AMS 207 HW Assignment 2

April 14 2018

List of questions:

1. 5.10: prove the posterior is improper
2. 5. 14(c)
3. SAT example with unknown variance
4. 7.5 power transformation normal (d) (e)
5. 7.4: the marginal distribution
6. 8.11
7. 8.14 (c)

1. Total law of expectation and Total law of Variance

5.7 Continous Mixuture Model

(a) $y|\theta \sim Poisson(\theta), \theta \sim Gamma(\alpha, \beta)$

$$E(y) = E(E(y|theta)) = E(\theta) = \frac{\alpha}{\beta}$$

$$Var(y) =]Var(E(y|\theta)) + E(Var(y|\theta)) = \frac{\alpha}{\beta^2} + \frac{\alpha}{\beta} = \frac{\alpha + \alpha\beta}{\beta^2}$$

(b) $y|\mu, \sigma^2 \sim N(\mu, \sigma^2)$, $p(\mu, \sigma^2) \propto 1/\sigma^2$.

$$\mu|\bar{y}, \sigma^2 \sim N(\bar{y}, \sigma^2/n)$$

$$\sigma^2|y \sim IG(\frac{n-1}{2}, \frac{S^2}{2})$$

.

Where $S^2 = \sum_{i=1}^{n}(X_i - \bar{X})^2$, $s^2 = \frac{1}{n-1}S^2$

$$E(\frac{\sqrt{n}(\mu - \bar{y})}{s}) = \frac{\sqrt{n}}{s}[E(\mu) - \bar{y}] = 0$$

$$Var(\frac{\sqrt{n}(\mu - \bar{y})}{s}) = \frac{n}{S^2}[Var(E(\mu) - \bar{y}) + E(Var(\mu - \bar{y}))] = \frac{n}{s^2}E(\sigma^2/n) = \frac{n}{S^2/(n-1)}\frac{S^2}{n(n-3)} = \frac{n}{n-3}$$

5.12 Meta analysis normal hierarchical model

$$\begin{aligned}
E(\theta_j|\tau,y) &= E_{\mu|\tau,y}(E(\theta_j|\mu,\tau,y)) \\
&= E_{\mu|\tau,y}(\frac{\frac{1}{\sigma^2}\bar{y_{.j}}}{\frac{1}{\sigma_j^2}+\frac{1}{\tau^2}} + \frac{1/\tau^2}{1/\tau_j^2+1\tau^2}\mu) \\
&= \frac{\frac{1}{\sigma^2}\bar{y_{.j}}}{\frac{1}{\sigma_j^2}+\frac{1}{\tau^2}} + \frac{1/\tau^2}{1/\tau_j^2+1\tau^2}\frac{\sum_{j=1}^J \frac{1}{\sigma_j^2+\tau^2}\bar{y_{.j}}}{\sum_{j=1}^J \frac{1}{\sigma_j^2+\tau^2}}
\end{aligned}$$

$$\begin{aligned}
Var(\theta_j|\tau,y) &= E_{\mu|\tau,y}(Var(\theta_j|\mu,\tau,y)) + Var_{\mu|\tau,y}(E(\theta_j|\mu,\tau,y)) \\
&= \frac{1}{1/\sigma^2+1/\tau^2} + (\frac{1/\tau^2}{1/\sigma^2+1/\tau^2})^2 \sum_{j=1}^J \frac{1}{\sigma_j^2+\tau^2}
\end{aligned}$$

2. Discrete Mixture Model with conjugacy

$p_m(\theta)$ for $m = 1, \cdots, M$ are conjugates prior densities for the sampling model $y|\theta$.

(a) Prove that prior defined by the mixture $p(\theta) = \sum_{i=1}^M \lambda_m p_m(\theta)$ is a conjugate prior.

$$f(\theta|y) = \frac{f(y|\theta)\sum_{m=1}^M \lambda_m p_m(\theta)}{m(y)}$$

$$\begin{aligned}
m(y) &= \int f(y|\theta) \sum_{m=1}^M \lambda_m p_m(\theta) d\theta \\
&= \sum_{m=1}^M \int f(y|\theta)\lambda_m p_m(\theta) d\theta \\
&= \sum_{m=1}^M \lambda_m p_m(y)
\end{aligned}$$

where $p_m(y)$ is the marginal given $p_m(\theta)$ prior.

$$\begin{aligned}
f(\theta|y) &= \frac{1}{\sum_{i=1}^M \lambda_m p_m(y)} \sum_{m=1}^M \lambda_m f(y|\theta) p_m(\theta) \\
&= \sum_{m=1}^M \frac{\lambda_m p_m(y)}{\sum_{i=1}^M \lambda_m p_m(y)} \frac{f(y|\theta)p_m(\theta)}{p_m(y)} \\
&= \sum_{m=1}^M \frac{\lambda_m p_m(y)}{\sum_{i=1}^M \lambda_m p_m(y)} p_m(\theta|y)
\end{aligned}$$

The posterior is a mixture of posterior distributions $p_m(\theta|y)$ with updated weights

$$\lambda_m' = \frac{\lambda_m p_m(y)}{\sum_{i=1}^M \lambda_m p_m(y)}$$

(b) Bimodal prior desnity for normal mean

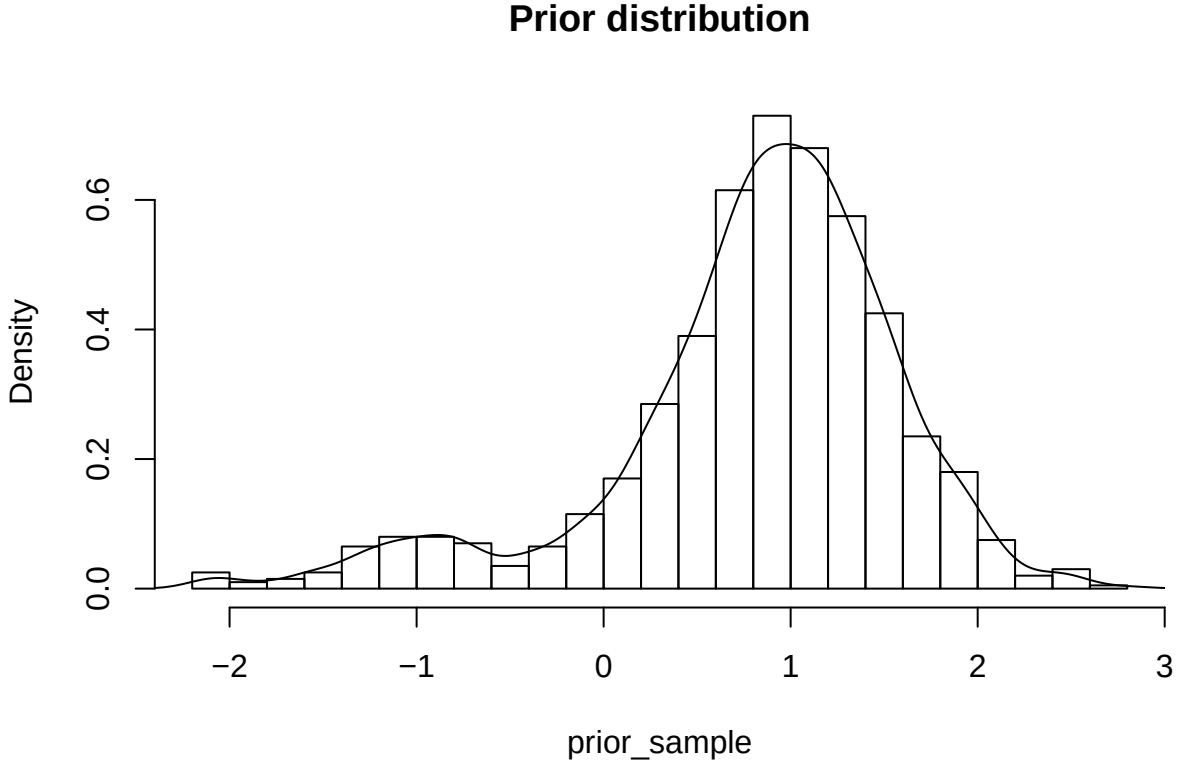
$$p(\theta) = cN(1, 0.5^2) + (1-c)N(-1, 0.5^2)$$

$$f(\theta|\bar{y}) = \frac{c\phi(\bar{y}|1, 1+0.5^2)}{c\phi(\bar{y}|1, 1+0.5^2) + (1-c)\phi(\bar{y}|-1, 1+0.5^2)}\phi(\theta|\mu_1^*, \sigma^*) + \frac{(1-c)\phi(\bar{y}|-1, 1+0.5^2)}{c\phi(\bar{y}|1, 1+0.5^2) + (1-c)\phi(\bar{y}|-1, 1+0.5^2)}\phi(\theta|\mu_2^*, \sigma^*)$$

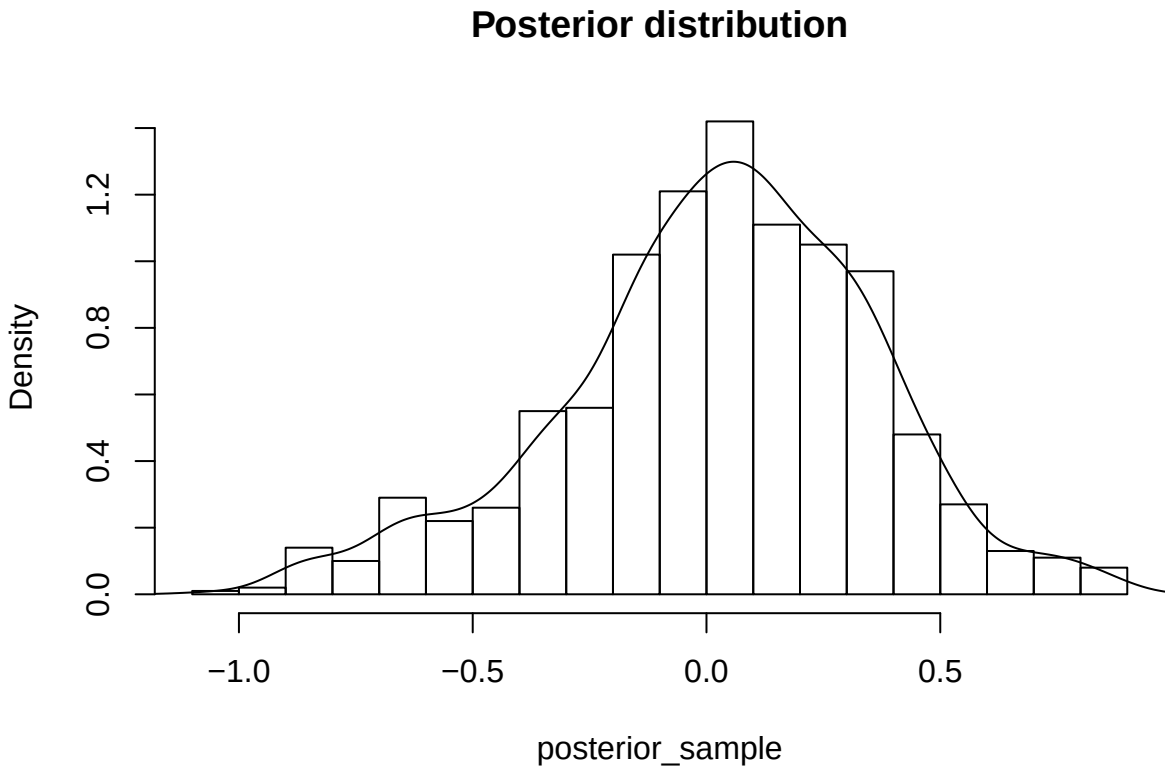
$$\mu_1^* = \frac{n\bar{y}0.5^2 + 1}{n0.5^2 + 1} \quad \mu_2^* = \frac{n\bar{y}0.5^2 - 1}{n0.5^2 + 1}$$

$$\sigma^* = \frac{0.5^2}{n0.5^2 + 1}$$

```
n = 10
c = 0.9
y_bar = -0.25
sample_size = 1000

get_prior = function(c) ifelse(runif(1) < c, rnorm(1, 1, 0.5),
    rnorm(1, -1, 0.5))

prior_sample = sapply(seq(1, sample_size), function(x) {
    get_prior(c)
})

hist(prior_sample, breaks = 20, main = "Prior distribution",
    prob = T)
lines(density(prior_sample))
```



```r
mu_1 = (n * y_bar * 0.5^2 + 1)/(n * 0.5^2 + 1)
mu_2 = (n * y_bar * 0.5^2 - 1)/(n * 0.5^2 + 1)
sd_post = 0.5/sqrt(n * 0.5^2 + 1)

c_prime = c * dnorm(y_bar, 1, sqrt(1 + 0.5^2))/(c * dnorm(y_bar,
    1, sqrt(1 + 0.5^2)) + (1 - c) * dnorm(y_bar, -1, sqrt(1 +
    0.5^2)))

get_porsterior = function(c_prime, mu_1, sd_post) ifelse(runif(1) <
    c_prime, rnorm(1, mu_1, sd_post), rnorm(1, mu_2, sd_post))
posterior_sample = sapply(seq(1, sample_size), function(x) {
    get_porsterior(c_prime, mu_1, sd_post)
})
hist(posterior_sample, breaks = 20, main = "Posterior distribution",
    prob = T)
lines(density(posterior_sample))
```



3. Hierarchical Models

Poisson-Gamma hierachical model, BDA 5. 14

a. Define hyperprior

First attempt $p(\alpha, \beta) \propto \frac{1}{\beta}$.

b. Write down the joint marginal posterior of the hyperparams

1. The marginal of $y_i|\alpha, \beta$ is a negative binomial
2. The joint marginal posterior for α, β is

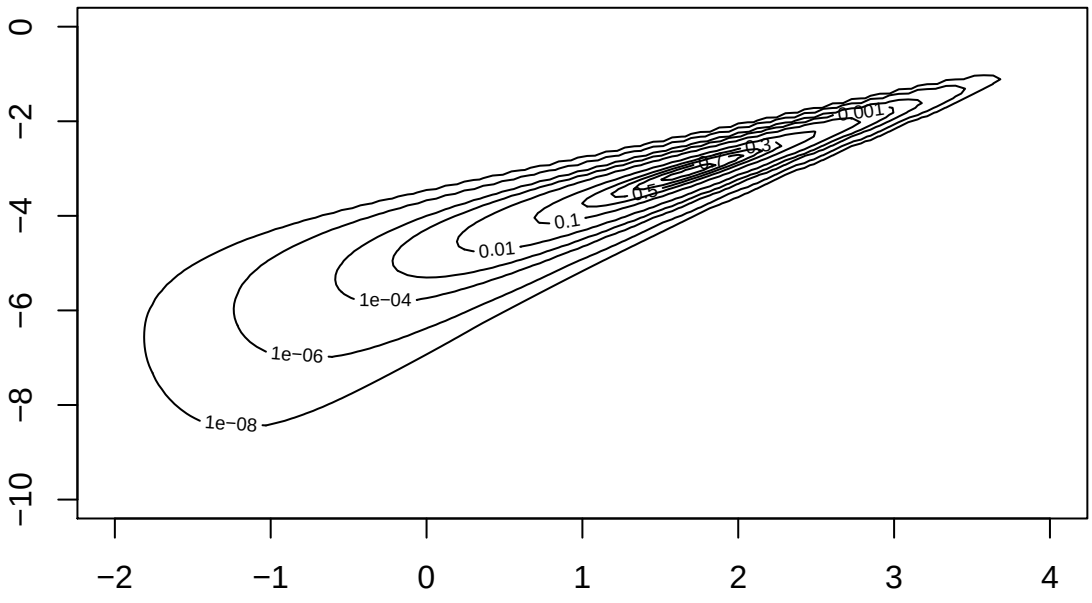
```
log_marginal_posterior = function(t_v_vec, dat) {
    t = t_v_vec[1]
    exp_t = exp(t)
    v = t_v_vec[2]
    exp_v = exp(v)
    sum_dat = sum(dat)
    J = length(dat)
    part_1 = sum(sapply(dat, function(x) {
        lgamma(exp_t + x) - (lgamma(exp_t) + lgamma(x + 1))
    }))

    part_2 = J * exp_t * v - (J * exp_t + sum_dat) * log(exp_v +
        1) + t

    return(part_1 + part_2)
}


dat = c(16 + 58, 9 + 90, 10 + 48, 13 + 57, 19 + 103, 20 + 57,
    18 + 86, 17 + 112, 35 + 273, 55 + 64)

grid_len = 100
t_grid = seq(-2, 4, length.out = grid_len)
v_grid = seq(-10, 0, length.out = grid_len)
joint_grid = expand.grid(t_grid, v_grid)
dim(joint_grid)
```

```
## [1] 10000     2
```

```
log_mat = matrix(apply(joint_grid, 1, function(x) log_marginal_posterior(x,
    dat)), nrow = grid_len)

dim(log_mat)
```

```
## [1] 100 100
```

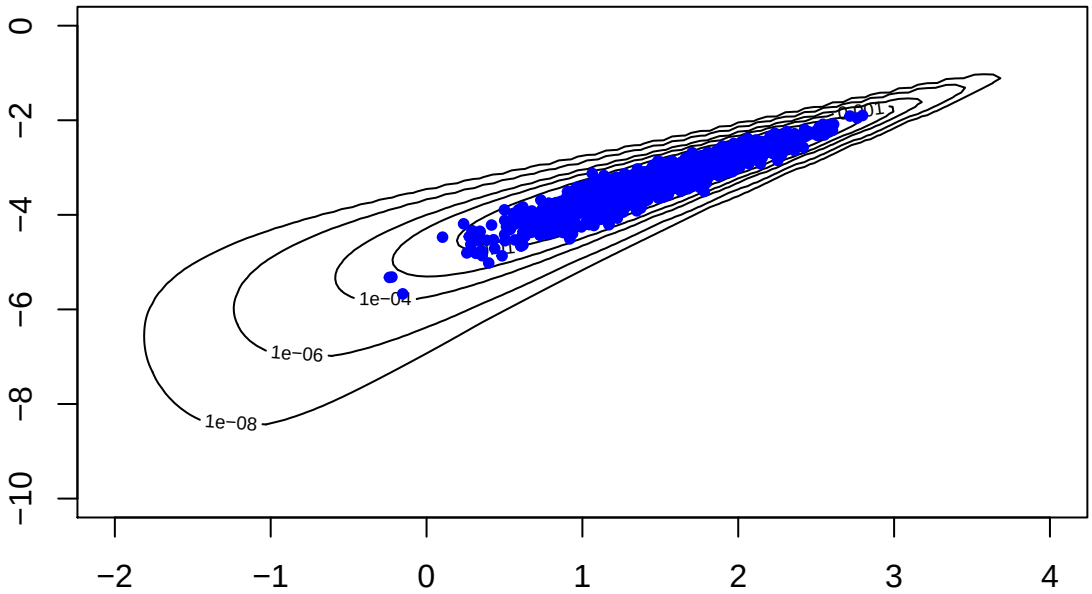
```
contours_norm = c(1e-08, 1e-06, 1e-04, 0.001, 0.01, seq(0.1,
    0.9, 0.2))
contour(t_grid, v_grid, exp(log_mat - max(log_mat)), levels = contours_norm,
    ylim = c(-10, 0), xlim = c(-2, 4))
```



c. Draw from the joint posterior and plot the scatterplots

```
library(mvtnorm)
optim_result = optim(c(1, -2), function(x) log_marginal_posterior(x,
    dat), hessian = T, control = list(fnscale = -1))

optim_loc = optim_result$par
optim_var = solve(-optim_result$hessian)

log_density_ratio = function(t_v_vec, dat, optim_loc, optim_var,
    df) {
    part_1 = log_marginal_posterior(t_v_vec, dat)
    part_2 = mvtnorm::dmvt(t_v_vec, optim_loc, optim_var, df = df,
        log = T)
    return(part_1 - part_2)
}

sir_sample_v_t = function(t_loc, t_var, dat, df, sample_size) {
    num_iter_needed = 5 * sample_size
    proposal_sample = rmvt(num_iter_needed, delta = t_loc, sigma = t_var,
        df = df)
    log_den_ratio = apply(proposal_sample, 1, function(x) {
        log_density_ratio(x, dat, t_loc, t_var, df)
    })
    den_ratio = exp(log_den_ratio)

    resample_weights = den_ratio/sum(den_ratio)
    index_selected = sample(1:num_iter_needed, size = sample_size,
        prob = resample_weights, replace = T)
    output = proposal_sample[index_selected, ]
    return(output)
}
```

```
df = 3
sample_size = 2000
posterior_sample = sir_sample_v_t(optim_loc, 2 * optim_var, dat,
    df, sample_size)

contours_norm = c(1e-08, 1e-06, 1e-04, 0.001, 0.01, seq(0.1,
    0.9, 0.2))
contour(t_grid, v_grid, exp(log_mat - max(log_mat)), levels = contours_norm,
    ylim = c(-10, 0), xlim = c(-2, 4))
points(posterior_sample[, 1], posterior_sample[, 2], pch = 20,
    col = "blue")
```



d. Is the posterior integrable? Looking at the contour plot, the posterior is integrable since the contour plot decays rather fast around the center. So it will be integrable.

e. Draw from the joint posterior of the parameters and hyperparameters.

1. Draw from the joint posterior of the hyperparameters. 2. Draw parameter from the conditional posterior $f(\theta|\alpha, \beta, y)$

The conditional posterior $\theta_j|\alpha, \beta, y_j \sim Ga(\alpha + y_j, \beta + 1)$

```
alpha_beta_sample = exp(posterior_sample)
param_posterior_param = lapply(as.list(dat), function(x) {
    cbind(alpha_beta_sample[, 1] + x, alpha_beta_sample[, 2] +
        1)
})
theta_posterior_sample = lapply(param_posterior_param, function(x) {
    rgamma(sample_size, shape = x[, 1], rate = x[, 2])
})
sum_stats = function(x) {
    c(mean(x), sd(x), quantile(x, c(0.025, 0.25, 0.5, 0.75, 0.975)))
}

theta_posterior_summ = lapply(theta_posterior_sample, sum_stats)
present_table = matrix(unlist(theta_posterior_summ), nrow = length(dat),
```

```
    byrow = T)
row.names(present_table) = c("$\\theta_1$", "$\\theta_2$", "$\\theta_3$",
    "$\\theta_4$", "$\\theta_5$", "$\\theta_6$", "$\\theta_7$",
    "$\\theta_8$", "$\\theta_9$", "$\\theta_{10}$")
knitr::kable(present_table, col.names = c("Mean", "Sd", "2.5%",
    "25%", "50%", "75%", "97.5%"))
```

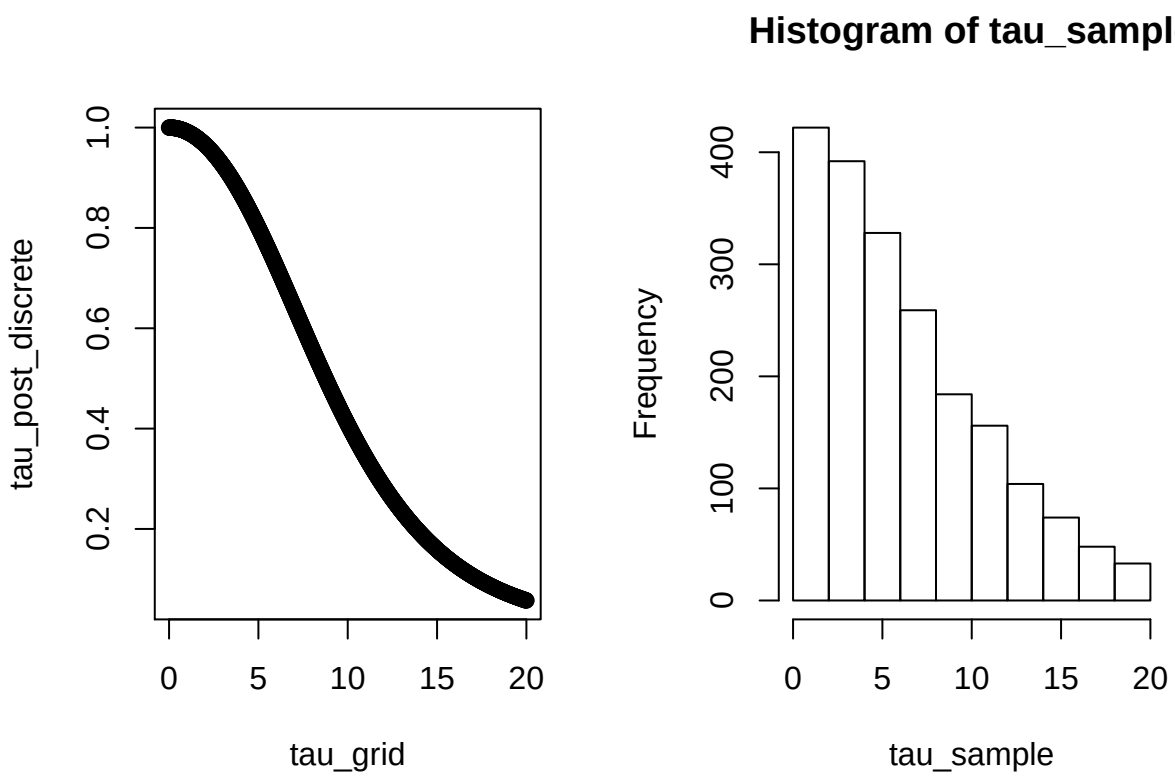
	Mean	Sd	2.5%	25%	50%	75%	97.5%
θ_1	75.75698	8.688157	59.56135	69.63315	75.42753	81.58680	93.60940
θ_2	99.68150	10.064253	81.13067	92.83463	99.63549	106.47189	120.30376
θ_3	60.46837	7.709734	46.12734	55.20746	60.10074	65.68694	76.48367
θ_4	71.53427	8.298472	55.55171	65.87711	71.24049	77.05074	88.50310
θ_5	121.90501	10.618895	101.37918	114.95907	121.65904	129.01991	142.67945
θ_6	78.31662	8.684089	62.06449	72.43816	77.91480	83.83910	96.86511
θ_7	104.31297	10.128752	85.30338	97.50604	103.89282	111.22182	124.61024
θ_8	128.30601	10.822161	108.51420	120.83816	127.79583	135.36157	150.46154
θ_9	299.37051	17.641202	265.43045	287.48021	299.24561	311.27874	333.48434
θ_{10}	118.75947	10.640592	99.24670	111.71848	118.20905	125.62690	141.09935

Repeat the SAT example

1. Direct Sampling

```
dat = c(29.39, 7.94, -2.75, 6.82, -0.64, 0.63, 18.01, 12.16)
sigma_dat = c(14.9, 10.2, 16.3, 11, 9.4, 11.4, 10.4, 17.6)
sigma_sqr_dat = sigma_dat^2

tau_log_posterior = function(tau, dat, sigma_sqr_dat) {
    tau_sqr = tau^2
    v_mu = 1/sum(1/(sigma_sqr_dat + tau_sqr))
    part_1 = 0.5 * log(v_mu)
    part_2 = sum(-0.5 * log(sigma_sqr_dat + tau_sqr))
    mu_hat = sum(dat/(sigma_sqr_dat + tau_sqr)) * v_mu
    part_3 = sum(-0.5 * ((dat - mu_hat)^2/(sigma_sqr_dat + tau_sqr)))
    return(part_1 + part_2 + part_3)
}



tau_opt_result = optim(1, function(x) {
    tau_log_posterior(x, dat, sigma_sqr_dat)
}, method = "Brent", lower = 0, upper = 300, hessian = T, control = list(fnscale = -1))
tau_post_mode = tau_opt_result$value
tau_post_var = 1/(-tau_opt_result$hessian)

tau_grid = seq(1e-05, 20, length.out = 1000)
tau_grid_log_den = sapply(tau_grid, function(x) {
    tau_log_posterior(x, dat, sigma_sqr_dat)
})
tau_post_discrete = exp(tau_grid_log_den - max(tau_grid_log_den))
sample_size = 2000
```

```r
tau_sample = sample(tau_grid, sample_size, prob = tau_post_discrete,
    replace = T)

par(mfrow = c(1, 2))
plot(tau_post_discrete ~ tau_grid)
hist(tau_sample)
```



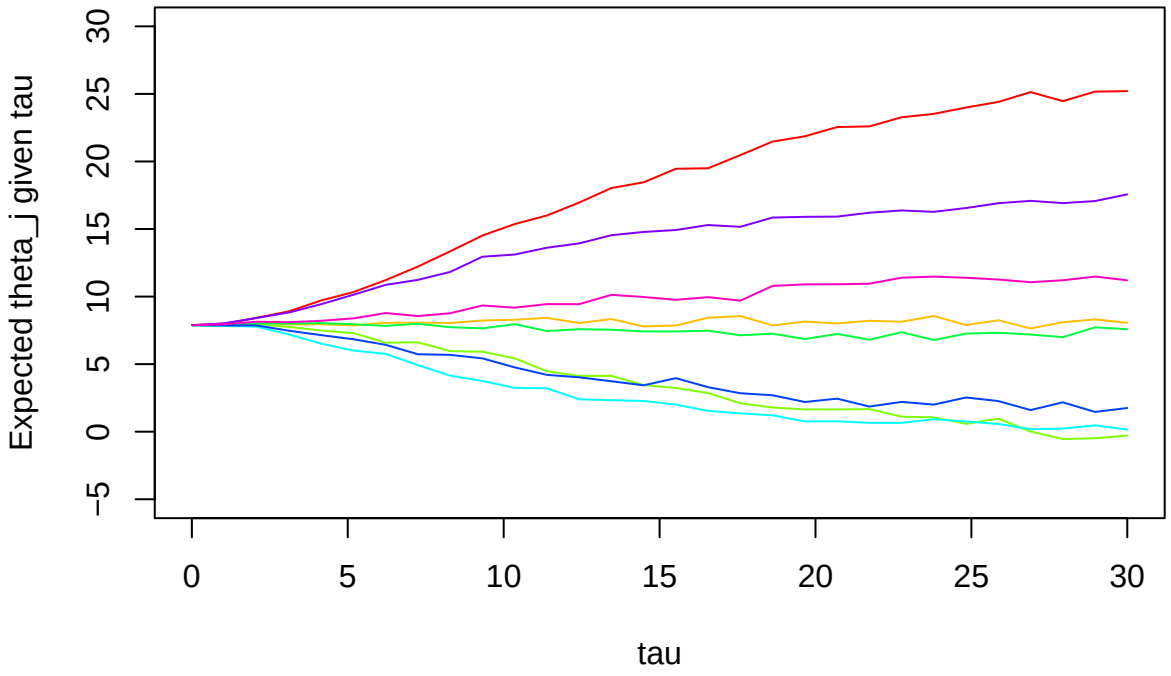

```r
sample_mar_mu = function(tau, sigma_sqr_dat, dat) {
    tau_sqr = tau^2
    v_mu = 1/sum(1/(sigma_sqr_dat + tau_sqr))
    mu_hat = sum(dat/(sigma_sqr_dat + tau_sqr)) * v_mu
    mu = rnorm(1, mu_hat, sd = sqrt(v_mu))
    return(mu)
}

mu_sample = sapply(tau_sample, function(x) {
    sample_mar_mu(x, sigma_sqr_dat, dat)
})



sample_theta_j_post = function(mu_sample, tau_sample, dat_j,
    sigma_j) {
    top_1 = dat_j * tau_sample^2
    top_2 = mu_sample * sigma_j
    bottom = tau_sample^2 + sigma_j
    mu = (top_1 + top_2)/bottom
    var = tau_sample^2 * sigma_j/(sigma_j + tau_sample^2)
    sample_size = length(mu_sample)
```

```
    return(rnorm(sample_size, mu, sd = sqrt(var)))
}

theta_sample = lapply(as.list(1:length(dat)), function(x) {
    sample_theta_j_post(mu_sample, tau_sample, dat[x], sigma_sqr_dat[x])
})

theta_post_summ = lapply(theta_sample, sum_stats)

output_table = matrix(unlist(theta_post_summ), nrow = length(dat),
    byrow = T)

row.names(output_table) = c("$\\theta_1$", "$\\theta_2$", "$\\theta_3$",
    "$\\theta_4$", "$\\theta_5$", "$\\theta_6$", "$\\theta_7$",
    "$\\theta_8$")

knitr::kable(output_table, col.names = c("Mean", "Sd", "2.5%",
    "25%", "50%", "75%", "97.5%"))
```

	Mean	Sd	2.5%	25%	50%	75%	97.5%
θ_1	11.438757	8.019370	-1.983898	6.096808	10.491578	15.789374	30.58532
θ_2	7.943405	6.299367	-5.002785	4.069199	7.965839	11.797388	20.45032
θ_3	6.486542	7.342489	-9.890623	2.261254	7.022100	11.221379	19.88905
θ_4	7.644230	6.325665	-5.249080	3.754763	7.763069	11.544482	20.59311
θ_5	5.470804	6.389392	-8.180697	1.438316	5.865742	9.754035	17.09239
θ_6	6.292634	6.556985	-7.852717	2.477329	6.785472	10.459572	18.28349
θ_7	10.722354	6.844581	-1.343582	6.212167	10.053642	14.635914	25.73749
θ_8	8.554159	7.618787	-6.292015	3.987489	8.456605	12.755109	25.20105

```
draw_expected_theta = function(dat, sigma_sqr_dat, sample_size) {
    tau_plot = seq(0.01, 30, length.out = 30)
    expected_theta_given_tau = c()
    for (tau in tau_plot) {
        mu_given_this_tau = replicate(sample_size, sample_mar_mu(tau,
            sigma_sqr_dat, dat))
        theta_sample = lapply(as.list(1:length(dat)), function(x) {
            sample_theta_j_post(mu_given_this_tau, tau, dat[x],
                sigma_sqr_dat[x])
        })
        expected_theta = unlist(lapply(theta_sample, mean))
        expected_theta_given_tau = rbind(expected_theta_given_tau,
            expected_theta)
    }
    return(list(tau = tau_plot, theta = expected_theta_given_tau))
}

result = draw_expected_theta(dat, sigma_sqr_dat, sample_size)
expected_theta = result$theta
tau_plot_grid = result$tau

par(mfrow = c(1, 1))
col_palette = rainbow(length(dat))
```

```
plot(expected_theta[, 1] ~ tau_plot_grid, type = "l", col = col_palette[1],
    ylim = c(-5, 30), xlab = "tau", ylab = "Expected theta_j given tau")
for (i in 2:length(dat)) {
    lines(expected_theta[, i] ~ tau_plot_grid, col = col_palette[i])
}
```



2. Gibbs Sampler

```
gibbs_sampler_for_sat = function(dat, sigma_sqr_dat, num_iters) {
    tau_cur = 1
    theta_cur = rep(0, length(dat))
    mu_cur = 0
    J = length(dat)
    output = c()

    for (i in 1:num_iters) {
        # First sample theta vec
        tau_cur_sqr = tau_cur^2
        theta_mean = (dat * tau_cur_sqr + mu_cur * sigma_sqr_dat)/(tau_cur_sqr +
            sigma_sqr_dat)
        theta_var = tau_cur_sqr * sigma_sqr_dat/(tau_cur_sqr +
            sigma_sqr_dat)
        theta_cur = rnorm(J, theta_mean, sd = sqrt(theta_var))

        mu_cur = rnorm(1, mean(theta_cur), sd = sqrt(tau_cur_sqr/J))

        tau_cur = sqrt(1/rgamma(1, shape = J/2 - 1, rate = sum((theta_cur -
            mu_cur)^2)/2))
        output = rbind(output, c(theta_cur, mu_cur, tau_cur))
    }
```

```
    colnames(output) = c(paste("theta_", 1:J, sep = ""), "mu",
        "tau")
    return(output)
}

num_iters = 8000
output = gibbs_sampler_for_sat(dat, sigma_sqr_dat, num_iters)
output_thin = output[seq(2000, num_iters, by = 2), ]
summ_table = matrix(unlist(apply(output_thin, 2, sum_stats)),
    nrow = 10, byrow = T)
row.names(summ_table) = c("$\\theta_1$", "$\\theta_2$", "$\\theta_3$",
    "$\\theta_4$", "$\\theta_5$", "$\\theta_6$", "$\\theta_7$",
    "$\\theta_8$", "$\\mu$", "$\\tau$")
knitr::kable(summ_table, col.names = c("Mean", "Sd", "2.5%",
    "25%", "50%", "75%", "97.5%"))
```

	Mean	Sd	2.5%	25%	50%	75%	97.5%
θ_1	15.931259	10.844161	-2.437327	8.6150298	14.510355	22.451348	40.21549
θ_2	8.182937	7.731979	-7.009815	3.1937939	8.192338	13.086906	23.42354
θ_3	4.796102	10.225556	-16.907985	-1.1140397	5.647899	11.255158	24.05419
θ_4	7.548272	8.107969	-9.480930	2.4940063	7.838875	12.792257	23.40218
θ_5	3.540214	8.097876	-13.412926	-1.7182571	4.033068	9.136796	18.07771
θ_6	4.768077	8.634252	-13.215810	-0.4084582	5.092643	10.502660	20.93819
θ_7	12.848920	8.439032	-2.222824	7.1776146	12.158660	18.221869	30.91890
θ_8	9.680618	10.828756	-10.910562	2.9363475	9.181049	15.815688	33.08785
μ	8.578726	6.943028	-4.459981	4.3479691	8.516948	12.498606	22.60049
τ	12.226027	8.600324	2.188283	6.5118966	10.309287	15.445225	32.94718

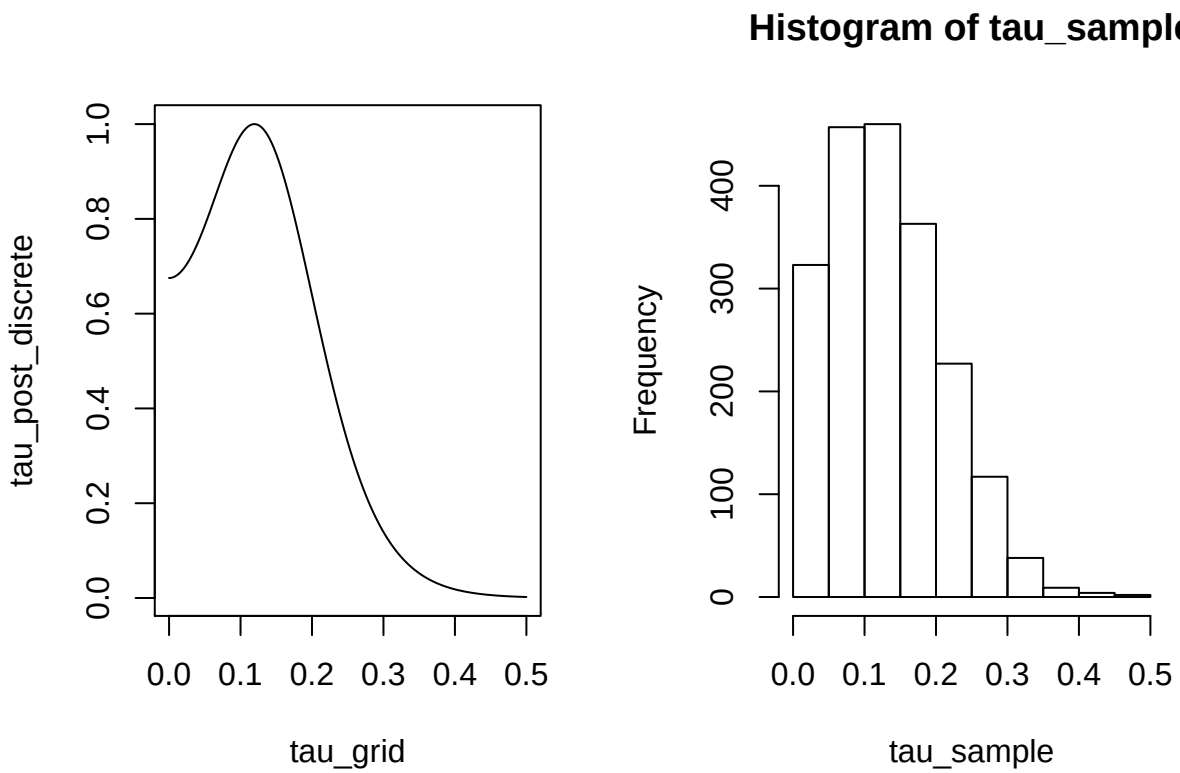
Meta analysis: 5.15

Data: y_j is the emperical log odds ratio and σ_j is the sd of group j. Data is obtained from table 5.4. Model: Same model as the SAT example.

(a)

```
dat = c(0.028, -0.741, -0.541, -0.246, 0.069, -0.584, -0.512,
    -0.079, -0.424, -0.335, -0.213, -0.039, -0.593, 0.282, -0.321,
    -0.135, 0.141, 0.322, 0.444, -0.218, -0.591, -0.608)

sigma_dat = c(0.85, 0.483, 0.565, 0.138, 0.281, 0.676, 0.139,
    0.204, 0.274, 0.117, 0.195, 0.229, 0.425, 0.205, 0.298, 0.261,
    0.364, 0.553, 0.717, 0.26, 0.257, 0.272)

sigma_sqr_dat = sigma_dat^2

tau_opt_result = optim(1, function(x) {
    tau_log_posterior(x, dat, sigma_sqr_dat)
}, method = "Brent", lower = 0, upper = 300, hessian = T, control = list(fnscale = -1))
tau_post_mode = tau_opt_result$value
tau_post_var = 1/(-tau_opt_result$hessian)

tau_grid = seq(1e-05, 0.5, length.out = 1000)
tau_grid_log_den = sapply(tau_grid, function(x) {
```

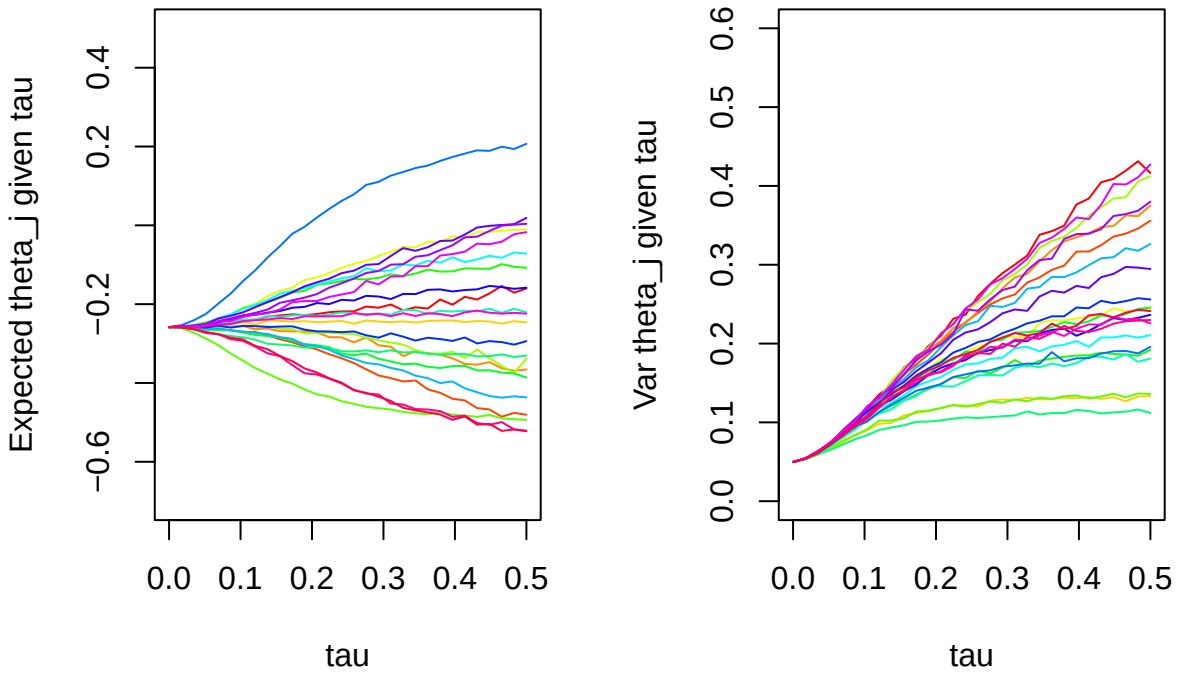
```
    tau_log_posterior(x, dat, sigma_sqr_dat)
})
tau_post_discrete = exp(tau_grid_log_den - max(tau_grid_log_den))
sample_size = 2000
tau_sample = sample(tau_grid, sample_size, prob = tau_post_discrete,
    replace = T)

par(mfrow = c(1, 2))
plot(tau_post_discrete ~ tau_grid, type = "l")
hist(tau_sample)
```



(b)

```
draw_theta_mean_var = function(dat, sigma_sqr_dat, sample_size) {
    tau_plot = seq(1e-05, 0.5, length.out = 30)
    expected_theta_given_tau = c()
    var_theta_given_tau = c()
    for (tau in tau_plot) {
        mu_given_this_tau = replicate(sample_size, sample_mar_mu(tau,
            sigma_sqr_dat, dat))
        theta_sample = lapply(as.list(1:length(dat)), function(x) {
            sample_theta_j_post(mu_given_this_tau, tau, dat[x],
                sigma_sqr_dat[x])
        })
        expected_theta = unlist(lapply(theta_sample, mean))
        var_theta = unlist(lapply(theta_sample, sd))
        expected_theta_given_tau = rbind(expected_theta_given_tau,
            expected_theta)
        var_theta_given_tau = rbind(var_theta_given_tau, var_theta)
    }
```

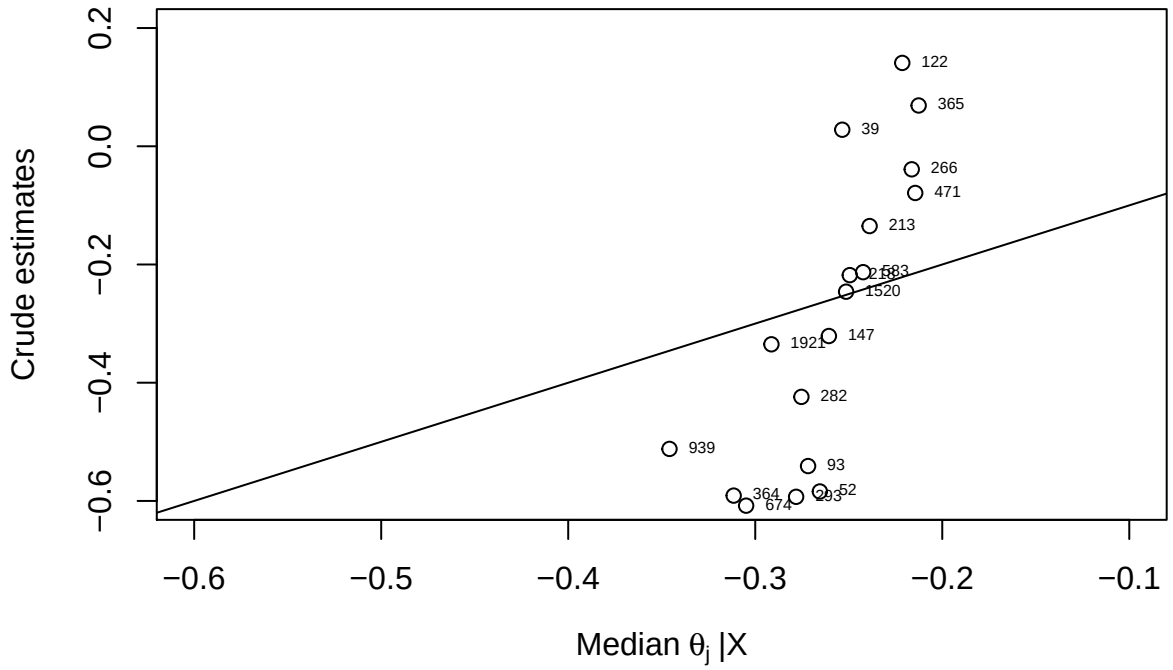
```
    return(list(tau = tau_plot, theta = expected_theta_given_tau,
        var = var_theta_given_tau))
}

result = draw_theta_mean_var(dat, sigma_sqr_dat, sample_size)
expected_theta = result$theta
tau_plot_grid = result$tau
var_theta = result$var

par(mfrow = c(1, 2))
col_palette = rainbow(length(dat))
plot(expected_theta[, 1] ~ tau_plot_grid, type = "l", col = col_palette[1],
    ylim = c(-0.7, 0.5), xlab = "tau", ylab = "Expected theta_j given tau")
for (i in 2:length(dat)) {
    lines(expected_theta[, i] ~ tau_plot_grid, col = col_palette[i])
}

plot(var_theta[, 1] ~ tau_plot_grid, type = "l", col = col_palette[1],
    ylim = c(0, 0.6), xlab = "tau", ylab = "Var theta_j given tau")
for (i in 2:length(dat)) {
    lines(var_theta[, i] ~ tau_plot_grid, col = col_palette[i])
}
```



(c)

```
## Draw samples of theta
mu_sample = sapply(tau_sample, function(x) {
    sample_mar_mu(x, sigma_sqr_dat, dat)
})

theta_sample = lapply(as.list(1:length(dat)), function(x) {
    sample_theta_j_post(mu_sample, tau_sample, dat[x], sigma_sqr_dat[x])
})
```

```
control_group_size = c(39, 116, 93, 1520, 365, 52, 939, 471,
    282, 1921, 583, 266, 293, 883, 147, 213, 122, 154, 134, 218,
    364, 674)
theta_sample_median = unlist(lapply(theta_sample, median))
plot(theta_sample_median, dat, xlim = c(-0.6, -0.1), ylim = c(-0.6,
    0.2), xlab = bquote("Median" ~ theta[j] ~ "|X"), ylab = "Crude estimates")
text(theta_sample_median, dat, labels = control_group_size, cex = 0.5,
    pos = 4)
lines(seq(-0.7, 0.7, length.out = 100), seq(-0.7, 0.7, length.out = 100),
    type = "l")
```

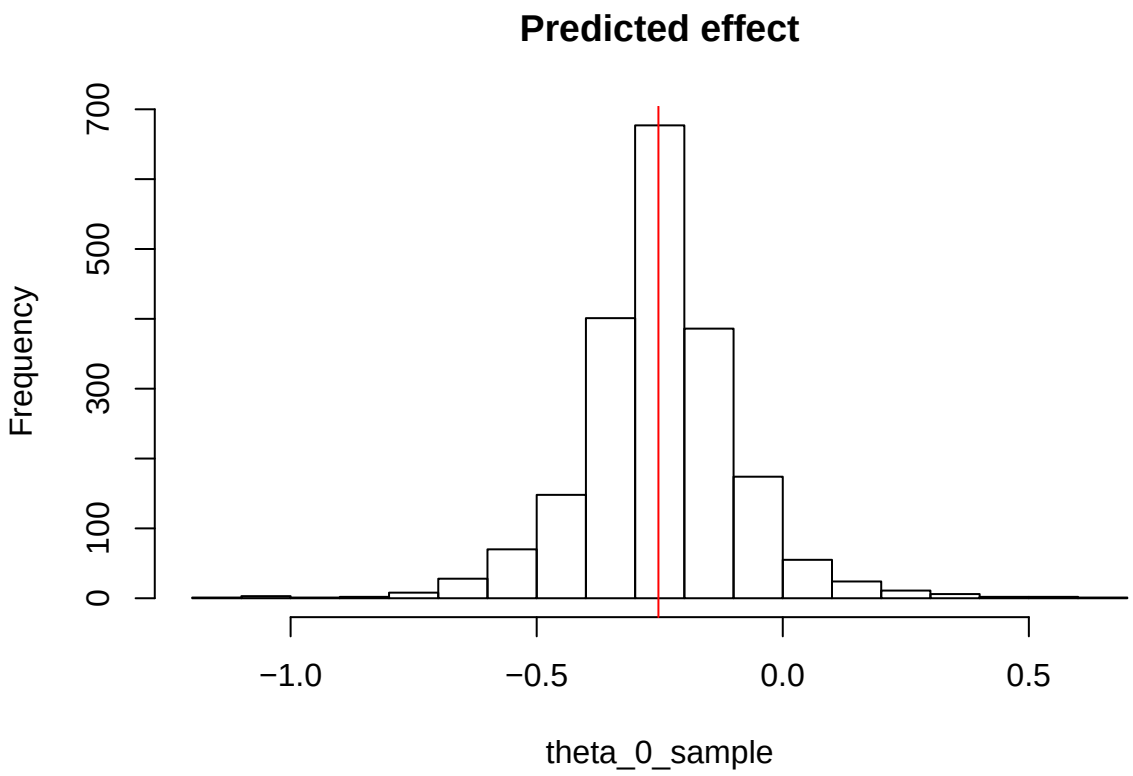


???????? How to see that the small group are more pulled to the mean?

(d) Posterior of the pretictive distribuiton of θ_0

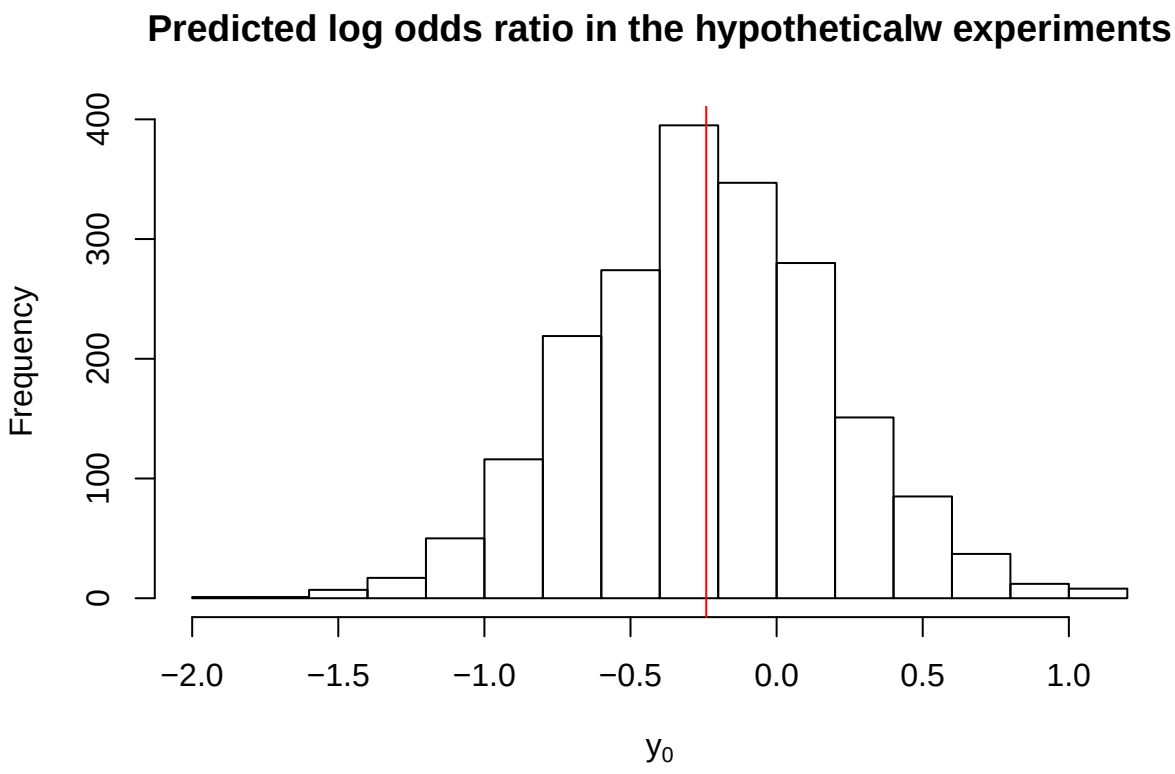
$\theta_j|\mu,\tau \sim N(\mu,\tau)$. To sample from posterior distribution of θ_0, aquired a posterior sample of (μ,τ). Draw $\theta_0^{(i)}$ from $N(\mu^{(i)},\tau^{2(i)}$.

```
mu_sample = sapply(tau_sample, function(x) {
    sample_mar_mu(x, sigma_sqr_dat, dat)
})

theta_0_sample = apply(cbind(mu_sample, tau_sample), 1, function(x) {
    rnorm(1, x[1], sd = x[2])
})
hist(theta_0_sample, breaks = 20, main = "Predicted effect")
abline(v = median(theta_0_sample), col = "red")
```



(e) Use avg variance to generate from predictive distribution We generate one new data point at each posterior sample point of θ_0. Not sure how control group size is useful here.

```
mean_sigma_sqr = mean(sigma_sqr_dat)
predictive_sample_of_new_y = sapply(theta_0_sample, function(x) {
    rnorm(1, x, sqrt(mean_sigma_sqr))
})

hist(predictive_sample_of_new_y, breaks = 20, main = "Predicted log odds ratio in the hypotheticalw exp
    xlab = bquote(y[0]))
abline(v = median(predictive_sample_of_new_y), col = "red")
```



4. Model Checking

Four poisson models: 2.13, 6.2

First, fit four poisson models according to 2.13 ### (a) Complete pooling poisson model on Fatal Accidents

$$y_i \sim Poi(\theta) \quad \theta \sim Gamma(a_1, b_1)$$

$$\theta | y_1 \cdots y_n \sim Gamma(a_1 + \sum y_i, n + b_1)$$

(b) Poisson model with Exposure parameter as passenger miles on Fatal Accidents

$$y_i \sim Poi(\theta n_j) \quad Gamma(a_2, b_2)$$

where n_j = passenger death/ death rate = passenger miles in 100MM
$\theta | y_1 \cdots y_n \sim Gamma(a_2 + \sum y_i, 10^8 \sum_{i=1}^{n} n_j + b_2)$
P45 on the parameterization of poisson model with rate (θ) and exposure n_j.

(c) Complete pooling poisson model on Passenger Death

$$x_i \sim Poi(\lambda) \quad \lambda \sim Gamma(a_3, b_3)$$

$$\lambda | x_1 \cdots x_n \sim Gamma(a_3 + \sum x_i, n + b_3)$$

(d) Poisson model with Exposure parameter as passenger miles on Passenger Death

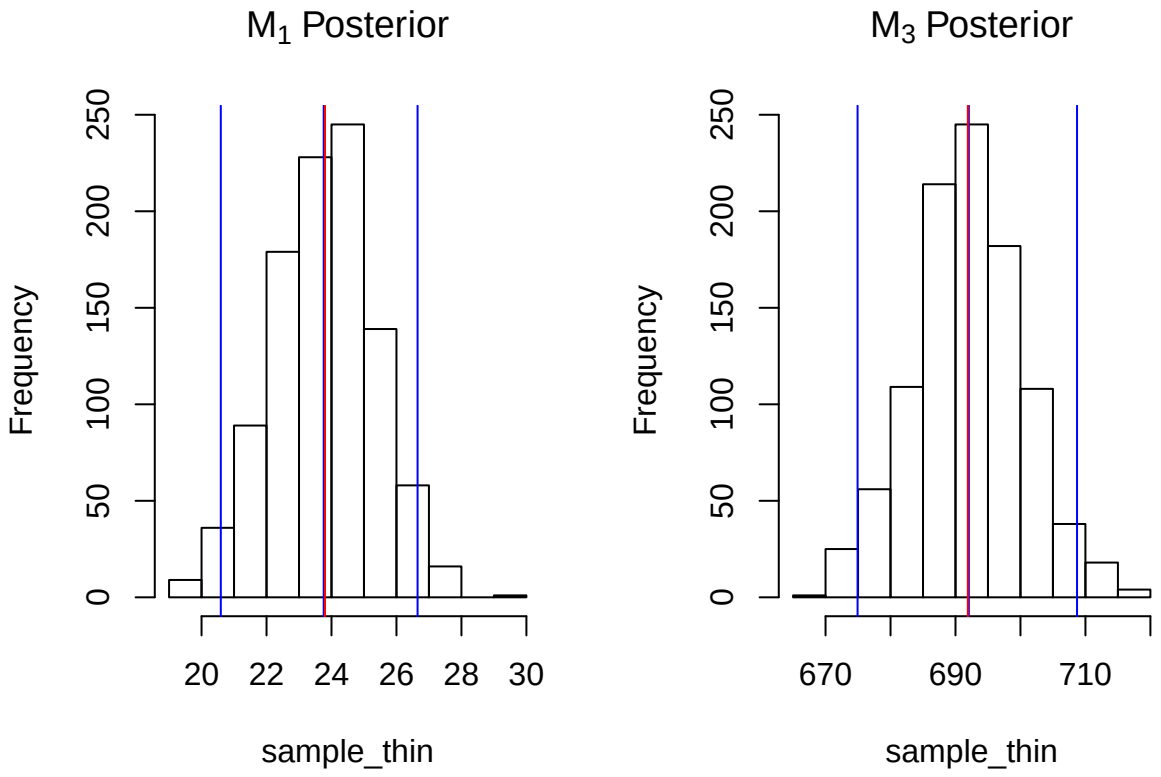
$$x_i \sim Poi(\lambda n_j) \quad Gamma(a_4, b_4)$$

where n_j = passenger death/ death rate = passenger miles in 100MM

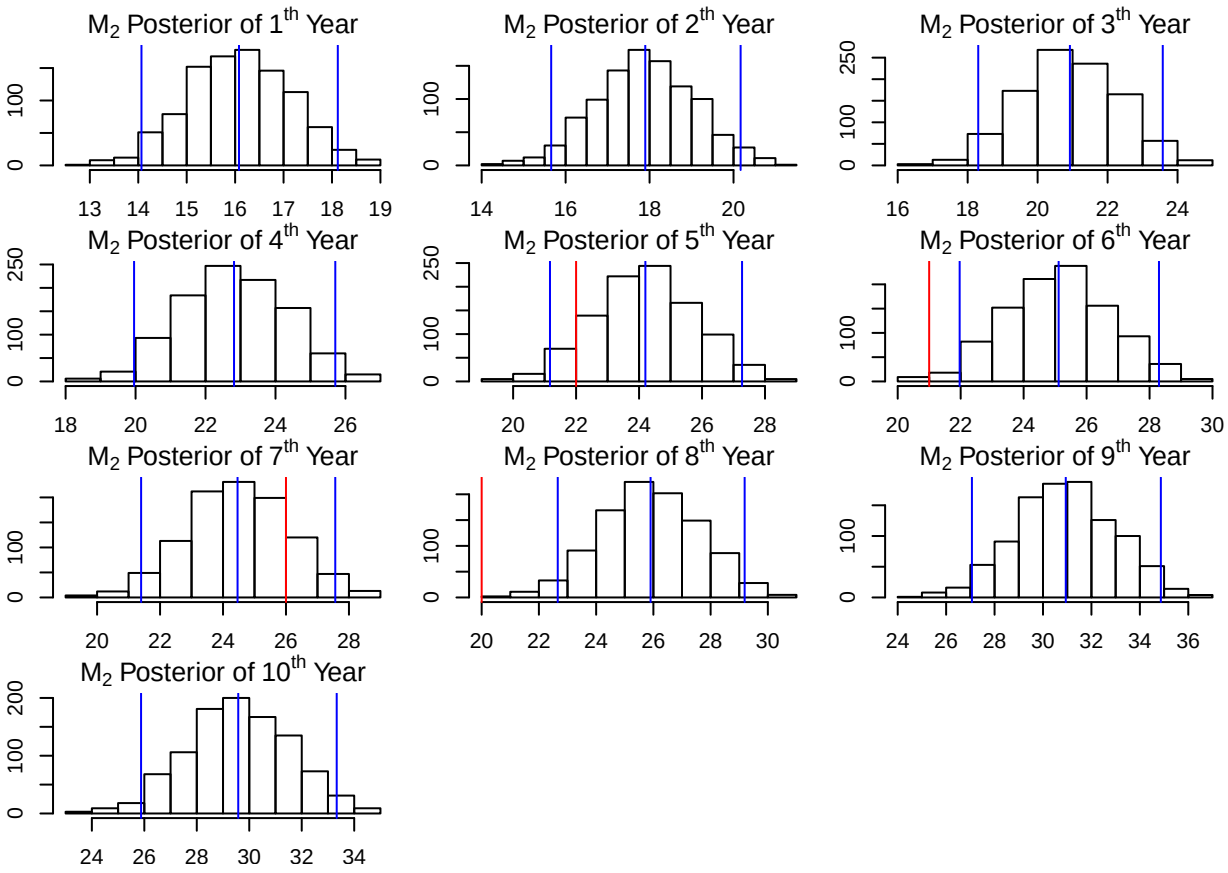
$\lambda | x_1 \cdots x_n \sim Gamma(a_4 + \sum x_i, 10^8 \sum_{i=1}^{n} n_j + b_4)$

```
fatal_dat = c(24, 25, 31, 31, 22, 21, 26, 20, 16, 22)

death_dat = c(734, 516, 754, 877, 814, 362, 764, 809, 223, 1066)

death_rate = c(0.19, 0.12, 0.15, 0.16, 0.14, 0.06, 0.13, 0.13,
    0.03, 0.15)

pass_miles = death_dat * 1e+08/death_rate

a_prior = 0.001
b_prior = 0.001
n = length(fatal_dat)

sum_fatal_dat = sum(fatal_dat)

# Posterior updates
m_1_shape = a_prior + sum_fatal_dat
m_1_rate = b_prior + n

m_2_shape = a_prior + sum_fatal_dat
m_2_rate = b_prior + sum(pass_miles)

sum_death_dat = sum(death_dat)

m_3_shape = a_prior + sum_death_dat
m_3_rate = b_prior + n

m_4_shape = a_prior + sum_death_dat
m_4_rate = b_prior + sum(pass_miles)

# Posterior samples
sample_size = 1000
m_1_post_sample = rgamma(sample_size, m_1_shape, m_1_rate)
m_2_post_sample = rgamma(sample_size, m_2_shape, m_2_rate)

m_3_post_sample = rgamma(sample_size, m_3_shape, m_3_rate)
m_4_post_sample = rgamma(sample_size, m_4_shape, m_4_rate)


plot_posterior_hist_ci_mean = function(this_title_name, sample_thin,
    true_value = NULL) {
    mean = mean(sample_thin)
    ci = quantile(sample_thin, c(0.025, 0.975))
    hist(sample_thin, main = this_title_name)
    abline(v = mean, col = "blue")
    abline(v = ci[1], col = "blue")
    abline(v = ci[2], col = "blue")
```

```
    if (!is.null(true_value)) {
        abline(v = true_value, col = "red")
    }

}
par(mfrow = c(1, 2))
plot_posterior_hist_ci_mean(bquote(M[1] ~ "Posterior"), m_1_post_sample,
    true_value = mean(fatal_dat))
plot_posterior_hist_ci_mean(bquote(M[3] ~ "Posterior"), m_3_post_sample,
    true_value = mean(death_dat))
```

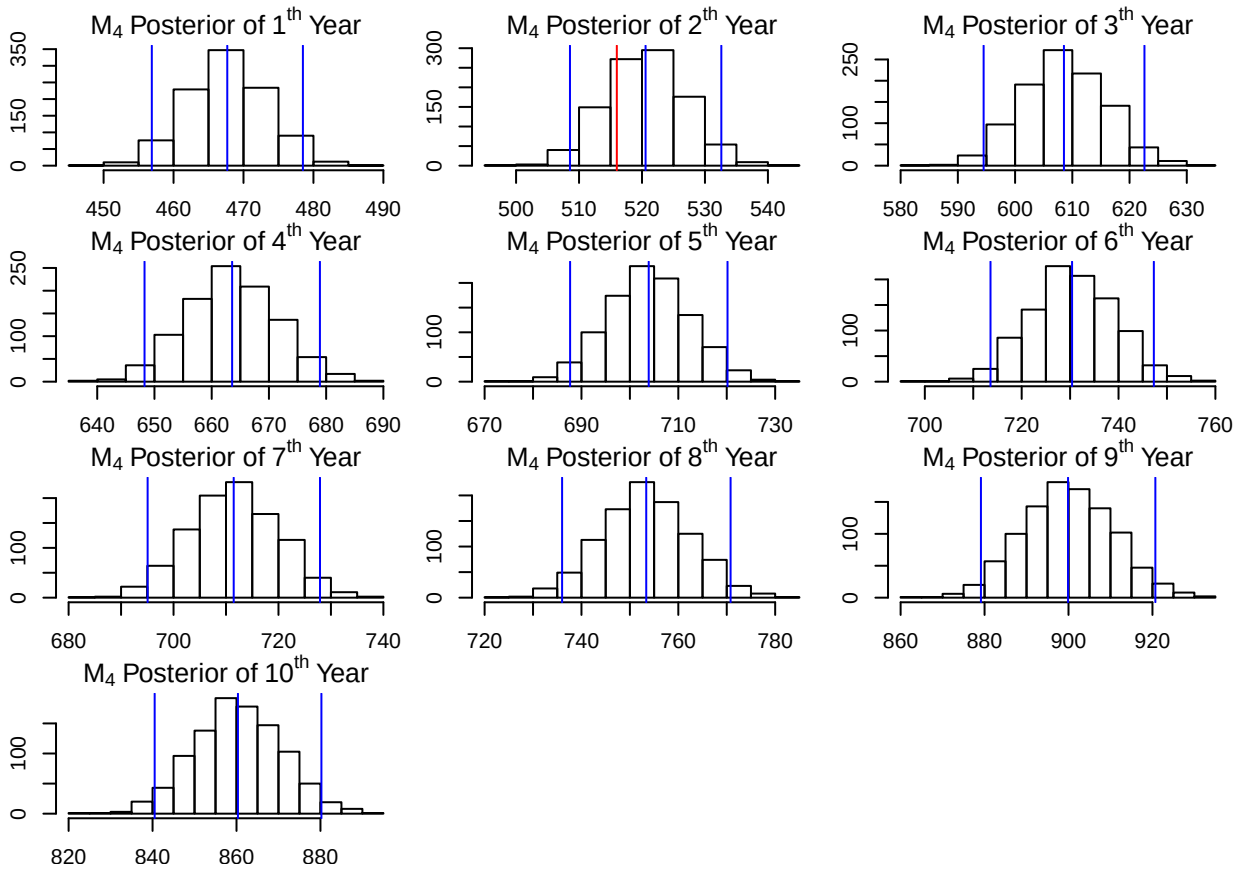



```
## How to generate a summary plots?
par(mfrow = c(4, 3), mar = c(1.8, 2, 1.8, 1))
for (i in 1:length(pass_miles)) {
    post_sample = m_2_post_sample * pass_miles[i]
    plot_posterior_hist_ci_mean(bquote(M[2] ~ "Posterior of" ~
        .(i)^th ~ "Year"), post_sample, true_value = fatal_dat[i])
}

par(mfrow = c(4, 3), mar = c(1.8, 2, 1.8, 1))
```




```
for (i in 1:length(pass_miles)) {
    post_sample = m_4_post_sample * pass_miles[i]
    plot_posterior_hist_ci_mean(bquote(M[4] ~ "Posterior of" ~
        .(i)^th ~ "Year"), post_sample, true_value = death_dat[i])
}

# Posterior predictive sample
generate_predictive_sample = function(post_sample, n, exposure,
    add_exposure = F) {
    if (add_exposure) {
        replica_list = lapply(as.list(post_sample), function(x) {
            rpois(n, x * exposure)
        })
    } else {
        replica_list = lapply(as.list(post_sample), function(x) {
            rpois(n, x)
        })
    }

    replica_mat = matrix(unlist(replica_list), byrow = T, ncol = n)
    return(replica_mat)
}

m_1_replica_mat = generate_predictive_sample(m_1_post_sample,
    n, pass_miles, add_exposure = F)
m_2_replica_mat = generate_predictive_sample(m_2_post_sample,
```

```
    n, pass_miles, add_exposure = T)

m_3_replica_mat = generate_predictive_sample(m_3_post_sample,
    n, pass_miles, add_exposure = F)
m_4_replica_mat = generate_predictive_sample(m_4_post_sample,
    n, pass_miles, add_exposure = T)
```

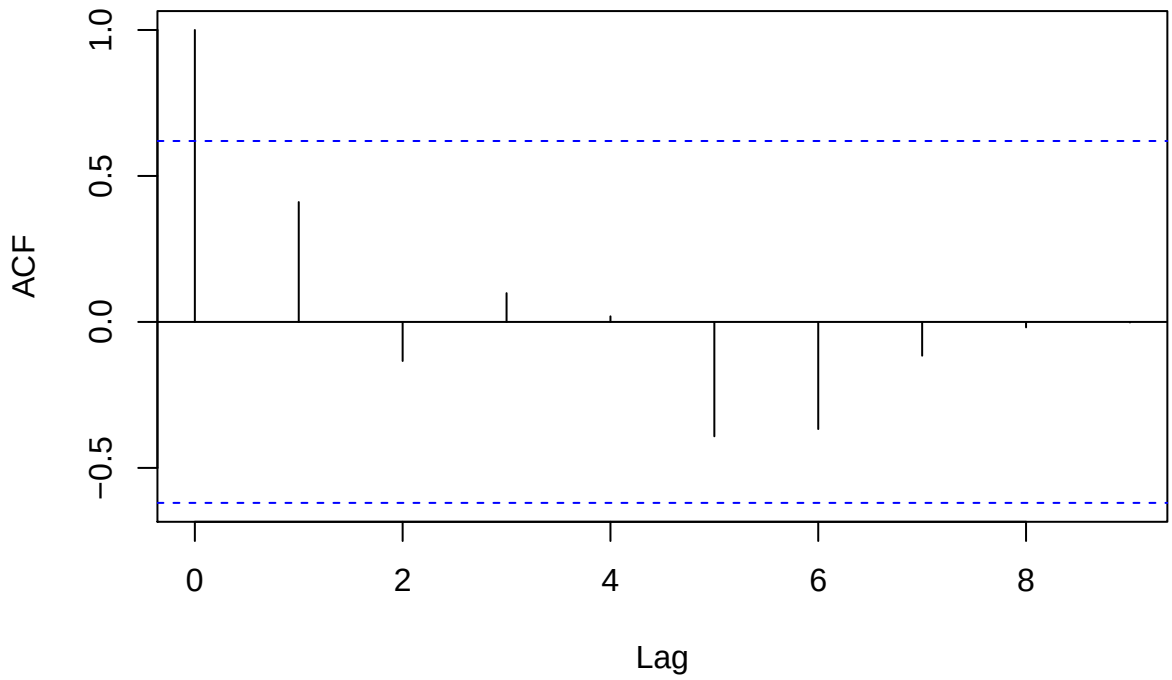


Model checking

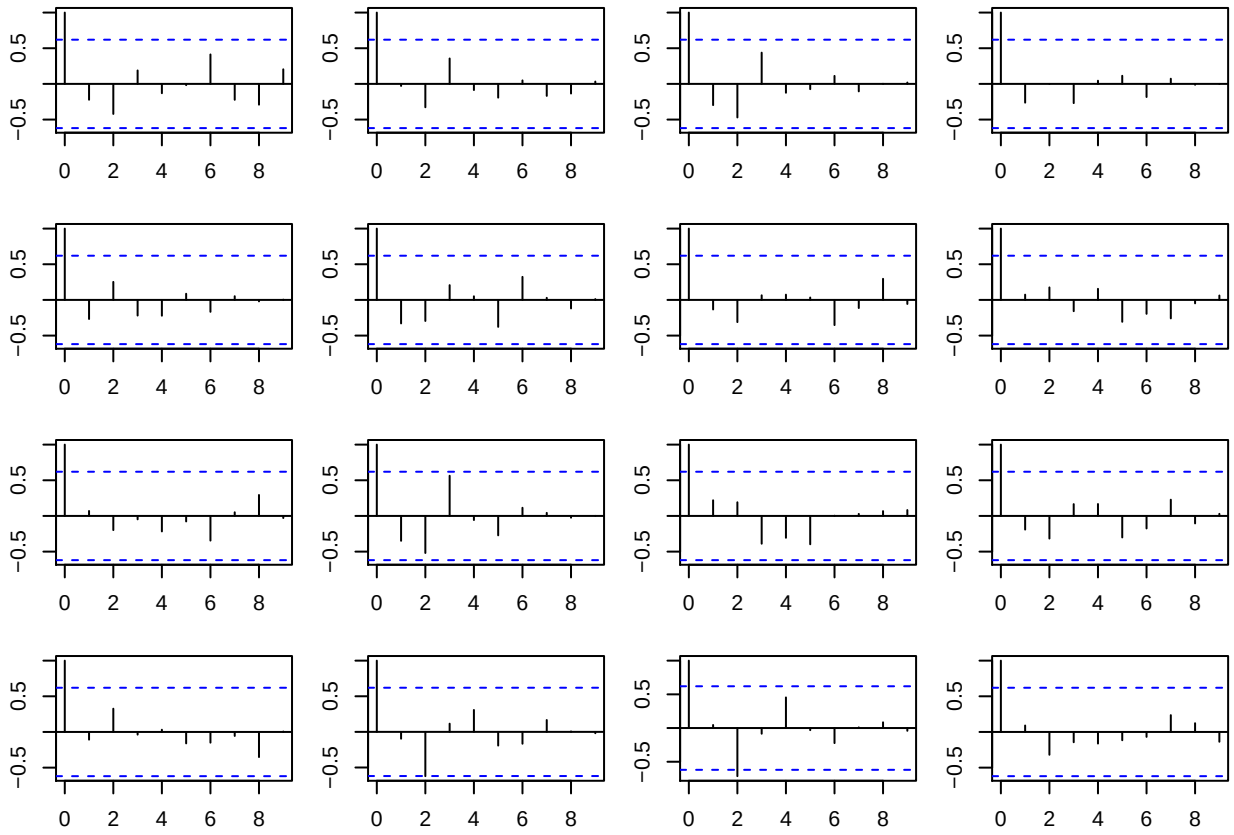
1. Independence Poisson distribuitons a. Plot the acf of replicas and original data, place them side by side. b. Define test statistics as the correlation coefficient between all y_t and y_{t-1}. Check average correlation coefficient and pointwise correlation coefficient.

2. Trend overtime a. Use AR(1) to fit original data and replicas. b. Calculate the p value.

```
plot(acf(fatal_dat, plot = F), main = "Auto-correlation of Fatal Accidents")
```

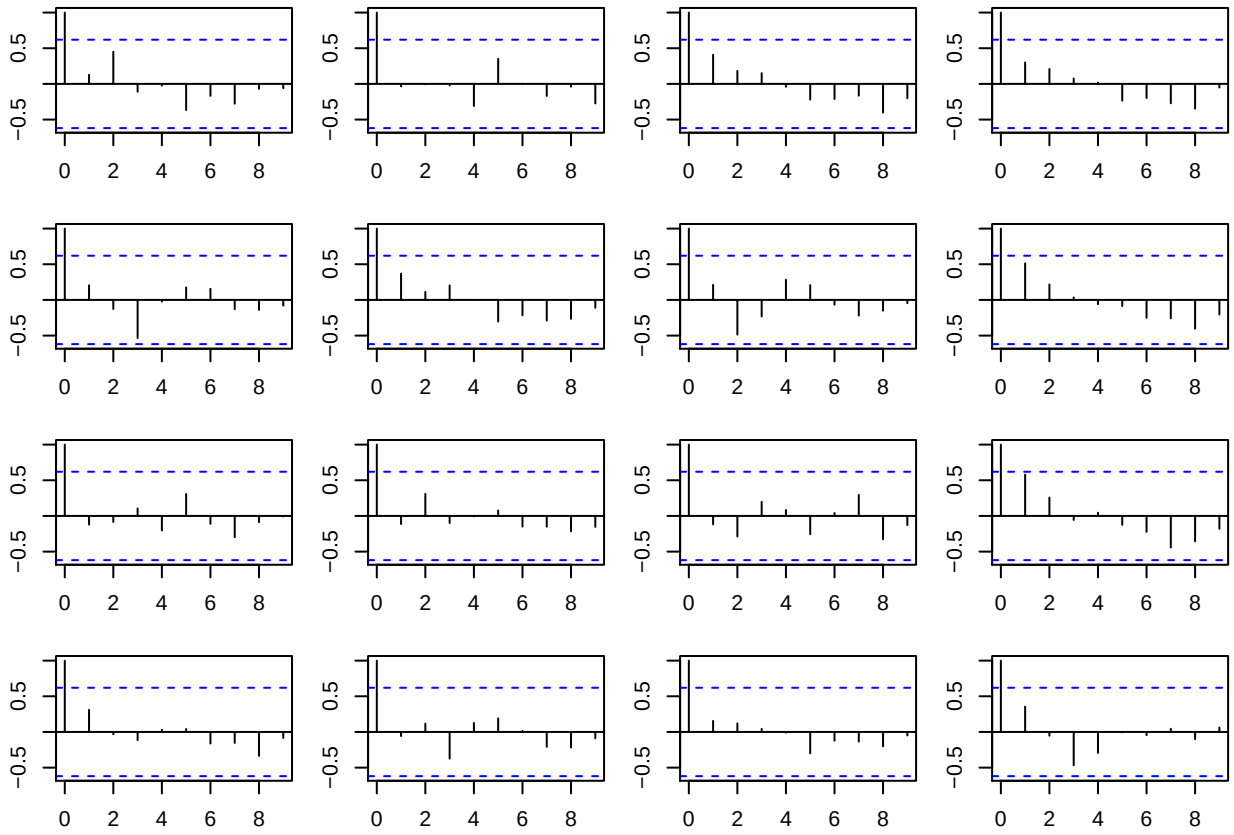



```
plot_acf = function(sampled_replica) {
    par(mfrow = c(4, 4), mar = c(1.8, 2, 1.8, 1))
    apply(sampled_replica, 1, function(x) {
        plot(acf(x, plot = F))
    })
}

plot_acf(m_1_replica_mat[sample(1:sample_size, 16), ])
```




```
## NULL
```

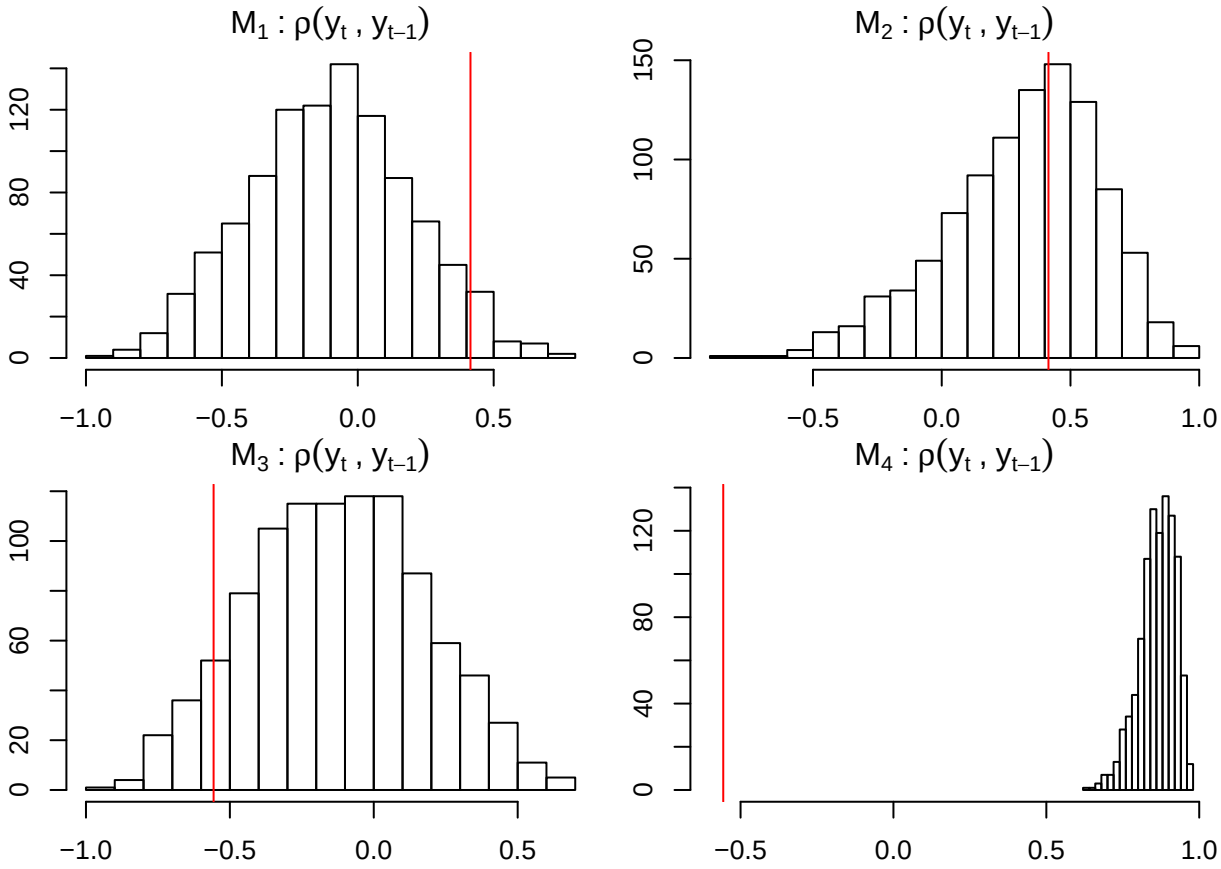
```
plot_acf(m_2_replica_mat[sample(1:sample_size, 16), ])
```




```
## NULL
```

```
calculate_lag_1_correlation = function(dat_seq) {
    y_t = dat_seq[2:length(dat_seq)]
    y_t_lag_1 = dat_seq[1:length(y_t)]
    return(cor(y_t, y_t_lag_1))
}

replica_list = list(m1 = m_1_replica_mat, m2 = m_2_replica_mat,
    m3 = m_3_replica_mat, m4 = m_4_replica_mat)

cor_list = lapply(replica_list, function(x) {
    apply(x, 1, calculate_lag_1_correlation)
})

fatal_accidents_cor = calculate_lag_1_correlation(fatal_dat)
death_cor = calculate_lag_1_correlation(death_dat)

par(mfrow = c(2, 2))
hist(cor_list[[1]], xlab = "", breaks = 20, main = bquote(M[1] ~
    ":" ~ rho(y[t] ~ "," ~ y[t - 1])))
abline(v = fatal_accidents_cor, col = "red")

hist(cor_list[[2]], xlab = "", breaks = 20, main = bquote(M[2] ~
    ":" ~ rho(y[t] ~ "," ~ y[t - 1])))
abline(v = fatal_accidents_cor, col = "red")

hist(cor_list[[3]], xlab = "", breaks = 20, main = bquote(M[3] ~
```

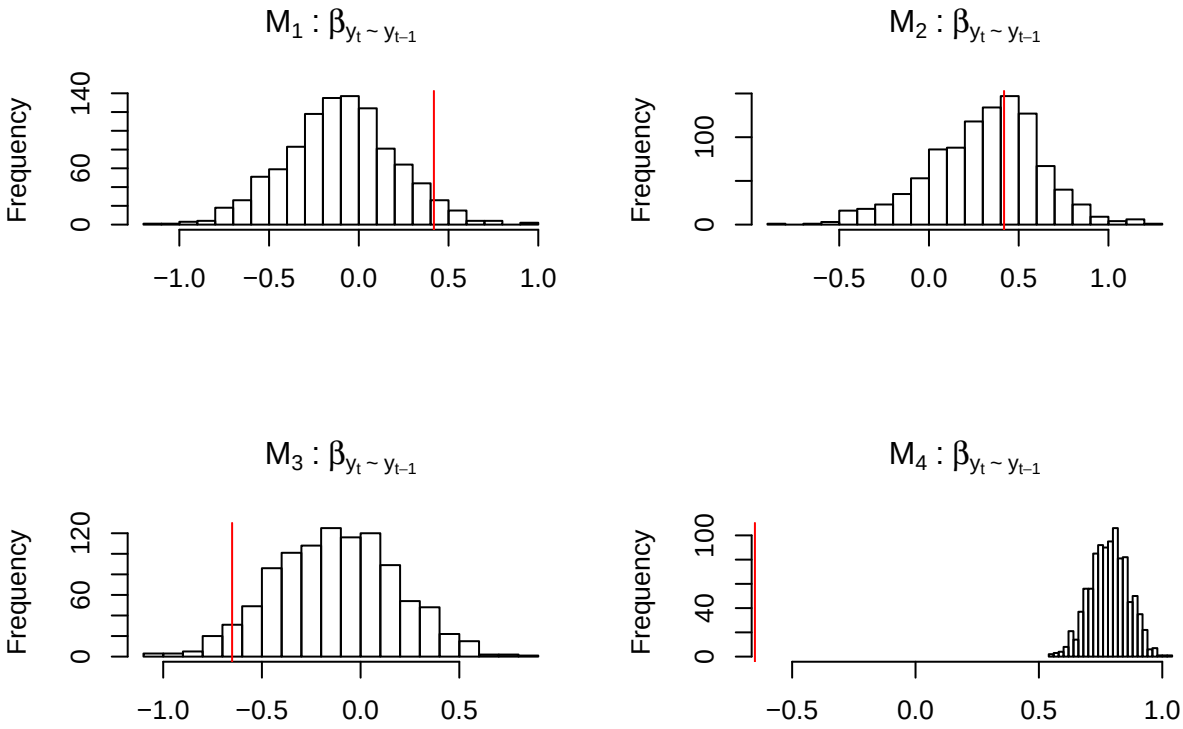
```
    ":" ~ rho(y[t] ~ "," ~ y[t - 1])))
abline(v = death_cor, col = "red")

hist(cor_list[[4]], xlab = "", breaks = 20, main = bquote(M[4] ~
    ":" ~ rho(y[t] ~ "," ~ y[t - 1])), xlim = c(-0.6, 1))
abline(v = death_cor, col = "red")
```




```
m_1_p_value = sum(cor_list[[1]] > fatal_accidents_cor)/sample_size
m_2_p_value = sum(cor_list[[2]] > fatal_accidents_cor)/sample_size
m_3_p_value = sum(cor_list[[3]] < death_cor)/sample_size
m_4_p_value = sum(cor_list[[4]] < death_cor)/sample_size
```

```
fatal_coef = ar(fatal_dat, aic = FALSE, method = "ols", order.max = 1)$ar
death_coef = ar(death_dat, aic = FALSE, method = "ols", order.max = 1)$ar

coef_list = lapply(replica_list, function(x) {
    apply(x, 1, function(row) {
        ar(row, aic = F, method = "ols", order.max = 1)$ar
    })
})

par(mfrow = c(2, 2))
hist(coef_list[[1]], xlab = "", breaks = 20, main = bquote(M[1] ~
    ":" ~ beta[y[t] ~ "~" ~ y[t - 1]]))
abline(v = fatal_coef, col = "red")

hist(coef_list[[2]], xlab = "", breaks = 20, main = bquote(M[2] ~
```

```
    ":" ~ beta[y[t] ~ "~" ~ y[t - 1]]))
abline(v = fatal_coef, col = "red")

hist(coef_list[[3]], xlab = "", breaks = 20, main = bquote(M[3] ~
    ":" ~ beta[y[t] ~ "~" ~ y[t - 1]]))
abline(v = death_coef, col = "red")

hist(coef_list[[4]], xlab = "", breaks = 20, main = bquote(M[4] ~
    ":" ~ beta[y[t] ~ "~" ~ y[t - 1]]), xlim = c(-0.6, 1))
abline(v = death_coef, col = "red")
```




```
m_1_p_value_coef = sum(sapply(coef_list[[1]], function(y) {
    y > fatal_coef
}))/sample_size
m_2_p_value_coef = sum(sapply(coef_list[[2]], function(y) {
    y > fatal_coef
}))/sample_size
m_3_p_value_coef = sum(sapply(coef_list[[3]], function(y) {
    y < death_coef
}))/sample_size
m_4_p_value_coef = sum(sapply(coef_list[[4]], function(y) {
    y < death_coef
}))/sample_size

knitr::kable(rbind(Independence = c(m_1_p_value, m_2_p_value,
    m_3_p_value, m_4_p_value), Trend = c(m_1_p_value_coef, m_2_p_value_coef,
    m_3_p_value_coef, m_4_p_value_coef)), col.names = c("$M_1$",
    "$M_2$", "$M_3$", "$M_4$"))
```

	M_1	M_2	M_3	M_4
Independence	0.042	0.414	0.081	0

	M_1	M_2	M_3	M_4
Trend	0.046	0.395	0.044	0

(C) Which is a suitable model???

Veriety of predictive reference sets:

In the example of binary outcomes on page 147, it is assumed that the number of measurements, n, is fixed in advance, and so the hypothetical replications under the binomial model are performed with n = 20. Suppose instead that the protocol for measurement is to stop once 13 zeros have appeared.

(a) Explain why the posterior distribuiton of thet parameter θ under assumed model does not change.

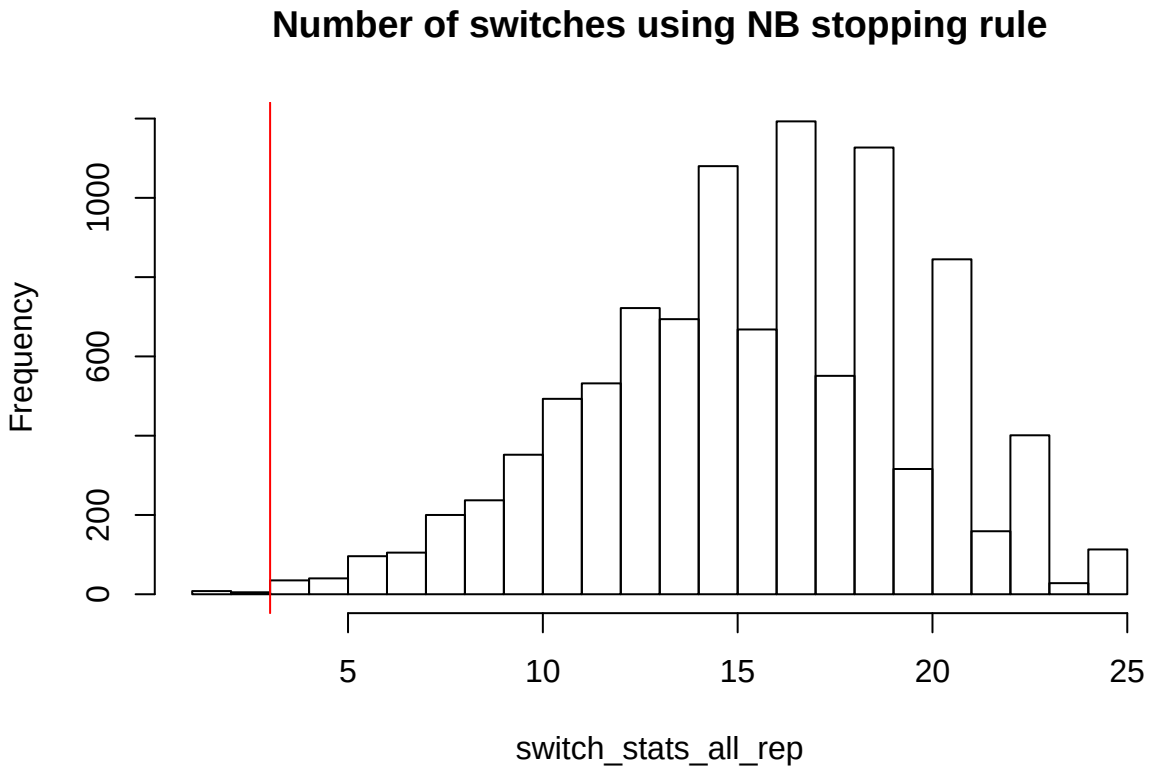
Accroding to likelihood principle, when $L_1(\theta|x) = L_2(\theta|x)$, i.e. the likelihood under one model is a proportional to that under the other model, the inference based on two likelihood should be the same. Here Negative-Binomial and Binomial have the same kernel, so their likelihood is proportional to each other, the predictive distribution based on both model should be the same.

(b) Simulate replicas of the data under the new stopping rule

```
S = 10000
theta_post = rbeta(S, 8, 14)
summary(theta_post)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07935 0.29103 0.35841 0.36373 0.42913 0.72995
```

```
number_heads = 13
sample_from_nb = function(theta, number_heads) {
    unif_draws = runif(5000)
    ber_draws = (unif_draws < theta) * 1
    cum_sum_ber = cumsum(ber_draws)
    cut_off_index = min(which(cum_sum_ber == number_heads))
    nb_seq = ber_draws[1:cut_off_index]
    return(nb_seq)
}

y_rep_list = lapply(as.list(theta_post), function(x) {
    sample_from_nb(x, number_heads)
})

get_number_of_switches = function(binary_seq) {
    y_t = binary_seq[2:length(binary_seq)]
    y_t_lag_1 = binary_seq[1:length(y_t)]
    return(sum(y_t != y_t_lag_1))
}

switch_stats_all_rep = unlist(lapply(y_rep_list, get_number_of_switches))

hist(switch_stats_all_rep, breaks = 20, main = "Number of switches using NB stopping rule")
abline(v = 3, col = "red")
```



!!! It does make a differnce !!! The number of switches is more than that of binomial replica datas in general.

5. Chapter 7

Power-transformed normal models: 7.5

(a) The problem is when $\phi \neq 0$

Let $x_i = (y_i^\phi - 1)/\phi$, $J = (\prod_{i=1}^n y_i)^{1-\phi}$.

The posterior of θ is given by:

$$p(\mu, log\sigma | x_1, \cdots, x_n) \propto \prod_{i=1^n} N(x_i|\mu, \log\sigma) p(log\sigma) \int Jp(\phi)d\phi$$

Now observe this posterior is not scale invariant. If we multiply each y_i with constant a then

$$J^* = a^n J$$

Which would cause the posterior to be multiplied by the arbituary number a^n

(b) When the prior of $p(\phi) \propto (\prod_{i=1}^n y_i)^{1/n}$ Then

$$\int J^* p(\phi)d\phi = \int \prod_{i=1}^n y_i p(\phi)d\phi = \int Jp(\phi)d\phi$$

Thus solves this problem

(c)

Integrate over μ and σ^2 we end up with:

when $\phi \neq 0$

$$p(\phi|y) \propto \prod_{i=1}^{n} y_i^{1+1/n}$$

o.w.

$$p(\phi|y) \propto e^{\sum_{i=1}^{n} y_i} \prod_{i=1}^{n} y_i^{1/n}$$

(d) ???

(e) ???

6. Data collection

8.11

8.14

(a) x = real death rate, y = total miles

$$x|x_{trunc} \sim unif(x_trunc - 0.005, x_trunc + 0.005)$$

```
x_trunc = 0.19

sample_size = 1000

x_real = runif(sample_size, x_trunc - 0.005, x_trunc + 0.005)

x_real_ci = quantile(x_real, c(0.025, 0.975))

y_ci = 734/x_real_ci * 10^8

cat(formatC(y_ci, digit = 3, format = "e"))
```

```
## 3.962e+11 3.769e+11
```

(b)

```
x_trunc_vec = c(0.12, 0.15, 0.16, 0.14, 0.06, 0.13, 0.13, 0.03,
    0.15)
x_real_list = lapply(as.list(x_trunc_vec), function(x) {
    runif(sample_size, x - 0.005, x + 0.005)
})
x_real_ci_vec = lapply(x_real_list, function(x) {
    quantile(x, c(0.025, 0.975))
})
death_vec = c(516, 754, 877, 814, 362, 764, 809, 223, 1066)
predicted_ci = lapply(as.list(c(1:length(death_vec))), function(x) {
    formatC(10^8 * death_vec[x]/x_real_ci_vec[[x]], digit = 3,
        format = "e")
})

predicted_ci
```

```
## [[1]]
##        2.5%       97.5%
## "4.475e+11" "4.140e+11"
##
## [[2]]
##        2.5%       97.5%
## "5.191e+11" "4.873e+11"
##
## [[3]]
##        2.5%       97.5%
## "5.649e+11" "5.322e+11"
##
## [[4]]
##        2.5%       97.5%
## "6.017e+11" "5.622e+11"
##
## [[5]]
##        2.5%       97.5%
## "6.554e+11" "5.593e+11"
##
## [[6]]
##        2.5%       97.5%
## "6.101e+11" "5.668e+11"
##
## [[7]]
##        2.5%       97.5%
## "6.461e+11" "6.000e+11"
##
## [[8]]
##        2.5%       97.5%
## "8.800e+11" "6.420e+11"
##
## [[9]]
##        2.5%       97.5%
## "7.341e+11" "6.895e+11"
```

(c) Assume normal, then same as 3.5