Masco; Probability and statistics: Practical 1

Academic year 2025-2026

Before starting

Instructions

This practical is composed of 3 parts:

- **Part I:** Discover the basics of the R software to generate random variables.
- **Part II:** Build an R function.
- **Part III:** Analyze neural spike data.

You will be working in groups of maximum 3 students. Log on the Ametice page and upload your reports with name of file: "Group.pdf".

You will need the packages 'STAR' and 'poisson'.

Quick overview of spike train data

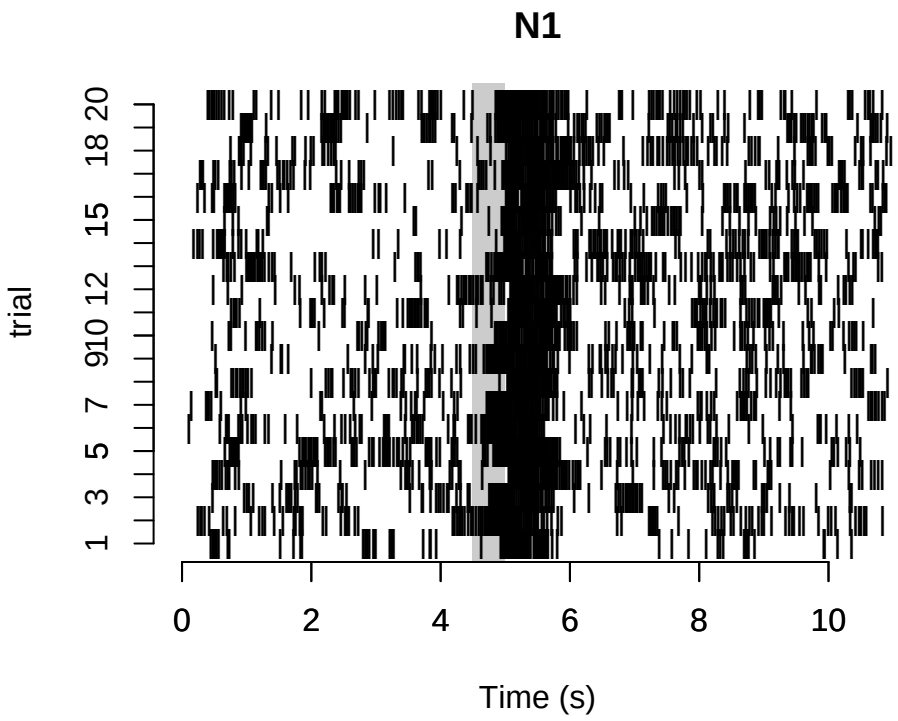
The study of series of action potentials (or spike trains) is a crucial issue to understand more about how neural information is represented and transmitted. It has been shown that neural firing rate increases in response to a stimulus.

The following figure has been generated using the library 'STAR'. More details will be given in Part III. This figure represents a raster plot. Each vertical line displays spike times. Measurements are taken on one neuron during approximately 10 seconds. 20 trials (replicates) have been done. The light grey part represents the time interval when stimulation is applied on the neuron.

```
library(STAR, quietly = TRUE)
```

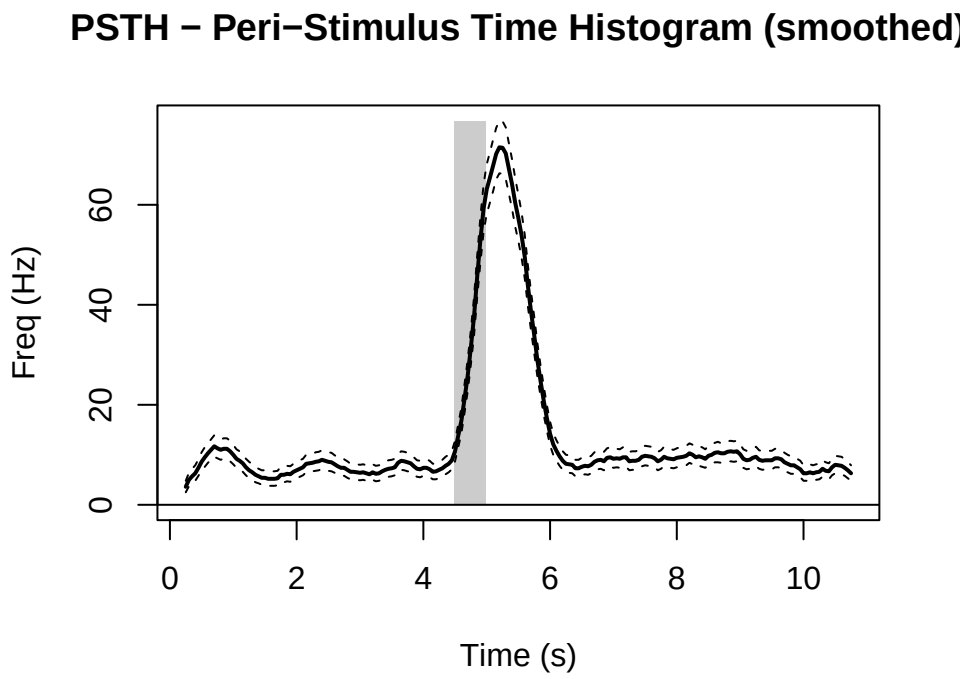
```
## This is mgcv 1.9-3. For overview type 'help("mgcv-package")'.
```

```
data(CAL1V)
# raster plot
plot(CAL1V[["neuron 1"]],stimTimeCourse=c(4.49,4.99),main="N1")
```



The following figure is obtained from a raster plot. It is known as a **Peri-Stimulus Time Histogram**. It represents an estimated firing rate of neurons over the time. This example clearly reveal the increase in firing rate just after the stimulation started.

```
## First get lists containing PSTHs from each neuron
psth1 <- psth(CAL1V[["neuron 1"]],breaks=c(bw=0.5,step=0.05),
              stimTimeCourse=c(4.49,4.99), plot=TRUE,
              main="PSTH - Peri-Stimulus Time Histogram (smoothed)")
```



Part I: Generating random variables

R includes standard functions to give the p.m.f (probability mass function), the p.d.f (probability density function), the c.d.f (cumulative distribution function) and to generate realizations of random variables from the most common distributions.

Questions:

- Generate a vector x of length 200 of equispaced values from 0 to 3 using the function 'seq'. Using the function 'dexp', create a plot of the probability density function of an exponential distribution with parameter 5.
- Draw a sample of 1000 values from an exponential distribution with parameter 5 using the function 'rexp'. Draw a histogram using the function 'hist'.
- Create a vector of integers from 0 to 10. Use the function 'barplot' to represent the probability mass function of a Poisson distribution with parameter 3. Generate 1000 random draws from this distribution. Use the functions 'table' and 'barplot' to give the probability mass function of these generated values.
- Run the CODE chunk 1 below. Explain what it does and what principles it is illustrating.
- Adapt the CODE chunk 1 by replacing the Gamma distribution with a Gaussian distribution with mean 0 and standard deviation 1 and the consider sample sizes of 30 and 150.

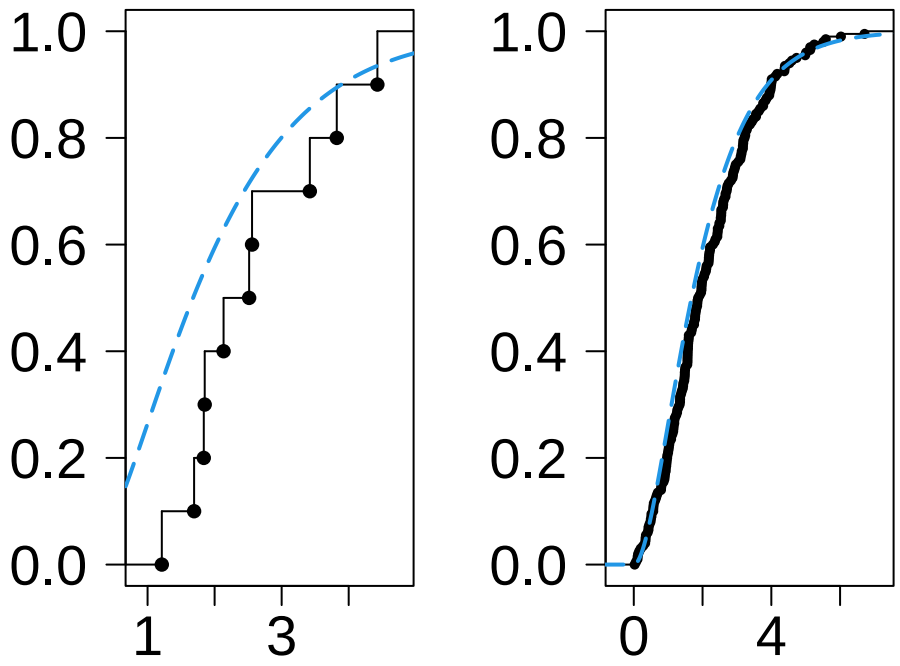
CODE chunk 1.

```
par(cex.lab=2)
par(cex.axis=1.75)
par(las=1)
par(mfrow=c(1,2))
par(mar = c(4, 4, 1, 1))

smallSample = rgamma(10, 2)
largeSample = rgamma(200, 2)
```

```r
xvals = sort(smallSample)
yvals = (0:10)/10
fhat = stepfun(xvals, yvals, right = TRUE)
plot(fhat, xlab = "", ylab = "", main = "", pch = 16, cex = 1)
xseq = seq(-1, 12, length = 300)
lines(xseq, pgamma(xseq, 2), lty = 5, lwd = 2, col = 4)

xvals = sort(largeSample)
yvals = (0:200)/200
fhat = stepfun(xvals, yvals, right = TRUE)
plot(fhat, xlab = "", ylab = "", main = "", pch = 16, cex = 0.7)
xseq = seq(-1, 12, length = 300)
lines(xseq, pgamma(xseq, 2), lty = 5, lwd = 2, col = 4)
```



Part II: simulating a Poisson process

Inverse Transformation Method

R alike other statistical softwares allows you to generate random variables. The starting point of many of these random value generators is the *Inverse Transformation Method.*

Remember that a c.d.f F takes values in $[0, 1]$. Let us consider a continuous F and let U be a uniform random variable on $(0, 1)$. Define X as follows:

$$X = F^{-1}(U),$$

where F^{-1} is the inverse c.d.f. Because F is continuous and monotonic, we can simulate a random variable from a **continuous** c.d.f F with inverse F^{-1} by simulating random numbers between 0 and 1.

The Inverse Transformation Method algorithm consists in the following steps:

- pick an integer value n,
- for i in $1:n$,
- draw u_i from a uniform and compute $x_i = F^{-1}(u_i)$,
- the resulting $(x_1, \ldots, x_n)$ is a realization of a random sample from F.

Generating a Poisson Process

Recall that a discrete random variable X has a Poisson distribution with parameter $\lambda > 0$ if its p.m.f can be written as follows:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!},\ k = 0, 1, 2, \ldots$$

A **counting process** $\{N(t),\ t \geq 0\}$ is a stochastic process defined by the number of occurrences of an event over a given time interval.

A **Poisson process** with intensity parameter $\lambda > 0$ is a counting process satisfying a set of conditions that we shortly describe here. It is required that:

- The number of occurrences in any time interval is *independent* of the number of events in any other disjoint time interval.
- Two events cannot occur at the same time.
- The number of events that occurs before time t, is Poisson distributed: $N(t) \sim Poi(\lambda t)$.
- The waiting time between two consecutive events is an exponentially distributed random variable with parameter λ.

Using these properties, one can simulate a Poisson Process by generating random variables $Y_i \sim Exp(\lambda)$, and defining the time of occurrence of events of a Poisson process with intensity λ at times $Y_0, Y_0 + Y_1, \ldots$

Questions:

- (**) Prove that the distribution of times between two successive events from a Poisson process with parameter λ are exponentially distributed.
- (*) Write a function to generate a Poisson process with parameter $\lambda = 5$. Use the Inverse transformation method to generate random values from the exponential distribution with parameter λ. Your R function will then have two input parameters λ and the total number of events N.
- Compare your function to the one available in the 'poisson' package 'hpp.sim'. To do so, plot the cumulative number of events for both functions. What is your conclusion?

Part III: real data analysis

In this section we will learn how to manipulate R objects alike lists, sublists, matrices... We will build some graphical descriptions of these data and discuss how to model them using Poisson processes.

Then, we will use the properties of Poisson processes and our knowledge about Bayesian statistics to 'test' if the intensity of the process is the same at two different time intervals.

Building Peri-Stimulus Time Histograms

We will be working with the dataset 'CAL1V' available from the package 'STAR'. Each sub-list contains the spike train (i.e, action potentials occurrence times) of one neuron during 1 stimulation (odor puff) with vanillin. Each acquisition was 10 seconds long. The command to the odor delivery valve was on between sec 4.49 and sec 4.99.

Neural spike train data are often modeled using Counting processes.

```
library(STAR)
## Load Vanillin responses data (first cockroach data set)
data(CAL1V)
## For details check
#?CAL1V
```

- Create a vector 'N1_S1' of spike event times of the first trial of the neuron 1.
- Check the number of events observed at each trial (i.e., compute the length of sub-list)
- Compute the range of values taken by the time of events
- Create a Peri-Stimulus Time Histogram. The idea is to aggregate (average) data over trials into bins of predetermined width (for example 0.5s), normalize the unit spike to the second, *Hint:* use the function 'hist'.

```
# set the disjoint interval from the hist function
breaks = seq(0,11, by=0.5) # number of spikes per half seconds !

# create a matrix of counts within each half second intervals
neuron1_mat_counts = matrix(nrow = 20, ncol = length(breaks)-1)
for (i in 1:length(Neuron_1))
{
  neuron1_mat_counts[i, ] =  hist(Neuron_1[[i]],breaks = breaks, plot = FALSE)$counts
}

plot(seq(0.1,10.9, by=0.5), 2*apply(neuron1_mat_counts, 2, mean), type='l', main = "", ylab="firing rat
```

Beyond (homogeneous) Poisson Processes

Plot the cumulative count of spike events (for neuron 1 trial 1) as a function of the event times and overline the cumulative count of your Poisson process with parameter λ. Choose λ as the average number of events per seconds over the 20 replicates.

Looking at the real data one may realize that the hypothesis of homogeneity for our Poisson process is not verified.

The aforementioned Poisson process had a constant intensity along the time. I.e., λ does not change with t. In that sense it is said to be a **homogeneous Poisson process**. There is actually a direct generalization so-called **inhomogeneous** Poisson process for which we allow the intensity to depend on time $\lambda(t)$. This is out of the scope of this practical but some R packages provide functions to model data using such processes.

Bayesian modeling

Questions

- Using R, create 2 vectors of counts giving the number of spikes for each trial in time intervals $I_1 = (3, 4]$ and $I_2 = (5, 6]$. Denote as $y_{i,j}$ the number of spikes for the i-th trial and j-th time interval.
- You decide to use a Bayesian approach and propose the following model

$$y_{i,j}|\theta_j \sim Poi(\theta_j),\ j \in \{1,2\}$$
$$\theta_j \sim Gamma(0.1, 0.1).$$

- Discuss this modeling choice.
- Using the Rjags code chunk below, compute the posterior distribution of the differences of the Poisson parameters.
- Give the histogram of the differences.
- From the Jags outputs what are your conclusions ?

```
# create the vector of counts


# Load jags
library(rjags)
library(R2jags)

# Prepare Jags function input
model_code = '
model
{
# likelihood

# Prior

}
'

model_data = list(y1 = y1,y2 = y2,  N  = 20)


model_parameters =  c("theta1","theta2","delta")

# run Jags
model_run = jags(data = model_data,
                 parameters.to.save = model_parameters,
                 model.file=textConnection(model_code),
                 n.chains=4,
                 n.iter=10000,
                 n.burnin=200,
                 n.thin=2)

# print the results
print(model_run)

# plot the posterior distribution of the parameter of interest
hist(model_run$BUGSoutput$sims.array[, , "delta"], main="", probability = TRUE, xlab="delta")
```