


```

model{
  ## sampling model for the data
  for(i in 1:2){      ## loop over observations
    r[i] ~ dbin(p[i],n[i])      ## p is unknown parameter
  }

  ## priors
  p[1] ~ dunif(0,1)      ## uniform distributions
  p[2] ~ dunif(0,1)

  ## compute quantities of interest
  delta <- p[1] - p[2]      ## difference in probs
  delta.up <- step(delta) ## is delta > 0???

  ## log of the odds ratio
  lambda <- log( (p[1]/(1-p[1])) / (p[2]/(1-p[2])) );
  lambda.up <- step(lambda) ## is lambda > 0???
}

## data
list(r=c(83,72),n=c(86,86))

```

Alternative Parameterization:

```

model{
  ## sampling model for the data
  for(i in 1:2){      ## loop over observations
    r[i] ~ dbin(p[i],n[i])      ## p is unknown parameter
  }

  ## compute quantities of interest
  ## log of the odds ratio
  delta <- p[1] - p[2]
  lambda <- log( (p[1]/(1-p[1])) / (p[2]/(1-p[2])) );
  lambda.up <- step(lambda) ## is lambda > 0???

  ## priors
  v[2] ~ dnorm(0,.01);      ## vague prior, logits
  logit(p[2]) <- v[2];      ## convert to probability
  v[1] <- v[2] + vdelta;    ## difference in logits
  vdelta ~ dnorm(0,.01);    ## vague prior on difference
  logit(p[1]) <- v[1];      ## convert to probability
}

## data
list(r=c(83,72),n=c(86,86))

## initial values
list(v=c(NA,.5),vdelta=0)

```

I. Cigarette Example (from Simon Jackman)

Classical Hypothesis Testing Structure

$$\mathbf{H_0: p_1 - p_2 = 0}$$

$$\mathbf{H_1: p_1 - p_2 \neq 0}$$

The decision rule for this problem is:

$$\text{If } \frac{(\hat{p}_1 - \hat{p}_2) - (p_1 - p_2)}{\sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{m} + \frac{\hat{p}_2(1-\hat{p}_2)}{n}}} > z_{\alpha/2} \text{ or}$$

$$\frac{(\hat{p}_1 - \hat{p}_2) - (p_1 - p_2)}{\sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{m} + \frac{\hat{p}_2(1-\hat{p}_2)}{n}}} < z_{\alpha/2} \text{ then Reject } H_0:$$

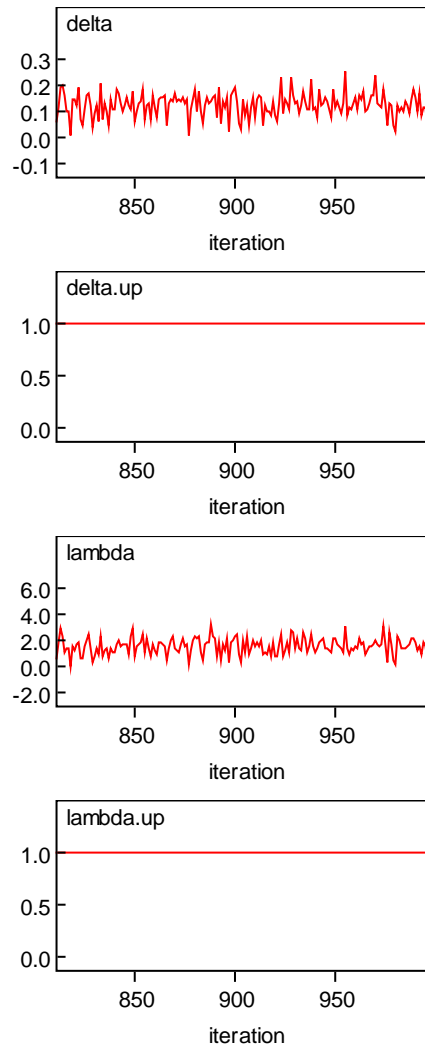
$$\text{Numerically: } \frac{.965 - .837}{\sqrt{\frac{.965 * .035}{86} + \frac{.837 * .163}{86}}} = 2.88$$

For a one tail test this corresponds to a p-value of about 0.002.

A. Bayes Results – Not much different! P-value of .004.

1. Load cigarette.odc into WINBUGS
2. Select “Model” and then the drop-down option “Specification”
3. Go to cigarette.odc and double-click “model” then push “Check Model” button.
4. Go to cigarette.odc and double-click “list” and then push “load data”
5. Now click “gen inits” (this generates initial values for the parameters).

6. Now we set up our sample monitoring. Click “Inference” and then select “Samples” from the drop-down options. “Sample Monitoring Tool” will appear with the cursor flashing in the “node” box.
7. In the WINBUGS language *everything* is couched in “nodes”. It is *not* a normal programming language! In WINBUGS the MLE is set up as a distribution, the priors are distributions, then the posterior is usually a formula – using R type syntax. In the cigarette case the nodes are “delta”, “delta.up”, “lambda”, and “lambda.up”.
8. One at a time type this in the “nodes” box and click “set”
9. Now, to see traces of the nodes use the drop-down option on the side of the “nodes” box to select each node in turn and click on “trace”
10. Now select “model” and select the drop-down option “update” and it brings up the “Update tool”. It defaults to 1000 updates so just accept that and click “update” and the model runs.



11. The traces are plots of the variable value against the iteration number.

12.

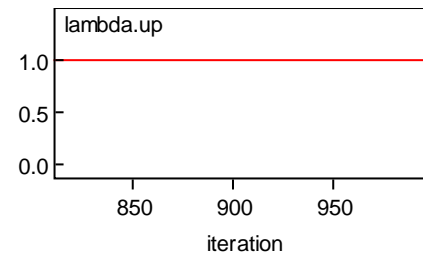
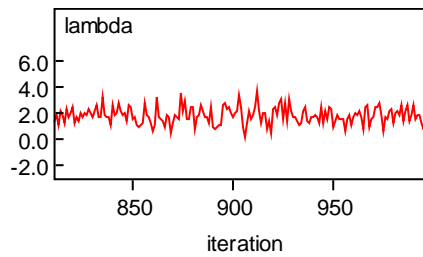
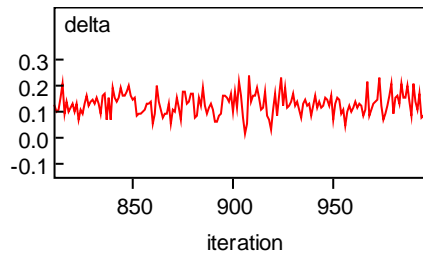
node	mean	sd	MC error	2.5%	median	97.5%	start	sample
delta	0.124	0.04575	0.001473	0.03341	0.1263	0.2158	1	1000
delta.up	0.996	0.06312	0.001908	1.0	1.0	1.0	1	1000
lambda	1.549	0.6269	0.01987	0.3899	1.526	2.856	1	1000
lambda.up	0.996	0.06312	0.001908	1.0	1.0	1.0	1	1000

13. The “step” function, $\text{step}(e)=1$, if $e \geq 0$, 0 otherwise. So the means of delta.up and lambda.up are just p-values! That is, 996 of 1000 times delta.up and lambda.up were greater than one.

B. Alternative Specification – Same steps as above but note that this illustrates the maddening fact about WINBUGS – It does not obey the normal programming rules!! Because everything is nodes the compiler does not care what order you type them in!

1. Select “Model” and the drop-down option “specification”, double-click on “model” under the “Alternative Parameterization” label, click “check model” and a dialog box pops up and tells you that “the new model will replace the old one” and you click “OK”.
2. Double-click on the word “list” in the Alternative Parameterization and click “load data”.
3. Now click “compile” but instead of choosing to generate initial values we have initial values for this example so double-click on “list” below where it says “## initial values” and then click “load inits”

4. Select “Inference” and the drop-down option “Samples” and select the nodes “delta”, “lambda”, and “lambda.up”. Set up a trace for each node.
5. Now select “model” and select the drop-down option “update” and it brings up the “Update tool”. It defaults to 1000 updates so just accept that and click “update” and the model runs.



6. The statistics are:

Node	mean	sd	MC error	2.5%	median	97.5%	start	sample
delta	0.1289	0.04222	0.001913	0.05163	0.1274	0.2162	1	1000
lambda	1.833	0.6945	0.02982	0.6387	1.811	3.308	1	1000
lambda.up	0.999	0.03161	0.001004	1.0	1.0	1.0	1	1000

7. So our p-value for this experiment is .001.

8. Note that it makes more sense to write the model after the MLE statement as follows:

```
## priors
v[2] ~ dnorm(0,.01); ## vague prior, logits
vdelta ~ dnorm(0,.01); ## vague prior on difference
v[1] <- v[2] + vdelta; ## difference in logits
logit(p[1]) <- v[1]; ## convert to probability
logit(p[2]) <- v[2]; ## convert to probability
```

Note how complex these priors are! The two variables of interest – p[1] and p[2] – are random variables embedded in the logit formula! Technically,

$$p \sim \frac{e^v}{1+e^v}, \quad -\infty < v < +\infty$$

And

$$v \sim \frac{1}{\sqrt{200\pi}} e^{-\frac{v^2}{20000}}$$

```
## compute quantities of interest
## log of the odds ratio
delta <- p[1] - p[2]
lambda <- log( (p[1]/(1-p[1])) / (p[2]/(1-p[2])) );
lambda.up <- step(lambda) ## is lambda > 0???
```