# Code Walk-Through

This code walk-through provides an annotated guide to the JAGS implementation of the model. We use the `ssLM_TTBTallyWADDGuess.txt` model as an example. This is the model applied to the Walsh and Gluck (2016) data. The `ssLM` indicates "strategy switch latent mixture'' and the `TTBTallyWADDGuess` indicates the four strategies considered. The models used for the other application are extremely similar, but differ in one key code block that depends on the strategies considered, as described below.

Each model is run by three sequential scripts, indicated by the `_A` , `_B` and `_C` suffixes. Just the one statistical model is being applied, but the use of three scripts allows for efficient inference of parameters conditional on other parameters, rather than via one large joint distribution that is hard to interpret.

## Script A

The first lines are a comment describing the model, and marking the beginning of the model.

```
# switching strategies, latent mixture: TTB, Tally, WADD, Guess
model{
```

The next section start loops over all of the participants and trials.

```
  for (i in 1:nSubjects){
    for (t in 1:nTrials){
```

The next section models the observed decisions. The variable `y[i, t]` is the decision the $i$th participant made on the $t$th trial, and is provided as observed data. It corresponds to $y_{it}$ in the equation. The variable `theta[i, t, a]` provides the prediction of the $a$th strategy on the $t$th trial completed by the $i$th participant, corresponding to $\theta_{ita}$. The variable `z[i, j]` is the strategy the $i$th participant is using in their $j$th stage, corresponding to $z_{ik}$. The variable `omega[i, t]` is the stage the $i$th participant is in on the $t$th trial, and corresponds to $\omega_{it}$. The variable `epsilon[i]` is the error-of-execution for the $i$th participant, corresponding to $\epsilon_i$.

```
      # Observed decisions
      y[i, t] ~ dbern(equals(theta[i, t, z[i, omega[i, t]]], 1) * (1 -
epsilon[i])
                  + equals(theta[i, t, z[i, omega[i, t]]], 0) * epsilon[i]
                  + equals(theta[i, t, z[i, omega[i, t]]], 0.5) * 0.5)
```

The next section determines the stage the $i$th participant is in on the $t$th trial. The variable `omegaTmp[i, t, k]` tests whether the current trial is greater than the $k$th potential switch point, represented by the variable `tau[i, k]`. The `step` function returns 1 if `t` is strictly greater than `tau[i, k]` and 0 otherwise. The stage `omega[i, t]` is calculated by summing over `omegaTmp[i, t, ]` to count the number of switch points exceeded by the current trial, and adding 1 to start at the first stage.

```
      # Stage for this subject at this trial
      omega[i, t] = sum(omegaTmp[i, t, ]) + 1
      for (k in 1:nMaxSwitches){
        # is trial to the right of boundary?
        omegaTmp[i, t, k] = step(t - tau[i, k] - 1)
      }
```

The next section defines the behavior of each strategy for each participant on each trial. This section of the script will vary according to the set of strategies considered in different applications. Our implementation for the Walsh and Gluck (2016) involves custom JAGS functions for the take-the-best, tally, and weighted-additive strategies, using the module developed by Selker (2018). This module provides the `TTB`, `TALLY`, and `WADD` functions. A simpler implementation could be achieved by supplying `theta` as an observed variable. The final brace closes the loop over trials.

```
      # Behavior for each strategy on each trial
      theta[i, t, 1] = TTB(cues[stimA[i, t], ], cues[stimB[i, t], ],
searchOrder)
      outputTALLY[i, t, 1:2] = TALLY(cues[stimA[i, t], ], cues[stimB[i, t], ],
searchOrder)
      theta[i, t, 2] = outputTALLY[i, t, 1]
      outputWADD[i, t, 1:2] = WADD(cues[stimA[i, t], ], cues[stimB[i, t], ],
cueEvidence, searchOrder)
      theta[i, t, 3] = outputWADD[i, t, 1]
      theta[i, t, 4] = 0.5
  }
```

The next section start the definition of participant-level priors, beginning with the prior on the accuracy of execution for each participant.

```
    # Error of execution
    epsilon[i] ~ dunif(0, 0.5)
```

The next section defines the potential switch points. The variable `tauTmp[i, x]` is the $x$th potential switch point for the $i$th participant. There are a maximum of `nMaxSwitches` potential switch points, which is provided as an observed variable, and corresponds to $m$. The variable `gamma[i]` is the switching propensity of the $i$th participant, and corresponds to $\gamma_i$. The variable `tau[i, ]` sorts the switch points in `tauTmp[i, ]` to implement the order constraint, and corresponds to the vector $\tau_i$.

```
    # Strategy switch point
    tau[i, 1:nMaxSwitches] = sort(tauTmp[i, ])
    for (x in 1:nMaxSwitches){
      tauTmp[i, x] ~ dunif(1, gamma[i]/(1-gamma[i])*nTrials)
    }
```

The next section defines the strategy propensities. The variable `x[i]` defines to which latent-mixture component for the strategy propensity distribution the $i$th participant belongs, and corresponds to $x_i$. The variable `gamma[i]` is the propensity for the $i$th participant. It is sampled from a Gaussian distribution truncated between 0.5 and 1, with mean `mu[g]` and standard deviation `sigma[g]` for the $g$th latent-mixture component.

```
    # Strategy propensity
    x[i] ~ dbern(phi)
    gamma[i] ~ dnorm(muGamma[x[i]+1], 1/sigmaGamma[x[i]+1]^2)T(0.5, 1)
```

The next section defines the strategies used in the various stages. The variable `z[i, 1]` is the strategy used by the $i$th participant in their first stage, and corresponds to $z_{i1}$. The variable `piPrime` is the base-rate over strategies for their use in the first stage, and corresponds to the vector $\pi'$. The loop over `x` indexes the strategies used in potential subsequent stages. For a maximum of `nMaxSwitches` switch points there is a maximum of `nMaxSwitches+1` stages. The variable `z[i, x]` is the strategy used by the $i$th participant in their $x$th potential stage, and corresponds to $z_{ix}$. The variable `pi` is the transition matrix, given the probability of transitioning from a row to a column. It corresponds to the matrix $\Pi = [\pi_{ab}]$. The strategy `z[i, x]` for the $i$th participant in their $x$th stage is sampled from the probabilities in the row corresponding to the previous strategy `z[i, x-1]`. The final brace closes the loop over participants.

```
    # Strategy use
    z[i, 1] ~ dcat(piPrime)
    for (k in 1:nMaxSwitches+){
      z[i, k+1] ~ dcat(pi[z[i, k], ])
    }
  }
```

The next section begins the definition of priors at the group level, starting with the two latent mixture components that comprise the distribution of switching propensities. The variable `muGammaTmp[g]` is the mean for the $g$th component. It is sorted to define `muGamma[g]` and impose the order constraint $\mu_1 > \mu_2$. The variable `sigmaGamma[g]` is the standard deviation of the $g$th component. The variable `phi` is the base-rate of participants belonging to the second components, and corresponds to $\phi$.

```
  # Gamma groups
  for (g in 1:2){
    muGammaTmp[g] ~ dunif(0.5, 1)
    sigmaGamma[g] ~ dunif(0, 0.5)
  }
  muGamma = sort(muGammaTmp)
  phi ~ dunif(0, 1)
```

The next section defines the base-rates and transition probabilities for the strategies. The variable `piPrime` is the base-rate for the initial strategy. Its definition in terms of the Dirichlet distribution uses the variable `baseStrategies`, which is simply a vector of length `nStrategies` with 1 in each position. The prior for the matrix of transition probabilities `pi` is defined in terms of the prior on each row, indexed by `j1`. These priors use the variable `baseStrategies`, and are defined cell-by-cell as indexed by `j1` and `j2`. Since it is not possible for a strategy to transition to itself, `basePi[j1,j2]` is 0 when `j1` equals `j2`, corresponding to $\pi_{aa} = 0$. All of the other cells in `basePi[j1,j2]` are equal to 1. The value of 1 is used to make the prior on the transition probabilities $\pi_{ab}$ equal, and relies on the `dcat` distribution in JAGS automatically normalizing probabilities. The final brace closes the model definition.

```
  # Strategy baserate
  baseStrategies = rep(1, nStrategies)
  piPrime ~ ddirch(baseStrategies)
  for (j1 in 1:nStrategies){
    for (j2 in 1:nStrategies){
      basePi[j1, j2] = 1 - equals(j1, j2)  # 0 if j1 == j2, 1 otherwise
    }
    pi[j1, 1:nStrategies] ~ ddirch(basePi[j1, ])
  }
}
```

This script formalizes the entire model, but the `_A` version is only used to determine the latent-mixture representation of the switching propensities. This means the only variables monitored are `x` and `phi`. The posterior distribution of the base-rate `phi` is used to determine if the latent-mixture distribution has one or two components. If there are two components, the mode of `x` is used to determine to which group each participant belongs.

## Script B

The `_B` script removes from the `_A` script the inferences about the $x_i$ latent indicator variables. This involves removing the lines

```
x[i] ~ dbern(phi)
```

and

```
phi ~ dunif(0, 1)
```

Instead, the modal values of `x` from the inferences for `_A` script are provided as observed to the `_B` script.

The parameters monitored for the `_B` script are `tau`, `gamma`, `muGamma`, `sigmaGamma`. Collectively, these define the inferred switching propensity distribution, the individual switch propensities, and the switch points themselves.

The joint posterior distribution for the switch points is then summarized as described in the paper, by finding the modal set of switch points, and all alternative combinations that are within some threshold of posterior probability, up to some maximum of alternative combinations.

## Script C

The `_C` script removes from the `_B` script the inferences about the switch points and switch propensities. This involves removing the lines

```
# Strategy switch point
tau[i, 1:nMaxSwitches] = sort(tauTmp[i, ])
for (x in 1:nMaxSwitches){
  tauTmp[i, x] ~ dunif(1, gamma[i]/(1-gamma[i])*nTrials)
}
```

and

```
# Strategy propensity
gamma[i] ~ dnorm(muGamma[x[i]+1], 1/sigmaGamma[x[i]+1]^2)T(0.5, 1)
```

and

```
# Gamma groups
for (g in 1:2){
  muGammaTmp[g] ~ dunif(0.5, 1)
  sigmaGamma[g] ~ dunif(0, 0.5)
}
muGamma = sort(muGammaTmp)
```

Instead, the sequences of switch points identified in the inference from the `_B` script are supplied as observed to the `_C` , which is run independently for each identified set.

The monitored parameters are `z`, `pi`, `piPrime`, and `epsilon`. These provide inferences about the strategies used, the base-rate of their first use and transitions, and the error of execution. It is also possible to monitor `theta` if these model behaviors were calculated within JAGS.