ISYE 6420: Bayesian Statistics **Spring 2020**

Final Exam

Instructor: Roshan Vengazhiyil, Brani Vidakovic *Name:* Nick Korbit, *gtID:* 903263968

Problem 1

Let y be the variable we would like to predict – probability of a person going to the beach. We model y as

$$y_i \sim \mathcal{B}er\left(p_i\right)$$
$$logit\left(p_i\right) = \log\frac{p_i}{1-p_i} = \beta_0 + \sum_{i=1}^{5}\beta_i x_i$$

We have 5 covariates – "Midterm", "Finances", "FriendsGo", "Forecast" and "Gender". For each of the covariate coefficients we set a flat normal prior with $\mu = 0$ and $precision = 0.5$. So that the OpenBUGS model is

```
# Training
for (i in 1:n) {
logit(p[i]) <- b0 + b1*Midterm[i] + b2*Finances[i] + b3*FriendsGo[i]
        + b4*Forecast[i] + b5*Gender[i]
Beach[i] ~ dbern(p[i])
}

# Priors
b0 ~ dnorm(0.0, 0.5)
b1 ~ dnorm(0.0, 0.5)
b2 ~ dnorm(0.0, 0.5)
b3 ~ dnorm(0.0, 0.5)
b4 ~ dnorm(0.0, 0.5)
b5 ~ dnorm(0.0, 0.5)
```

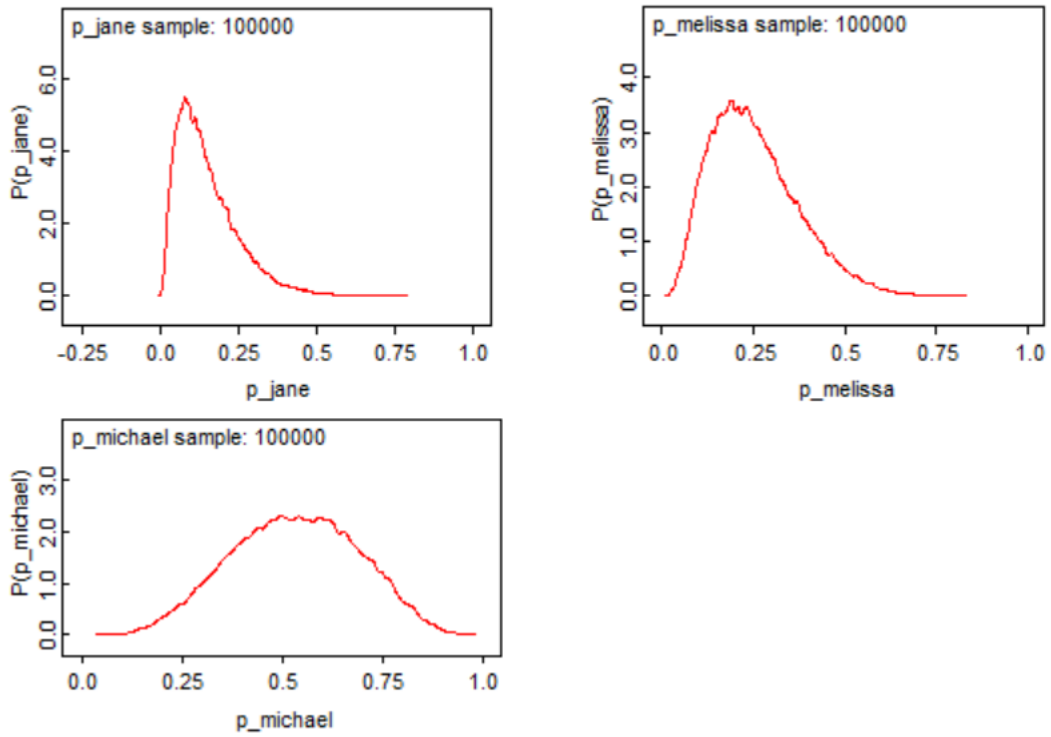
Having trained the model, we would like to estimate probability distributions for Jane, Michael and Melissa. Knowing the parameters for all persons, we specify:

```
jane=c(1,1,0,0,1), michael=c(0,0,1,1,0), melissa=c(1,1,0,1,1)
```

And then we proceed to inference:

```
logit(p_jane) <- b0 + b1*jane[1] + b2*jane[2] + b3*jane[3] + b4*jane[4] + b5*jane[5]
logit(p_michael) <- b0 + b1*michael[1] + b2*michael[2] + b3*michael[3]
        + b4*michael[4] + b5*michael[5]
logit(p_melissa) <- b0 + b1*melissa[1] + b2*melissa[2] + b3*melissa[3]
        + b4*melissa[4] + b5*melissa[5]
```

Let's now run an OpenBUGS simulation. We start with burning the first 10000 observation and update the model with the next 100000 samples. First, we plot densities for the new points:



We notice that Jane has the lowest (and most dense) probability of going to the beach, with Michael having the widest probability density. Let's now investigate the stats:

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
p_jane	0.1505	0.09699	5.0E-4	0.02753	0.1281	0.3957	10001	100000
p_melissa	0.2517	0.1182	6.552E-4	0.07149	0.2351	0.5218	10001	100000
p_michael	0.5319	0.1563	8.496E-4	0.2293	0.534	0.8215	10001	100000

Comparing with the results from the first assignment we get:

	OpenBUGS mean	OpenBUGS CS95 low	OpenBUGS CS95 high	Naive Bayes (HW1)
Jane	0.15	0.03	0.4	0.17
Melissa	0.25	0.07	0.52	0.28
Michael	0.53	0.23	0.82	0.4

The means of our OpenBUGS simulation are close to the original results from Homework 1. All original results are within 95% of OpenBUGS credible sets. The worst match is for Michael, but that's also not surprising provided that Michael's probability density is the widest.

Note that we should not expect one-to-one match between OpenBUGS simulation and Naive Bayes from HW1 since the model specifications are different.

Note: the full OpenBUGS code is available at *jmmatbeach.odc* in the attached archive.

Problem 2

Let y be the variable we would like to predict – the LC50 value. We model y as

$$y_i \sim \mathcal{N}(\mu, \tau)$$
$$\mu_i = \beta_0 + \sum_{i=1}^{8} \beta_i x_i$$

We have 8 covariates – TPSA(Tot) (Molecular properties), SAacc (Molecular properties), H-050 (Atom-centred fragments), MLOGP (Molecular properties), RDCHI (Connectivity indices), GATS1p (2D autocorrelations), nN (Constitutional indices) and C-040 (Atom-centred fragments). For each of the covariate coefficients we set a normal prior with $\mu = 0$ and $precision = 0.001$. We also set a prior for τ as a gamma distribution with parameters $(0.001, 0.001)$. So that the OpenBUGS model is

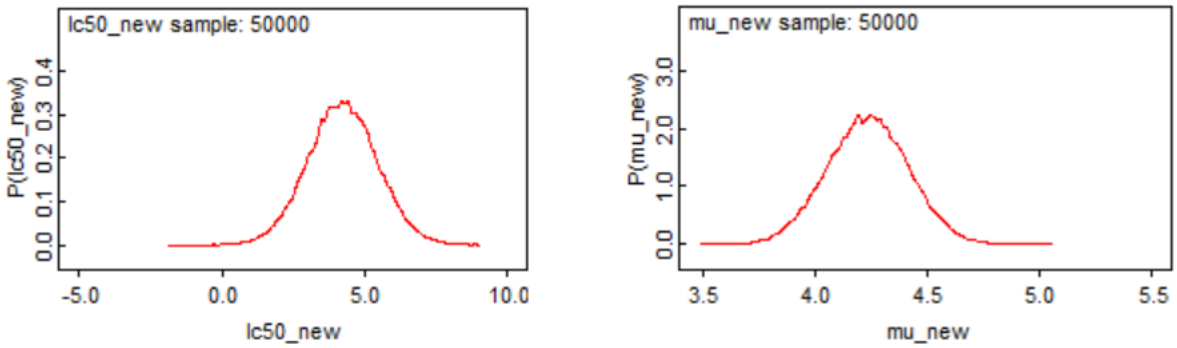
```
# Training
for (i in 1:n) {
mu[i] <- b[1] + b[2]*tpsa[i] + b[3]*saacc[i] + b[4]*h050[i] + b[5]*mlogp[i]
        + b[6]*rdchi[i] + b[7]*gats1p[i] + b[8]*nn[i] + b[9]*c040[i]
lc50[i] ~ dnorm(mu[i], tau)
}

# Priors
tau ~ dgamma(0.001, 0.001)
for (j in 1:m) {
        b[j] ~ dnorm(0, 0.001)
}
```

Next, we specify calculations for the new data, both μ_{new} and y_{new}:

```
# Inference
mu_new <- b[1] + b[2]*x[1] + b[3]*x[2] + b[4]*x[3] + b[5]*x[4]
        + b[6]*x[5] + b[7]*x[6] + b[8]*x[7] + b[9]*x[8]
lc50_new ~ dnorm(mu_new, tau)
```

Let's now run an OpenBUGS simulation. We start with burning the first 5000 observation and update the model with the next 50000 samples.

a) First, we plot densities for the new points:



We notice that the density for prediction for a new observation is much more wider than the density for the mean response. We are much more uncertain about the actual prediction than μ. Let's look at the stats:

lc50_new	4.235	1.219	0.006902	1.854	4.231	6.635	5001	50000
mu_new	4.237	0.1815	0.004039	3.882	4.238	4.592	5001	50000

The means of μ_{new} and y_{new} are almost the same, however, the credible sets are different. For μ_{new} we have a $(3.9, 4.6)$ 95% set and for y_{new} – $(1.9, 6.6)$.

b) Let's now analyse the coefficients $\beta_0, .., \beta_8$:

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
b[1]	2.697	0.2414	0.005926	2.23	2.696	3.178	5001	50000
b[2]	0.02719	0.002691	6.31E-5	0.02197	0.02719	0.03248	5001	50000
b[3]	-0.01509	0.002089	4.419E-5	-0.01912	-0.01513	-0.01097	5001	50000
b[4]	0.04128	0.05982	9.66E-4	-0.07653	0.04121	0.1574	5001	50000
b[5]	0.4464	0.06232	0.001825	0.3244	0.4464	0.5692	5001	50000
b[6]	0.5143	0.1307	0.00432	0.256	0.5138	0.7769	5001	50000
b[7]	-0.5708	0.1549	0.003473	-0.8733	-0.572	-0.2659	5001	50000
b[8]	-0.2244	0.04846	5.236E-4	-0.3198	-0.2245	-0.1289	5001	50000
b[9]	0.003639	0.07808	7.78E-4	-0.15	0.00409	0.1557	5001	50000

We notice that β_3 (b[4]) and β_8 (b[9]) contain 0 in their respective 95% credible sets. So, we can consider these predictors (H-050 and C-040) insignificant and exclude them from the model.

c) Let's investigate the overall quality of the regression by look at R^2 and R^2_{adj} metrics:

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BR2	0.4841	0.03166	1.457E-4	0.4186	0.4853	0.5427	5001	50000
BR2adj	0.4765	0.03213	1.479E-4	0.4099	0.4776	0.5358	5001	50000

We see that the overall model quality is moderate – only 0.48. In order to improve our model we could do feature engineering or collect more data. On the feature engineering side we can exclude the predictors we find insignificant as well as modify the existing features (like we did it in Assignment 6 with time features). On the other hand, we could collect more data points for the same features or include other factors.

Note: the full OpenBUGS code is available at *DaphniaMagna.odc* in the attached archive.

Problem 3

a) In our meta-analysis we would like to study the effect of using amantadine to prevent influenza. As a basis for the meta-analysis we have the results of 8 studies of amantadine conducted from 1970 to 1989.

Borrowing from the Blocker OpenBUGS example, we assume that in a random effects meta-analysis the true effect (on a log-odds scale) δ_i in a trial i is drawn from some population distribution. Let r_i^c denote the number of cases when the subject got influenza in the control group in trial i, and r_i^t denote number of influenza cases under active amantadine treatment in trial i. Our model is then:

$$
\begin{aligned}
r_i^c &\sim \mathcal{B}in(p_i^c, n_i^c) \\
r_i^t &\sim \mathcal{B}in(p_i^t, n_i^t) \\
logit(p_i^c) &= \mu_i \\
logit(p_i^t) &= \mu_i + \delta_i \\
\delta_i &\sim \mathcal{N}(d, \tau)
\end{aligned}
$$

We also specify the model in OpenBUGS terms:

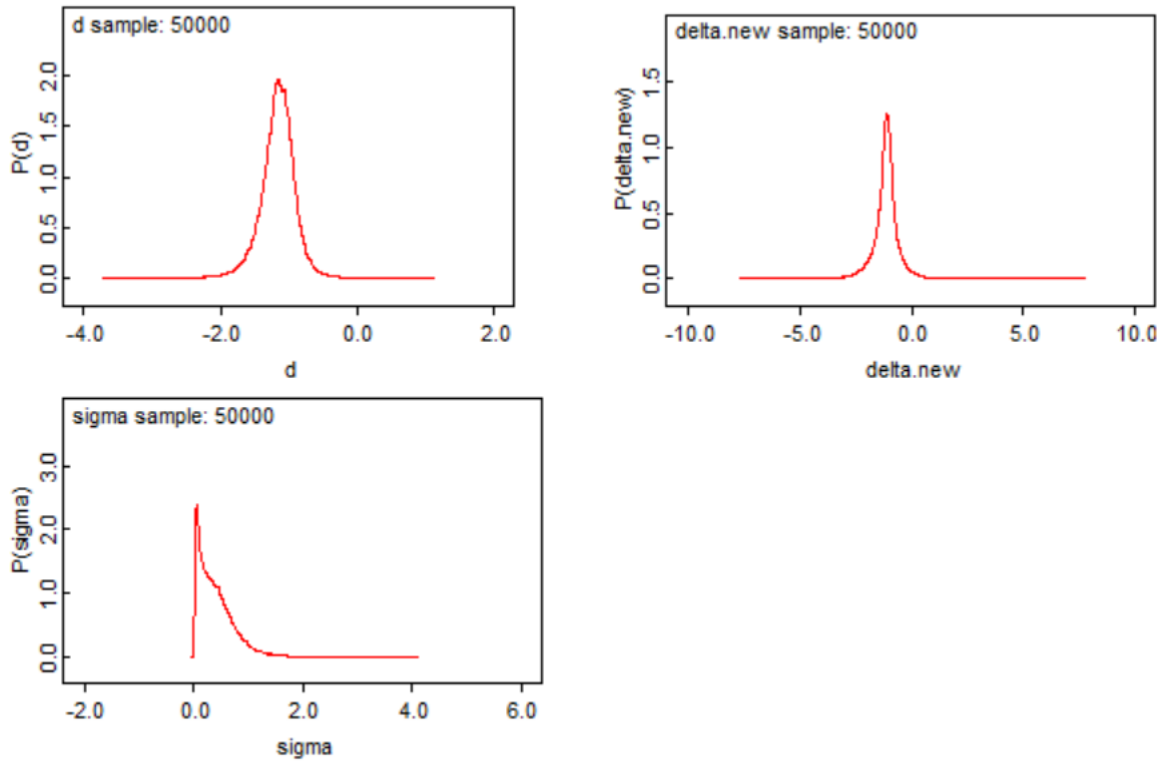
```
# Training
for(i in 1:n) {
        # Placebo
        rc[i] ~ dbin(pc[i], nc[i])
        logit(pc[i]) <- mu[i]
        mu[i] ~ dnorm(0.0, 0.00001)

        # Drug
        rt[i] ~ dbin(pt[i], nt[i])
        logit(pt[i]) <- mu[i] + delta[i]
```

```
        delta[i] ~ dnorm(d, tau)
}
```

We set uniformative priors for all variables:

```
d ~ dnorm(0.0, 0.000001)
tau ~ dgamma(0.001, 0.001)
```

And run the simulation. We start with burning the first 5000 observation and update the model with the next 50000 samples. We then investigate the densities for δ and its mean d and variace σ:



Visually we notice that the mean of δ is tightly distributed around -1, so we might conclude that amantadine indeed causes the descrease in influenza cases.

Let's look at the stats:

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
d	-1.16	0.2514	0.002703	-1.713	-1.147	-0.6974	5001	50000
delta.new	-1.161	0.582	0.003618	-2.436	-1.137	0.004146	5001	50000
sigma	0.4041	0.331	0.006602	0.03284	0.332	1.222	5001	50000

We see that the mean of d is -1.16 with 95% credible set of $(-1.71, -0.70)$. That clearly points to the effectiveness of amantadine. However, the distribution for δ is wider, with 95% credible set of $(-2.44, 0.004)$. Although the right tail is slightly greater than 0 we still believe that the posterior shows the effectiveness of the drug.

Why do we use Bayesian analysis instead of conventional meta-analysis (eg. as specified in [2])? We could find several reasons [3]:

- By modeling τ the Bayesian meta-analysis takes into account the uncertainty around the heterogeneity variance;
- Posterior distribution for τ makes heterogeneity evaluation and investigation more reliable;
- We could perform sensitivity analysis by changing distributional assumptions and incorporating a priori knowledge into the model (it's a bit hack-ish way, but we could make the right tail of the 95% credible set for δ be less than 0 by tweaking the priors);

- We could add complexity by making a deep hierarchical Bayes model.

b) A good overview of the meta-analysis visualization toolbox is made by Kiran et al (2016) [4]. The authors define 2 main charts for meta-analysis – a forest plot and a funnel plot. With the forest plot we investigate the parameter estimates of each study and the overall pooled estimate. The funnel plot is a scatter plot, where each dot represents an individual study and is positioned according to its effect size or strength of association (x-axis) and the precision around its estimate (y-axis) [4].

Let's build both plots for our influenza data with a Python library PyMeta [5]. We start with a forest plot:

Study ID	Experiment Group		Control Group	
	event	number	event	number
Oker-Blom(1970)	16	141	41	152
Muldoon(1986)	1	53	8	52
Monto(1979)	8	136	28	139
Kantor(1980)	9	59	9	51
Pettersson(1980)	32	95	59	97
Quarless(1981)	15	107	20	99
Dolin(1982)	2	113	27	132
Reuman(1989)	3	317	5	159

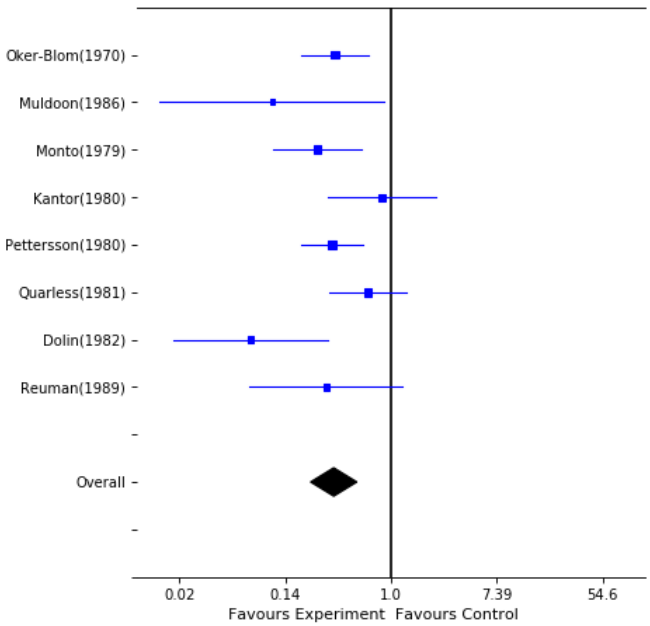
OR,MH,Random

Study ID	n	Effect(95% CI)	Weight(%)
Oker-Blom(1970)	293	0.35 [0.18, 0.65]	18.95
Muldoon(1986)	105	0.11 [0.01, 0.88]	3.77
Monto(1979)	275	0.25 [0.11, 0.57]	14.81
Kantor(1980)	110	0.84 [0.31, 2.31]	11.74
Pettersson(1980)	192	0.33 [0.18, 0.59]	19.97
Quarless(1981)	206	0.64 [0.31, 1.34]	16.66
Dolin(1982)	245	0.07 [0.02, 0.30]	6.99
Reuman(1989)	476	0.29 [0.07, 1.25]	7.11
Total	**1902**	**0.34 [0.22, 0.53]**	**100.00**

8 studies included (N=1902)

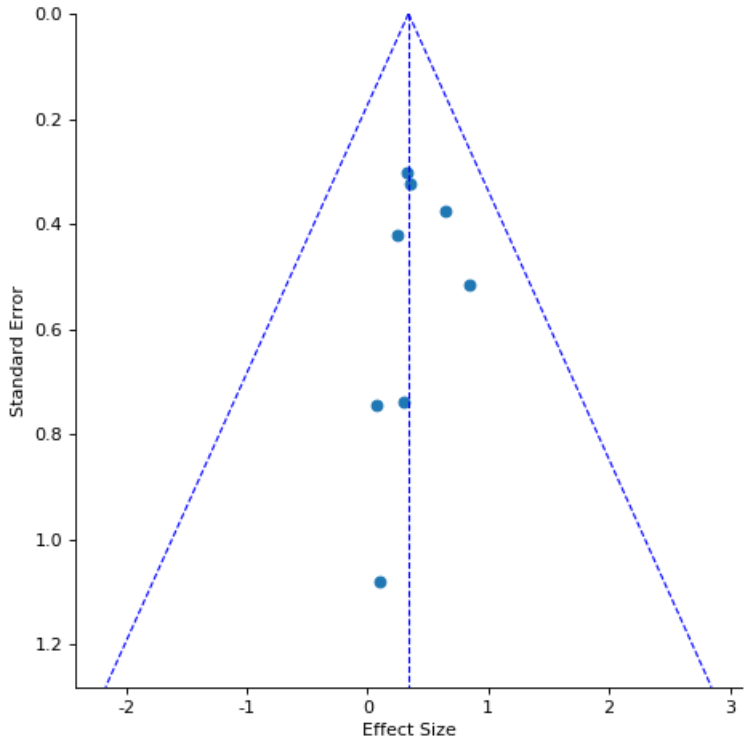
Heterogeneity: Tau²=0.160, Q=12.44 (p=0.087), I²=43.75%

Overall effect test: z=4.84, p=0.000



The squares on our chart show the weight given to each study – the larger the square the bigger the weight. The pooled effect is denoted by a diamond. In our case all the studies and the cumulative effect favour the amantadine treatment. We also notice that the studies that favour amantadine most are Muldoon(1986) and Dolin(1982). However, those are also the studies that have longer confidence intervals (represented by horizonatal lines). The studies which confidence intervals cross the vertical lines (in our case Kantor(1980), Quarless(1981) and Reuman(1989)) are deemed inconclusive.

Then we build a funnel plot:



We see that our funnel chart is symmetric with all points located inside the triangle. We might expect the lower points to be more widely spread (i.e. results from smaller studies tend to deviate from the average), however, that's not the case for the influenza data. So we can conclude that the data is "well-behaved" with no publication bias.

Note: the full OpenBUGS code is available at *Influenza.odc* in the attached archive, the python code for generating plot is named *plot_q3.py* and could be run by *python3 plot_q3.py* command.

References

[1] Engineering Biostatistics: An Introduction using MATLAB and WinBUGS. Brani Vidakovic – Wiley Series in Probability and Statistics.

[2] Introduction to Meta-Analysis – Charles DiMaggio. http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/resources/Bayes/Bayes4/metaAnalysis2011.pdf

[3] Bayesian Meta-Analysis. https://e-l.unifi.it/pluginfile.php/371990/mod_resource/content/1/Meta-analisi_lezione2.pdf

[4] Graphics and Statistics for Cardiology: Data visualisation for meta-analysis. https://heart.bmj.com/content/103/1/19.full

[5] PythonMeta library for visualizing meta-analysis. https://www.pymeta.com/help/#pythonmeta