)))
y.mat <- matrix(0, time.pts, J)
for(t in 1:time.pts){
  y.mat[t, ] <- rmultinom(1, size = size[t], prob = pi[t,])
}

# plot
ggthemr::ggthemr('dust', layout = 'scientific')
tibble(
  time = rep(1:time.pts, J),
  x = rep(X[,2], J),
  size = rep(size, J),
  y = c(y.mat),
  group = rep(c('a', 'b', 'c'), each = time.pts),
  pi = c(pi)
) %>%
  ggplot() +
  geom_line(aes(x = time, y = pi, col = group)) +
  geom_point(aes(x = time, y = y /size, col = group)) +
  ylim(0, 1) +
  labs(title = 'Simulated data',
       y = expression(pi)) +
  theme(axis.title.y = element_text(angle = 0, vjust = .5))
```

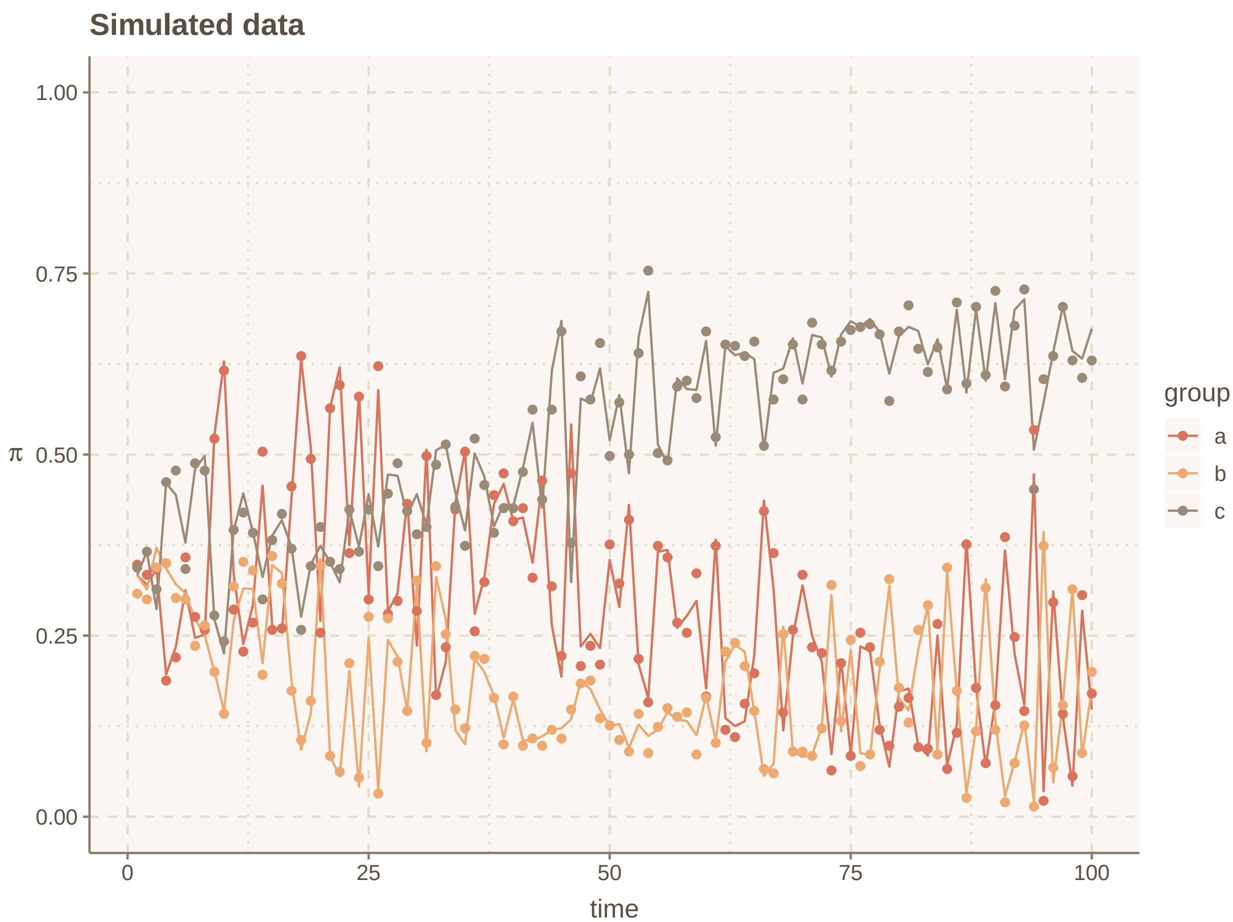

Figure 5: True probability given by the line, sample proportions of success with size of 500 given by points.

```r
# setup sampler - need: n, p, X, y, time.pts, J
num.mcmc <- 1000
n <- size

beta.mcmc <- list()
for(j in 1:(J-1)){
  beta.mcmc[[j]] <- array(0, dim = c(time.pts, p, num.mcmc))
  dimnames(beta.mcmc[[j]])[[2]] <- c('b0', 'b1')
  beta.mcmc[[j]][,,1] <- matrix(rnorm(p * time.pts), time.pts, p)
}

tau2.mcmc <- matrix(1, num.mcmc, J)
tau2 <- tau2.mcmc[1,1]

# priors
mu0 <- matrix(0, ncol = 1, nrow = p)
Sigma0 <- 100*diag(p)
a0 <- b0 <- .000000001

# precompute kappa
kappa <- y.mat - n/2
begin <- Sys.time()

# sample
for(i in 2:num.mcmc){
  for(j in 1:(J-1)){
    # organize
    beta <- beta.mcmc[[j]][,,i-1]
    y <- y.mat[,j]

    # update omega - inefficient, could speed up
    linpred <- matrix(0, time.pts, J)
    for(k in 1:(J-1)){
      linpred[,k] <- rowSums(X * beta.mcmc[[k]][,,i-1])
    }
    C <- log(rowSums(exp(linpred[,-j])))
    eta <- linpred[,j] - C
    omega <- BayesLogit::rpg(time.pts, n, eta)
    z <- (1/omega)*kappa[,j]
    z.star <- z + C

    # update betas
    beta <- ffbs(y = z.star, X = X, mu0 = mu0, phi0 = Sigma0, tau2 = tau2, sigma2 = 1/omega)$beta

    # update tau2
    # gamma.n <- a0 + .5*time.pts
    # tau.n <- b0 + .5 * sum((eta[2:time.pts] - eta[1:(time.pts-1)])^2)
    # tau2 <- LearnBayes::rigamma(1, gamma.n, tau.n)
    tau2 <- .01

    # store results
    beta.mcmc[[j]][,,i] <- beta
    tau2.mcmc[i,j] <- tau2
```

```r
  }

  # print progress
  if(F) my.prog(begin = begin, num.mcmc = num.mcmc, i = i)
}

pi.est <- array(0, dim = c(time.pts, J, num.mcmc))
eta.est <- array(0, dim = c(time.pts, J, num.mcmc))
for(j in 1:(J-1)){
  eta.est[,j,] <- t(apply(beta.mcmc[[j]], 3, FUN = function(x) rowSums(x * X)))
}

for(i in 1:num.mcmc){
  tmp <- exp(eta.est[,,i])
  pi.est[,,i] <- tmp / rowSums(tmp)
}

# estimates
apply(pi.est, c(1,2), mean)
```

```
##            [,1]      [,2]      [,3]
##  [1,] 0.2934766 0.2016252 0.5048982
##  [2,] 0.3020751 0.1830986 0.5148263
##  [3,] 0.2946844 0.1832925 0.5220232
##  [4,] 0.2939245 0.1810474 0.5250281
##  [5,] 0.2929497 0.1802894 0.5267609
##  [6,] 0.2931349 0.1800009 0.5268642
##  [7,] 0.2937004 0.1800884 0.5262112
##  [8,] 0.2930809 0.1804388 0.5264802
##  [9,] 0.2939337 0.1799670 0.5260993
## [10,] 0.2932115 0.1805781 0.5262104
## [11,] 0.2932575 0.1801582 0.5265842
## [12,] 0.2929108 0.1803501 0.5267391
## [13,] 0.2915177 0.1807699 0.5277124
## [14,] 0.2930973 0.1794771 0.5274257
## [15,] 0.2923397 0.1804178 0.5272425
## [16,] 0.2912905 0.1812583 0.5274512
## [17,] 0.2933399 0.1787761 0.5278840
## [18,] 0.2917612 0.1795474 0.5286914
## [19,] 0.2927135 0.1793040 0.5279825
## [20,] 0.2915323 0.1806742 0.5277934
## [21,] 0.2927586 0.1799916 0.5272498
## [22,] 0.2930907 0.1803611 0.5265482
## [23,] 0.2929607 0.1803224 0.5267169
## [24,] 0.2936529 0.1802682 0.5260789
## [25,] 0.2929187 0.1801509 0.5269305
## [26,] 0.2933222 0.1812742 0.5254035
## [27,] 0.2929575 0.1810310 0.5260115
## [28,] 0.2916478 0.1810986 0.5272536
## [29,] 0.2921916 0.1809002 0.5269083
## [30,] 0.2920319 0.1808923 0.5270757
## [31,] 0.2917801 0.1807184 0.5275015
## [32,] 0.2916710 0.1801704 0.5281586
## [33,] 0.2930244 0.1797045 0.5272711
```

```
## [34,] 0.2920728 0.1813958 0.5265314
## [35,] 0.2925374 0.1808345 0.5266282
## [36,] 0.2930797 0.1812816 0.5256387
## [37,] 0.2927936 0.1812953 0.5259112
## [38,] 0.2924782 0.1800737 0.5274481
## [39,] 0.2925869 0.1800308 0.5273823
## [40,] 0.2920326 0.1802963 0.5276711
## [41,] 0.2922515 0.1789851 0.5287634
## [42,] 0.2913290 0.1806974 0.5279735
## [43,] 0.2934279 0.1796630 0.5269090
## [44,] 0.2920506 0.1800409 0.5279086
## [45,] 0.2921403 0.1806515 0.5272082
## [46,] 0.2937708 0.1796638 0.5265654
## [47,] 0.2912692 0.1811779 0.5275529
## [48,] 0.2940583 0.1789560 0.5269857
## [49,] 0.2924607 0.1808397 0.5266996
## [50,] 0.2930099 0.1798359 0.5271542
## [51,] 0.2917902 0.1803973 0.5278125
## [52,] 0.2914443 0.1801257 0.5284300
## [53,] 0.2929478 0.1788247 0.5282275
## [54,] 0.2935761 0.1800091 0.5264148
## [55,] 0.2936905 0.1798573 0.5264522
## [56,] 0.2931377 0.1793694 0.5274929
## [57,] 0.2915565 0.1805058 0.5279376
## [58,] 0.2922048 0.1802163 0.5275790
## [59,] 0.2909923 0.1803747 0.5286330
## [60,] 0.2920232 0.1798926 0.5280842
## [61,] 0.2913782 0.1800001 0.5286217
## [62,] 0.2924974 0.1791888 0.5283138
## [63,] 0.2923718 0.1790012 0.5286270
## [64,] 0.2924828 0.1790807 0.5284365
## [65,] 0.2916363 0.1799460 0.5284177
## [66,] 0.2924736 0.1800131 0.5275133
## [67,] 0.2920878 0.1801418 0.5277704
## [68,] 0.2919016 0.1803248 0.5277737
## [69,] 0.2924924 0.1787904 0.5287171
## [70,] 0.2916961 0.1804378 0.5278661
## [71,] 0.2921236 0.1802650 0.5276114
## [72,] 0.2925231 0.1801090 0.5273680
## [73,] 0.2931057 0.1805285 0.5263658
## [74,] 0.2929340 0.1812722 0.5257939
## [75,] 0.2921865 0.1806035 0.5272100
## [76,] 0.2939398 0.1792136 0.5268466
## [77,] 0.2925946 0.1797045 0.5277009
## [78,] 0.2930351 0.1795890 0.5273758
## [79,] 0.2924859 0.1806046 0.5269096
## [80,] 0.2927977 0.1803634 0.5268389
## [81,] 0.2918437 0.1814176 0.5267387
## [82,] 0.2925165 0.1805449 0.5269385
## [83,] 0.2932010 0.1797363 0.5270627
## [84,] 0.2922182 0.1811713 0.5266105
## [85,] 0.2920051 0.1804876 0.5275072
## [86,] 0.2918976 0.1799943 0.5281081
## [87,] 0.2919318 0.1798387 0.5282295
```

```
##  [88,] 0.2930425 0.1791003 0.5278572
##  [89,] 0.2915186 0.1793913 0.5290901
##  [90,] 0.2917113 0.1796450 0.5286437
##  [91,] 0.2919666 0.1795495 0.5284839
##  [92,] 0.2921374 0.1804437 0.5274189
##  [93,] 0.2942060 0.1795505 0.5262434
##  [94,] 0.2921114 0.1800888 0.5277997
##  [95,] 0.2924105 0.1800660 0.5275236
##  [96,] 0.2926924 0.1802136 0.5270941
##  [97,] 0.2928845 0.1791538 0.5279618
##  [98,] 0.2922119 0.1805717 0.5272163
##  [99,] 0.2933397 0.1797744 0.5268859
## [100,] 0.2915215 0.1805410 0.5279375
```

Appendix S1: R Functions

Any functions not explicitly defined in the document above are defined here.

```
ffbs <- function(y, X, mu0, phi0, tau2, sigma2) {
    # setup storage
    y <- matrix(y, ncol = 1)
    time.pts <- nrow(y)
    p <- ncol(X)
    beta <- matrix(0, time.pts, p)
    m.mcmc <- matrix(0, time.pts, p)
    C.mcmc <- matrix(0, time.pts, p^2)

    if (length(sigma2) == 1)
        sigma2 <- rep(sigma2, time.pts)

    # forward filter
    m.t <- matrix(mu0, time.pts, p)
    C.t <- matrix(0, time.pts, p^2)
    C.t[1, ] <- c(phi0 * diag(p))
    a.t <- matrix(0, time.pts, p)
    R.t <- matrix(0, time.pts, p^2)
    # G.t <- diag(p)

    W.t <- tau2 * diag(p)
    for (t in 2:time.pts) {
        F.t <- t(X[t, ])

        Cmat.t <- matrix(C.t[t - 1, ], p, p)

        # a.t[t,] <- G.t %*% m.t[t-1, ]
        a.t[t, ] <- m.t[t - 1, ]
        # R.t[t,] <- c(G.t %*% Cmat.t %*% t(G.t) + W.t)
        R.t[t, ] <- c(Cmat.t + W.t)

        Rmat.t <- matrix(R.t[t, ], p, p)

        f.t <- F.t %*% a.t[t, ]
        Q.t <- F.t %*% Rmat.t %*% t(F.t) + sigma2[t]

        Qinv.t <- solve(Q.t)

        m.t[t, ] <- a.t[t, ] + Rmat.t %*% t(F.t) %*% Qinv.t %*% (y[t, ] - f.t)
        C.t[t, ] <- c(Rmat.t - Rmat.t %*% t(F.t) %*% Qinv.t %*% F.t %*% Rmat.t)
        m.mcmc[t, ] <- m.t[t, ]
        C.mcmc[t, ] <- C.t[t, ]
    }

    # backwards sample
    beta[time.pts, ] <- mvtnorm::rmvnorm(1, m.t[time.pts, ], sigma = matrix(C.t[time.pts,
        ], p, p))
    for (t in 1:(time.pts - 1)) {
        ndx <- time.pts - t
        h.t <- m.t[ndx, ] + matrix(C.t[ndx, ], p, p) %*% solve(matrix(R.t[ndx +
```

```
            1, ], p, p)) %*% (beta[ndx + 1, ] - a.t[ndx + 1, ])
        H.t <- matrix(C.t[ndx, ], p, p) - matrix(C.t[ndx, ], p, p) %*% solve(matrix(R.t[ndx +
            1, ], p, p)) %*% matrix(C.t[ndx, ], p, p)
        beta[ndx, ] <- mvtnorm::rmvnorm(1, h.t, sigma = round(H.t, 5))
    }

    out <- list(beta = beta, m = m.mcmc, C = C.mcmc)
    return(out)
}
```

```
my.prog <- function(print = .05*num.mcmc, begin, num.mcmc, i){
  if(i %% print == 0){
    cat("\014")
    runtime <- (Sys.time() - begin)
    percent <- round(i/num.mcmc * 100, 2)
    message <- paste('\nIteration ', i, ' of ', num.mcmc, '; ', percent, '% done. Current runtime of ',
    cat(message)
    txtProgressBar(min = 2, max = num.mcmc, initial = i, style = 3)
  }
}
```

References

Holmes, C.C. and Held, L. (2006) Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Analysis*, **1**, 145–168.

Petris, G., Petrone, S. and Campagnoli, P. (2009) *Dynamic Linear Models with R*. Springer.

Polson, N.G., Scott, J.G. and Windle, J. (2013) Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American Statistical Association*, **108**, 1339–1349.