# Characterizing Neural Responses: feature selectivity, spiking statistics, and information

Leslie Osborne, Matthew Macellaio

September 7, 2015

## Contents

Tutorial 1

# Opening the black box

When you drop an electrode into the brain, you can detect the tiny electrical currents emitted by neurons as they communicate. The currents are typically brief pulses termed "spikes", and it is the temporal pattern of spikes and the intervening silences that constitute the code by which neurons transmit information. Reading that code has been, and continues to be, one of the central challenges of the field. In this tutorial you will learn how to analyze the responses of a sensory neuron to determine what features of the sensory stimulus trigger spikes, how variable that response is, and how much information about stimulus features those spikes encode – in other words, how to (begin to) answer the question, "what does this neuron do?"

You will be analyzing the responses of a neuron in extrastriate cortical area MT, a visual area in which many neurons respond selectively to visual motion stimuli. The recording used an extracellular electrode to detect voltage fluctuations near the membrane surface while visual stimuli were displayed on a monitor. We have filtered that electrical signal and isolated the action potentials (spikes) for you.

Load the data file `MTneuron.RData` into R.

```
1       > load(../Data/MTneuron.RData)
        > print(ls))
```

You'll see the following arrays:

```
RFmap
dirtune
directions
theta
```

An experimenter's first task is to locate the neuron's receptive field (RF) within the visual field so that stimuli can be properly configured to drive the neuron. We did this by presenting small patches of moving dots at different screen locations and recording the activity of the isolated neuron. The array `RFmap` represents the neuron's response to the moving dot patterns with respect to location on the screen. The random dot patterns moved within a 2°x2° (square) aperture that was positioned on a 15 x 10 grid with 2° spacing. The center of the screen is located at grid position (x=12, y=9) with the grid

position (1,1) in the upper left corner of the screen. The entire stimulus set tiled a 30° by 20° field of view, from -22° to +6° in x and -2° to +16° in y with respect to the center of gaze. The stimulus moved for 250 ms and was repeated 16 times at each location. We have pre-sorted the data structure array for you, such that `RFmap` has dimensions of y-position (10) by x-position (15) by repetition (16) by spike times (maximum number 24). Because there were different numbers of spikes at different locations and repetitions, that dimension is zero-padded. Inspect the size of `RFmap`.

```
> dim(RFmap)
```

## 1.1 Exercise 1. Map the receptive field of a neuron in cortical area MT

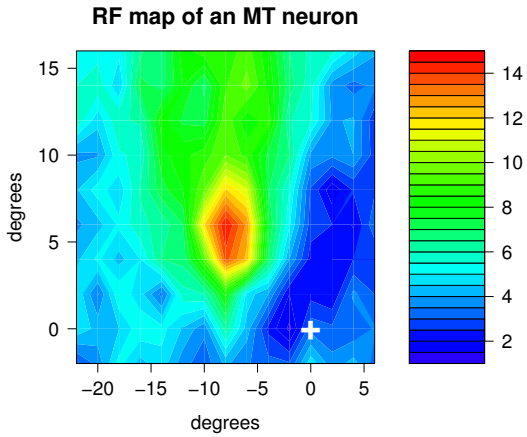
What we mean by receptive field is how many spikes does the neuron fire as a function of where the stimulus is shown on the screen (the grid positions). This exercise involves counting the spikes at each spatial location across all repeats. If you are familiar with coding, how to construct an algorithm to do this may be obvious. If you are not, then one way to break the task down is to start with "pseudo-code", meaning to write out what you want to do in words, then figure out how to implement the steps you came up with in your programming language (e.g. R or Matlab, etc).

```
# Writing pseudo-code is a convenient starting point
# The more explicit you can be, the more obvious
# how to substitute real code for the words

      for (each y position on the grid, out of 10){
           for (each x position on the grid, out of 15){
             find the non-zero entries in the 3rd (repeats) and
                    4th (spike times with extra 0s) dimensions of RFmap
                        at that grid position
             count the number spikes
             store the sum as a function of x,y grid location in order
                to plot later
         }
      }
```

If you can understand this pseudo-code, you are thinking like a computational neuroscientist. Translating the steps into actual code is a detail. How fast you are at it depends on how well you know the language you are working in. If you are comfortable with `R`, perhaps you know how to write a script to do the above already, or can figure it out with the hints we have written below. If not, or to check your work, feel free to use the script

we wrote, `plot_RFmap_v2.R`, in the Code subfolder. However, no matter how you choose to use this tutorial, we encourage you to think the steps through for yourself.




### Hint 1 (How to find the total number of spikes fired at each spatial location)

A convenient way to determine the total number of spikes fired on each stimulus presentation is to count how many non-zero spike times there are. For the grid position (1,1), get spikes for all repeats:

```
> numspks(1,1)<- sum(RFmap[1, 1, , ] != 0)
```

You will get a tidier figure at the end if you only include stimulus driven spikes, in this case, spikes that occur later than 50ms after the onset of the stimulus. Why? This neuron, like most, has a latency period before it will begin to fire spikes. This neuron's latency is about 50ms. So if you get spikes that happened later than 45ms then you are pretty sure to be counting all of the relevant spikes and ignoring the irrelevant spikes.

```
> numspks(1,1)<- sum(RFmap[1, 1, , ] >45)
```

### Hint 2 (How to plot the a 2D color density map of the neural responses)

If x and y are vectors indicating locations of the stimulus, use image to plot the average spike count at each location.

```
> x = -14:2:14
> y = 9:-2:-9
> image(x, y, numspks / nTrials) #create a color density plot
```

You might add a fixation point as well, to indicate where the receptive field is located with respect to the center of gaze.

```
> image(x, y, numspks / nTrials,
>  plot.axes={ axis(1); # plot the x−axis
> axis(2); # plot the y axis
> # The center of the visual field of the monkey was at (7.5, −7.5)
> # Let's put a + sign to mark the spot
> points(7.5, -7.5, pch = "+", cex = 2, col = "white", font = 2)})
```

Consult the script file `plot_RFmap_v2.R` for a complete algorithm for help or to check your work.

## 1.2 Determining feature selectivity: tuning for motion direction

Many sensory neurons are "tuned" for a particular feature of a stimulus, meaning that the firing rate changes smoothly as a function of that value. In MT, many neurons are tuned for the direction and the speed of motion, such that the rate changes smoothly as a function of direction or speed. The data array `dirtune` contains the responses of an MT neuron to motion in different directions. The visual stimulus consisted of random patterns of dots that moved coherently (all had the same direction and speed) in an aperture centered on the RF. Each trial in the experiment consisted of a 250 ms motion step in a single direction that was preceded and followed by 300 ms of a stationary pattern. The array `dirtune` has the dimensions trials by spike times, where the spike times are relative to motion onset (i.e. we clipped out the first stationary interval). Each stimulus was repeated a different number of times, so we have also included the array `theta`, a vector the same length as the first dimension of `dirtune`. Each number in `theta` represents a motion direction, from 1 to 24 with the actual directions in degrees stored in the vector `directions`. The ordering of `theta` is the same as `dirtune`, so `which(theta==1)` will return the indices corresponding to all the trials in `dirtune` where the motion direction was -180°.

## 1.3 Exercise 2. Plot a direction tuning curve

Apply the skills you learned in Exercise 1 to find the average number of spikes fired for each motion direction from the array `dirtune` and plot that value as a function of direction (stored in the vector `directions`) for this neuron. Refer to `plot_tuncurve_v2.R` for tips.

### Hint 1 (Getting the spike count for a single direction)

As with the RF mapping, we want to get the average number of spikes fired per motion direction. However, unlike the RF mapping results, there are different numbers of repeats

for each direction.

```
> nReps <- rep(0, nDirs) #create a vector of zeros of the needed size
> for (n in 1:nDirs){
> nReps[n] <- sum(theta == n) # populate the vector with the number
                              # of repeats for each direction
> }

#You'll need to use nReps to normalize the spike counts for each direction.

> getreps <- which(theta == 1)
> for (i in 1:length(getreps)){
> spks <- dirtune[getreps[i], ]) # get the data for one motion direction
> spks <- spks[spks > 0] # keep only those elements > 0
> counts[1,i] <- length(spks) # count how many, store
> }

# Or another way that doesn't require a loop on stimulus repeats:

> getreps <- which(theta==1)
> nspks[1] = sum(dirtune[getreps,]>0)
```

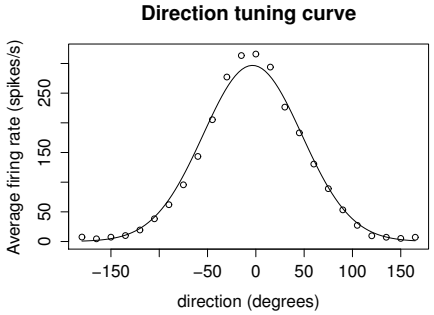
**Hint 2 (How to construct a for loop to do all the directions)**

Here we select a direction, and then loop through all the repeats of that direction to count the spikes, but you could also use the previous tip to make your code more compact and elegant.

```
> # for each direction
> for (n in 1:nDirs){
> # find which trials are associated with that direction
> index <- which(theta == n)
> # for each trial
> for (i in 1:length(index)){
> # find the spikes
> spks <- which(dirtune[index[i], ] > 0)
> # take only those between 45 and 355, since this includes the stimulus-driven
    spikes
> spks <- spks[spks > 45 & spks < 355]
```

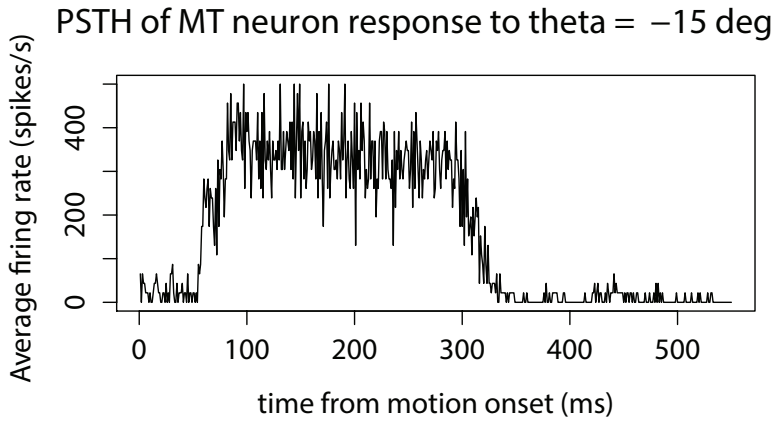
```
> # count them, store answer
> counts[n, i] <- length(spks)
  }
> # mean count per direction across repeats
> mcounts[n] <- mean(counts[n, 1:length(index)])
> }
```

This is what your figure will look like if you use the code we provided, `plot_tuncurve_v2.R`. We added a Gaussian fit to emphasize the shape of the curve.




## 1.4 Exercise 3: Plot the firing rate as a function of time and direction

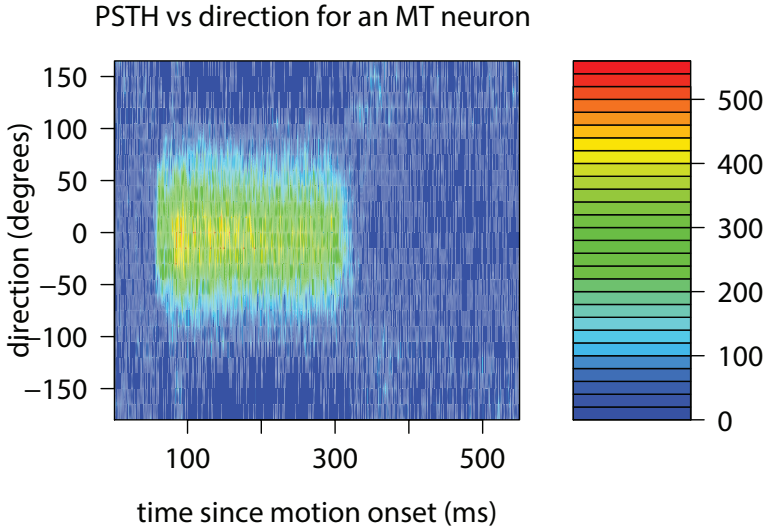
In this exercise you will be forming a peri-stimulus time histogram (PSTH).




The PSTH is the firing rate of the neuron, determined by averaging the response across repeats of the stimulus. If you compute the PSTH for each direction and plot what you get in direction and time, you'll be making this plot. In practice, you want to create an array called PSTH that is directions (24) by time (550) (or vice versa) and use the `plot` command

to plot the PSTH for one direction or the `image` to plot the result for all directions as a 2D color density plot.

Note: It is helpful here to reformat the data arrays from spike times (padded with 0s) into time vectors where each element represents a 1 ms time bin with respect to motion onset. Thus the response to each stimulus presentation will be a vector of 1s and 0s with the location of the 1s corresponding to the (integer) spike times in 1ms increments. An easy way to get spike times in ms is to use the command `round`, which rounds the numbers to their nearest integer value. To get the vectors, you want to create an array of zeros that we called `mydata` that is time by directions by repeats (550,24,46). Then for each integer spike time on each trial, set the value of `mydata` at that location to 1. The function `rowMeans`, which takes the mean of an array across its last dimension, will be helpful here.




**Hint 1 (How to convert spike times to a vector array of 1s and 0s at 1 ms resolution)**

```
> maxTime <- round(max(dirtune))
> mydata <- array(0, c(maxTime, nDirs, max(nReps)))

> # For each direction
> for (n in 1:nDirs){
> # which are the corresponding thetas,
> # i.e. where is the data for this direction stored in dirtune?
> index <- which(theta == n)
```

```
> # For each trial
> for (i in 1:length(index)){
> spks <- round(dirtune[index[i], ]) # make spike times integers
> spks <- spks[spks > 0] # take only those > 0
> mydata[spks, n, i] <- 1 # set to 1 in the array mydata
> }
> }
```

**Hint 2 (Creating the 2D PSTH array and plotting)**

You should already have a variable (`nReps` in the provided code) that indicates the number of repeats for each motion direction. Compute the PSTH, which is the neuron's trial–averaged response as a function of time for each motion direction.

```
> PSTH <- matrix(0, maxTime, nDirs)
> for (n in 1:nDirs){
> PSTH[ ,n] <- rowMeans(mydata[, n, 1:(nReps[n])]) # it is important to
    average just over the actual trials
> }
```

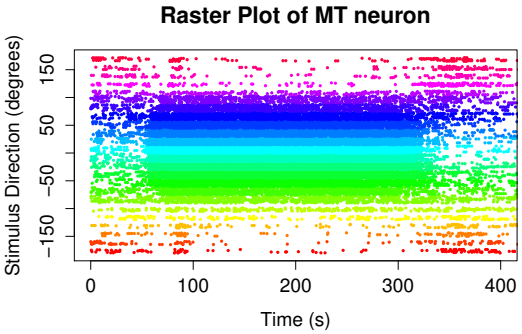
See the script `plot_2Dpsth_v2.R` to check your work or jump to the answer.

Tutorial 2

# The statistics of neural responses

How reliable are the neuron's responses to repeated stimuli? If we controlled all of the inputs to the neuron, the answer is that its outputs would be extremely reliable (e.g. Mainen ZF, Sejnowski TJ (1995) Reliability of spike timing in neocortical neurons. Science 268: 15031506). Therefore, the cellular mechanism of generating spikes is not inherently noisy. However, while recording from a neuron embedded in the brain, the responses to repeated presentations of visual stimuli are variable because we control only a limited number of experimental parameters that can modulate a neuron's input. In this exercise you will characterize the variability in an MT neuron's response from the experimenter's point of view. How much of that variability constitutes noise that affects the brain's estimate of motion from that neuron's responses is an open (and interesting) question.

To get a visual sense of the variability in spiking to repeated motion stimuli, run the script `plot_MTraster.R` to create a raster plot of the spike times on different trials for the dirtune data set. Each row of a raster plot indicates a spike time on one stimulus presentation with a dot or line. The rows indicate the neuron's responses on different trials. Color indicates the different direction values in the same order as directions. Remember that the motion stimulus begins at time 0 and lasts for 250 ms. What is the latency of this neuron's response to motion? Does the latency depend on direction? What do you notice about the variability of spiking during driven vs. spontaneous firing periods?




The time at which spikes are fired relative to the onset of visual motion varies from trial to trial, as does the number of spikes fired. Suppose that MT neurons encode motion

direction with the number of spikes fired in a certain time window. In that case, variability in the spike count will create variability in the estimates of motion direction for any downstream decoding of this neuron's responses.

## 2.1 Exercise 4. Calculate the probability of spiking for a range of directions

What is the probability of observing a particular count value in a time window of duration $T$, given the motion direction, $\theta$? i.e. $P_T(n|\theta = \theta_i)$. How does the shape of the distribution change with direction (and thus with the average firing rate)? Choose the preferred direction and an off-preferred direction and compare the count distributions by making a figure: you can find these directions from the tuning curve plot you made, or the 2D PSTH plot that shows the firing rate change with direction. Choose a time window of 250 ms from motion onset, and then try other time windows if you have time. The objective of this exercise is to gain experience in computing a probability distribution and to gain insight about spike count variability.

In Exercise 2, you used the array `dirtune` to find the spike count for each repetition and direction. You can use the same approach to count spikes in a particular time window. You could also use the array `mydata` that you created in Exercise 3. A helpful R command to quickly sum spikes across the entire time window is `colSums`, which takes the sum of an array along its first dimension (in this case, time), leaving us with an array of total spikes fired for each repetition of each stimulus direction presentation. At this point, you can compute the probability of the count given the motion direction in your time window, i.e. $P_T(n|\theta)$. The probability is like a frequency or a fraction - how many times did you observe a count of n out of all the repetitions?

A useful R command for measuring the fraction of trials is `hist(array, breaks)`, which outputs a histogram, computing the number of data points contained in bins, whose edges are defined by `breaks`. `hist(array, breaks)` returns a structure, and you will want the variable `counts` within that structure, which contains the counts within each bin. To access variables within a structure, you use the $ operator, as shown below and in `pcount_v2.R`.

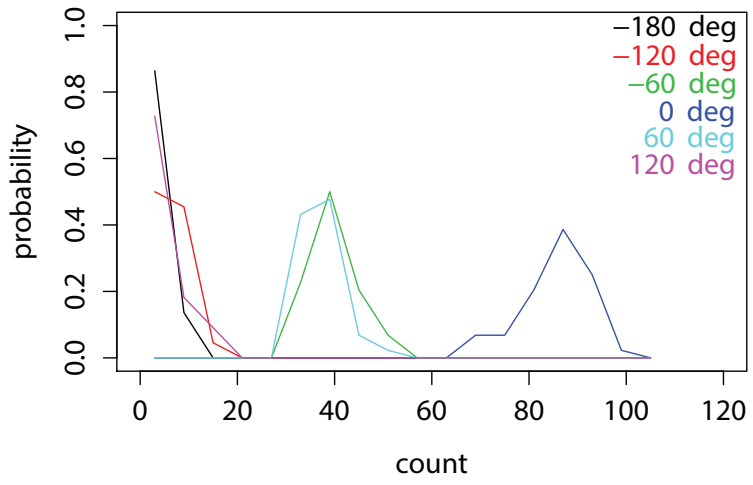
```
> hh <- hist(tmp, breaks = countbins, plot = FALSE)
> counts<-hh\$counts
```

IMPORTANT consideration for probability distributions: make sure to normalize the output to sum to 1 in order to translate the frequency of observation to a probability distribution. For the complete script, see `pcount_v2.R`.

### Hint 1

Make a 1D plot of $P(n|\theta = \theta_i)$ (probability of observing counts of different values given

that the motion was in a particular direction, $\theta_i$). Do this for the preferred direction, the highest firing rate, and an off-preferred direction with a lower firing rate. Use color or line type to denote the 2 different directions. How does the likelihood of observing a particular count value change as a function of direction? To really visualize this you could step through all the directions, but you will have a crowded figure unless you just choose a few! Is the mapping between count and direction unique? These distributions are the answer to the question, "Given the direction, what is the most likely spike count". The problem that the brain faces is better phrased, "Given the spike count, what is the most likely direction?" That distribution is $P(\theta|n)$.




### 2.1.1 Optional Exercise

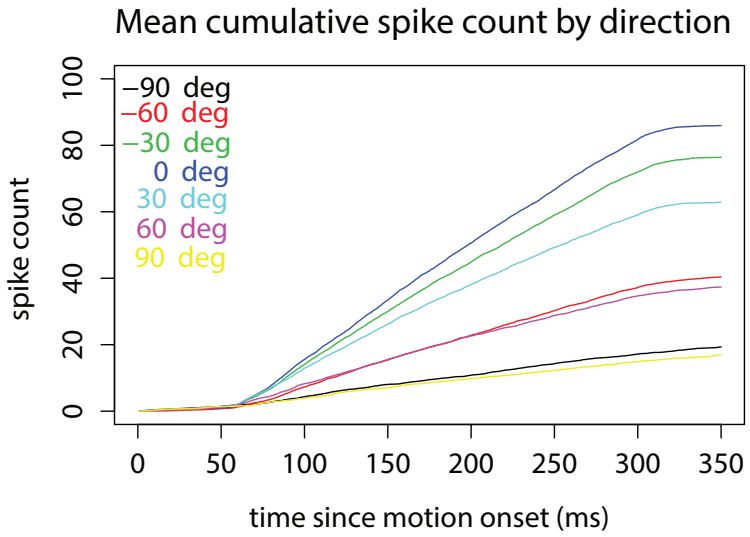
Compute and plot $P(\theta|n)$ for several values of $n$. To form that distribution, you want to select particular count values and then measure the frequency of directions. You can do this easily from the joint distribution of counts and directions. Again, make sure your distributions are properly normalized. Are the distributions overlapped? What is the significance of that to discriminating motion directions? The function `apply(array, dimension, function)`, which iteratively applies the given function to your array along the given dimension, may be useful here.

You estimated the count distributions assuming that the brain counts spikes in a given time window. In the sense that the visual system might integrate incoming signals about motion direction over time, it might be plausible to consider the total count as it accumulates over time. That is like taking an integrator and just counting spikes such that the number can only increase with each time step. The cumulative count stays the same until

another spike arrives and then it increments by 1. We are going to use the cumulative spike count in the next section, so if you have time do the following.

## 2.2 Exercise 5. Generate cumulative spike counts

Make an array `cumcounts` of the cumulative spike count as a function of time for each direction and repetition using the R command `cumsum`. This array will have the same size as data. Find the trial-averaged cumulative spike count for each motion direction and plot the result for several directions. We are most interested in the motion driven part of the neural response, so we might restrict the time window to the first 350 ms.




For further help, see the script `info_count_dir_v2.R` or ask us.

Tutorial 3

# Entropy and information

Our goal is to characterize not just the distributions of spike count themselves but rather the average extent to which the neural response allows us to identify the stimulus. This exercise will guide you through this process using information theory (Shannon and Weaver, 1949). For an extremely readable and elegant introduction to the application of information theory to neuroscience, see the book Spikes: Exploring the Neural Code by Rieke et al (1997).

The mutual information describes the relationship between two variables, say the spike count in a time window and the motion direction. One can choose other features of the spike train, say the time interval between spikes or the pattern of spikes and silences over time as the variable, but for simplicity we will continue with spike count. Consider count to be a symbol that this neuron uses to represent motion direction. The range of possible symbols that the neuron has in its repertoire might be determined by recording from it for a long period of time and playing every possible visual stimulus. To compute information, we want to express the variability in neural responses as an entropy in the units of bits. The entropy, $S$, of a probability distribution $P(n)$ over a time interval $T$ is given by

$$S_T(n) = -\sum_n P_T(n) \log_2(P_T(n)). \tag{3.1}$$

The entropy of the response is the upper bound on the amount of information a cell can transmit about the stimulus, i.e., its coding capacity. Intuitively, the distribution of counts determines the number of different responses or symbols that the cell can use to encode aspects of the stimulus. The entropy of the count distribution is itself constrained by the average count: $n(T)$. The higher the firing rate, the greater the range of counts that can be observed and, therefore, the more symbols available to encode the stimulus parameter.

While the entropy of the count distribution over all stimuli describes the total symbol space of the neuron, what we learn about motion direction from counting spikes depends on how variable the responses are. For each direction value $\theta$ in our stimulus set, we express the variability of the neural response to repetitions of that stimulus, $P(n|\theta)$, as a conditional entropy:

$$S_T(P(n|\theta)) = -\sum_{n_i} P_T(n_i|\theta) \log_2(P_T(n_i|\theta)) \tag{3.2}$$

The amount by which recording a certain spike count, $n$, at time $T$, reduces our uncertainty about a stimulus direction, $\theta_i$, is the information gained in bits, or the difference between

the total and noise entropies, as follows:

$$I_T(n,\theta) = S_T(P(n)) - \sum_{\theta_i} P(\theta_i) S_T(P(n|\theta_i)) \tag{3.3}$$

The mutual information is the weighted average of this quantity over all stimuli:

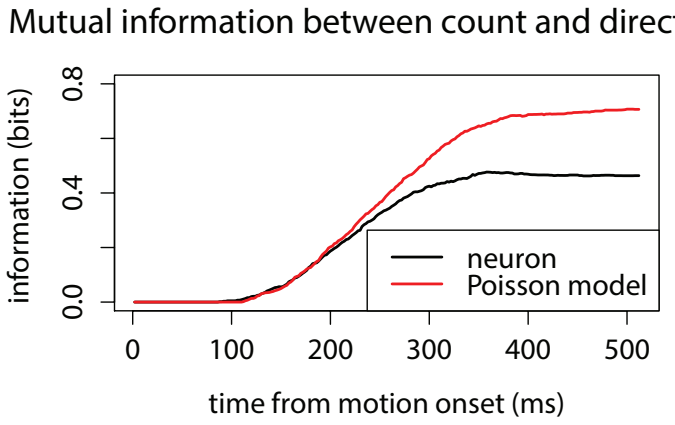
$$I_T(n,\theta) = \sum_{\theta_i} P(\theta_i) I_T(n,\theta_i) \tag{3.4}$$

You should be able to convince yourself that putting these equations together would yield the expression

$$I_T(n,\theta) = \sum_{\theta_i} P(\theta_i) \sum_{n_i} P_T(n_i|\theta_i) log_2 \frac{P_T(n_i|\theta_i)}{P_T(n_i)} \tag{3.5}$$

## 3.1 Exercise 6. Compute mutual information between spike count and direction

You will do this as a function of time since motion onset. Here, count at time $T$ means the number of spikes fired up to $T$, i.e. the cumulative count, which you may have calculated in Exercise 5. Note that the probability of occurrence of each motion direction and the probability of observing a count of n at time $T$ can be computed from the array data. What is the upper bound on the amount of information that could have been encoded (the total stimulus entropy)? Look at the script `info_countdir_v2.R` for help.




### 3.1.1 Optional Exercise

Normalize the information to its maximum value and then divide by the direction and trial-averaged cumulative spike count and plot. How many spikes, on average, does it take for this neuron to signal the motion direction? (The answer is about 3!)

### 3.1.2 Optional Exercise

Compare the information time course from the neural data to a Poisson model neuron with the same time varying firing rate. This isn't as difficult as it sounds! To create a Poisson model neuron with the same PSTH as the actual data, we need to make sure we remove any temporal correlations in the spiking. That means within each trial, the probability of spiking at time $t$ might depend on what happened just before. For a Poisson process, that would not be true. To simulate a Poisson model neuron, we are going to shuffle the array `data` in a particular way. The array `data_shuffle` will be the same size as `mydata`. To create the shuffled trials which are also vectors of 1s and 0s like data, we want to step through time and randomly select a trial to determine if a 1 or a 0 is at that time step. In other words, a shuffled trial would be the value of trial 12 at 1 ms, trial 23 at 2 ms, trial 5 at 3 m s, etc to 550 ms. In order for the PSTH to stay the same (the trial-averaged spike count), then we need to draw the trials without replacement. A convenient way to do that is to use the R command `sample`. `sample(1:nReps)` will return a randomized list of numbers from 1 to `nReps`. Be careful to shuffle trials only within each direction. Calculate the mutual information between the cumulative spike count and direction for the actual neuron and the shuffled model neuron. Are they different? What does that imply about this neuron's statistical behavior? See the script `info_countdir_v2.R` for help. To see a different behavior, load the second sample neuron, `MTneuron_2.RData` and redo the calculation and plot.