Offline replay of trajectories

Balazs B Ujfalussy

20/11/2017

This is a demo for illustrating replay of place cell activity in the hippocampus. We will analyse the same 30 min recording session from a rat running back and forth in a 1.6 m ling linear track, from the laboratory of Gyuri Buzsaki as in the previous demo. There are about 120 pyramidal cells recorded simulataneously, half of them are place cells. There are also 20 interneurons.

There are two homeworks here you can choose from. You don't need to do both.

- Compare place cells in left and right runs.
- Detect offline replay event and analyse their behavioural correlates.

The source of the data: Grosmark, A.D., Long J. and Buzsáki, G (2016). Recordings from hippocampal area CA1, PRE, during and POST novel spatial learning. CRCNS.org http://dx.doi.org/10.6080/K0862DC5

Paper related to the dataset: Grosmark, A.D., and Buzsáki, G. (2016). Diversity in neural firing dynamics supports both rigid and learned hippocampal sequences. Science 351, 1440–1443.

Load the dataset and observe place cells

We will read a preprocessed data file 'Achilles.RData' - Achilles is the name of the rat. It is a list storing different variables describing the experiment.

```
load('./Achilles.RData')
```

The `summary()` function tells you what variables are encoded int the list `rat`:

```
summary(rat)
```

```
##            Length  Class  Mode
## pos         161524 -none- numeric
## spt        1731934 -none- numeric
## iruns.up        84 -none- numeric
## iruns.down      84 -none- numeric
## PyrIDs         120 -none- numeric
## IntIDs          17 -none- numeric
## MazeRange        2 -none- numeric
```

- The main variables encoded are the position (`pos`), spike times (`spt`). The position is a matrix of two columns: time in seconds and smoothed 1D position of the animal along the linear track. The spt is also a two columnn matrix, time in seconds and the ID of the cell that emitted the spike. (Cell id's refer to the elctrodes the cell was recorded from, so they do not start from 1. . .)
- Before and after each run the animal stays at the end for a while to consume reward. Up and down runs (left and right, sorry :-)) are associated with different neuronal activity, so they are treated differently. The variable `iruns.down` and `iruns.up` stores the start and the end of the individual runs indexing the rows of the matrix `pos`.
- The variable `PyrIDs` and `IntIDs` stores the name of the (putative) pyramidal cells and interneurons.
- `MazeRange` is the x coordinates associated with the start and the end of each run.

The individual variables in the list can be referred by the $ sign. For example the dimensionality of the position variable can be prompted as `dim(rat$pos)` which returns 80762, 2 meaning that this matrix has 80762 rows and 2 columns.

Now we will load a function that will analyse this data to return the spike counts for each cell on each runs in the function of the (discretised) position.

```
source("PlaceCellFunctions.R")
require(viridis)
```

```
## Loading required package: viridis
```

```
## Loading required package: viridisLite
```

```
## tha package Matrix implements an efficient representation of sparse matrices (most of the elements a
require(Matrix)
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 3.4.2
```

```
pos <- rat$pos
spt <- rat$spt
```

We define the spatial discretization in 5 cm.

```
dx <- 0.05 # cm, resolution
x.breaks <- seq(rat$MazeRange[1], rat$MazeRange[2], by=dx)
x.mids <- round(x.breaks[-1] - dx/2, 3)
```

`act.runs` is an array with the spikes of each cell on each trial with spatial resolution dx. Its dimensions are {number of cells} x {distance} x {trials}. The last neuron is not a true neuron, but stores the time (in seconds) the rat spent at each location at each trial. Now we analyse both left and right runs, to see that place cell activity depends on the diraction of motion, but later we will only look for replay events related to the left runs.

```
act.runs.left <- cell.maps.runs(spt, pos, i.runs=rat$iruns.up, dx=0.05, MazeRange=rat$MazeRange, cell.I
act.runs.right <- cell.maps.runs(spt, pos, i.runs=rat$iruns.down, dx=0.05, MazeRange=rat$MazeRange, cel
```

We divide spike count with the occupancy time to get firing rates, and plot the firing rate of all cells in the function of distance.

```
ratemaps.left.t <- apply(act.runs.left[1:120,,], c(1,2), sum)
Tmap.left <- apply(act.runs.left[121,,], 1, sum)

ratemaps.right.t <- apply(act.runs.right[1:120,,], c(1,2), sum)
Tmap.right <- apply(act.runs.right[121,,], 1, sum)

ratemaps.left.all <- t(ratemaps.left.t) / Tmap.left
ratemaps.right.all <- t(ratemaps.right.t) / Tmap.right
matplot(x.mids, ratemaps.left.all, t='l', lty=1, col=rainbow(120), xlab='x position (m)', ylab='firing
```

```
#matplot(x.mids, ratemaps.right.all, t='l', lty=1, col=rainbow(120), xlab='x position (m)', ylab='firin
```

Now we prepare a vector containing the index pf those cells that were active on the left runs and another list for the right run-related cells. Finally we prepare a third vector with cells that were active in either left or right runs.

```
i.cells.active.left <- which(apply(ratemaps.left.all, 2, max) > 5)
N.cells.active.left <- length(i.cells.active.left)
```

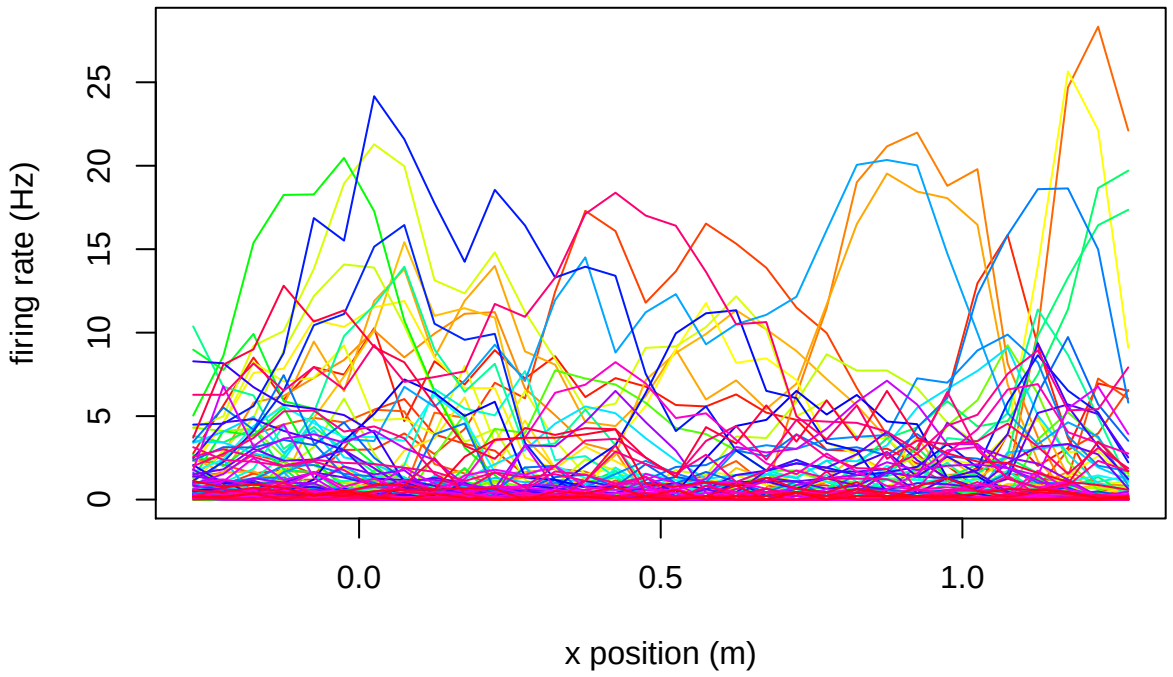


Figure 1: Ratemap of all pyramida cells.

```
i.cells.active.right <- which(apply(ratemaps.right.all, 2, max) > 5)
N.cells.active.right <- length(i.cells.active.right)


i.cells.active <- which((apply(ratemaps.left.all, 2, max) > 5) | (apply(ratemaps.right.all, 2, max) > 5)
N.cells.active <- length(i.cells.active)
```

Now we plot all place cell's activity in both left and right runs in the function of position. The different directions are plotted next to each other using different colormaps. For some cells, the place fields are similar in left and right runs, while in most cells they are different. We save the plot in the png file 'allPlaceCells.png'.

```
png(file='allPlaceCells.png', 2500, 1500)
par(mfcol=c(8,10)); par(mar=c(1,1,1,1))
for (i.cell in i.cells.active){
    image(x.mids, 1:42, act.runs.left[i.cell,,] / act.runs.left[121,,], col=viridis(24), xlab='', ylab=
    image(x.mids+1.65, 1:42, act.runs.right[i.cell,,] / act.runs.left[121,,], col=viridis(24, option='B
    lines(x.mids, ratemaps.left.all[,i.cell], col=viridis(3)[3], lwd=2)
    lines(x.mids+1.65, ratemaps.right.all[,i.cell], col=viridis(3)[3], lwd=2)
    # info.cell <- skaggs93.info(ratemaps.all[,i.cell], Tmap) # bits/sec
    # title(main=paste('info:', round(info.cell, 3), 'bit / s'))
    # readline(i.cell)
}
dev.off()
```

```
## pdf
##   2
```

```
ratemaps.left <- ratemaps.left.all[,i.cells.active]
ii.maxs.left <- apply(ratemaps.left, 2, which.max)
sort.peaks.left <- sort(ii.maxs.left, ind=T)$ix
```

```
ratemaps.right <- ratemaps.right.all[,i.cells.active]
ii.maxs.right <- apply(ratemaps.right, 2, which.max)
sort.peaks.right <- sort(ii.maxs.right, ind=T)$ix
```

```
par(mfcol=c(2,2))
image(x.mids, 1:N.cells.active, ratemaps.left[,sort.peaks.left], col=viridis(24), xlab='x position (m)'
image(x.mids, 1:N.cells.active, ratemaps.right[,sort.peaks.left], col=viridis(24), xlab='x position (m)
image(x.mids, 1:N.cells.active, ratemaps.left[,sort.peaks.right], col=viridis(24), xlab='x position (m)
image(x.mids, 1:N.cells.active, ratemaps.right[,sort.peaks.right], col=viridis(24), xlab='x position (m
```

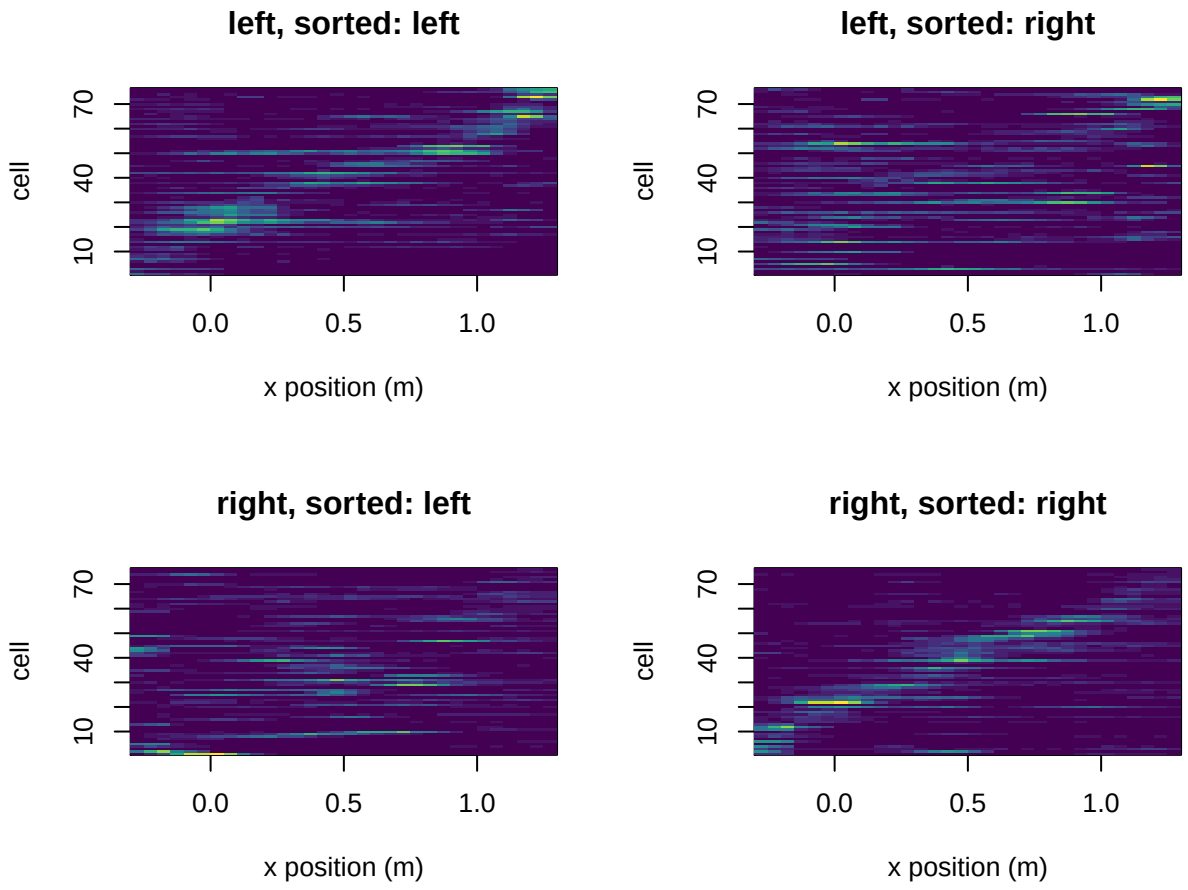


Figure 2: Place cells sorted according to their peak firing rate

Homework 1

- Calculate the distribution of correlations (`cor()`) among place fields in the two different running directions! Plot it on a histogram (`hist()`), and indicate its mean!
- As a control, shuffle the cells randomly, and calculate the correlations again. Calculate the mean of the shuffled distribution as well.
- Observe the two distributions. Are they similar or different? What does it tell you about the place representation in the two running directions? Could you decode the direction of motion from the cell's activity? Could you decode the position of the animal? (Now you don't need to decode either.)

Preplay and replay analysis

We collect all spikes in a huge matrix with columns being the time points ar 40 Hz resolution and rows being the active cells. We only focus on cells that are active during the left runs. We will detect candidate replay events by a transient increase in the population activity while the animal is immobile.

```
## collect all spikes in a big population activity matrix

i1 <- 3001 # start a bit before the first run
i2 <- 80000 # end after the last run
isp.1 <- min(which(spt[,1] > pos[i1,1]))
isp.2 <- max(which(spt[,1] < pos[i2,1]))
# spt contains all spikes with two columns: time, cell ID
# we select only points between 3001 and 80000
spt.pop.all <- spt[isp.1:isp.2,]

# next we select the spikes of the cells active during left runs
ii.act.pop <- spt.pop.all[,2] %in% rat$PyrIDs[i.cells.active.left] # index of cells active during the l
spt.pop.IDs <- spt.pop.all[ii.act.pop,]

# next we rename the cell IDs -
# recall, cells are named by the tetrodes they are recorded from
# now we give them numbers from 1 to 50 or so.
spt.pop <- spt.pop.IDs
for (i.cell in 1:N.cells.active.left){
    ii <- rat$PyrIDs[i.cells.active.left[i.cell]]
    i.sp <- which(spt.pop.IDs[,2] == ii)
    spt.pop[i.sp,2] <- i.cell
}

## time and position vectors for the population activity...
tpop <- pos[i1:i2, 1]
xpop <- pos[i1:i2, 2]
xpop <- xpop - min(xpop)
dt.pos <- mean(diff(tpop))

## the population spike matrix - SPike Train POPulation
## tha package Matrix implements an efficient representation of sparse matrices (most of the elements a
popact <- Matrix(0, N.cells.active, i2-i1+1)

## for each active cell we find its spikes and add to the population activity matrix
for (i.cell in 1:N.cells.active.left){
    t.sp <- spt.pop[which(spt.pop[,2] == i.cell),1]
    i.sp <- (t.sp - pos[i1,1]) %/% dt.pos
    for (jj in i.sp) popact[i.cell,jj] <- popact[i.cell,jj] + 1
    cat('cell', i.cell, 'done \n')
}
```

Next, we will detect candidate events in the population activity. Replay events are short (~0.1 s) periods with increased population activity. They are typical during immobility. Here we detect them using a threshold for speed and total spike count in a 100 ms window.

```
## total population activity - to detect candidate replay events
poprate <- colSums(popact)
sum(poprate)
```

```
## [1] 166938
```

```
poprate.f <- filter(poprate, rep(1, 4))

## speed of the animal
speed <- abs(diff(xpop))
speed <- c(speed[1], speed)


## candidate spw is where there are at least 40 spikes in 0.1 s and the rat is not moving
## this is somewhat conservative, but will do for the present purposes
ind.spw <- which((speed < 0.0005) & (poprate.f > 40))

## some spw-s are detected more than once. Here we remove duplicates.
ind.different.spw <- rep(T, length(ind.spw))
for (ii in 2:length(ind.spw)){
    if ((ind.spw[ii] - ind.spw[ii-1]) < 10) ind.different.spw[ii] <- F
}
ind.spw <- ind.spw[ind.different.spw]
t.spw <- tpop[ind.spw]
x.spw <- xpop[ind.spw]

## finally we will need to sort the cells according to their peak of their place field.
## this will be useful to detect order in the activity of the cells.
ratemaps.left <- ratemaps.left.all[,i.cells.active.left]
ii.maxs.left <- apply(ratemaps.left, 2, which.max)
sort.peaks.left <- sort(ii.maxs.left, ind=T)$ix
```

Finally, we will take a look on all candidate event. We will plot the activity of the cells in the function of time in a 0.5 s long window around the event. The cells are sorted and coloured according to the position of their place field, so their sequential activation should be observed in these plots as slant lines with either positive or negative slopes, depending on the direction of the replay (forward or backward). Here I selected 6 events, but the plot can also be saved into a file `ReplayEvents.png`.

Spikes are collected from the matrix `spt.pop` that contains all spikes of the active cells during the analysed session:

- For each candidate event we select a 0.5 s long interval around the event.
- These define the start and the end of the event.
- We find the corresponding spikes, stored in the vector `t.spw`.
- Then we sort the cell according to their place field location.
- Finally, we plot the cells' activities in each event.

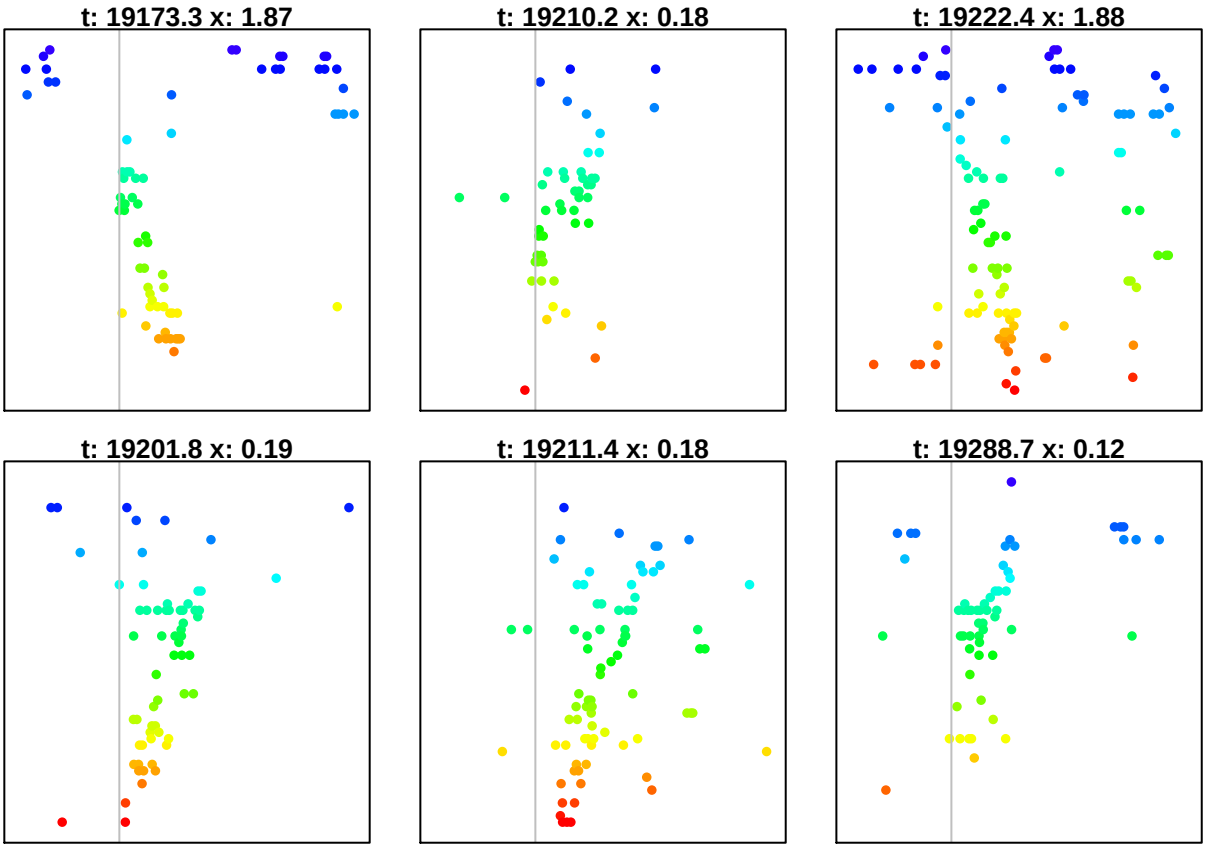
```
#png('ReplayEvents.png', 2500, 1800, pointsize=36)
#par(mfcol=c(7,8)); par(mar=c(1,1,1,1))
#for (i.spw in 1:length(ind.spw)){
par(mfcol=c(2,3)); par(mar=c(1,1,1,1))
for (i.spw in c(15:20)){
    t.start <- t.spw[i.spw] - 0.15
    t.end <- t.spw[i.spw] + 0.35

    isp.1 <- min(which(spt.pop[,1] > t.start))
    isp.2 <- max(which(spt.pop[,1] < t.end))
    spw <- spt.pop[isp.1:isp.2,]

    #Sort the cell according to their place field location.
```

```
    cells <- spw[,2]
    cells.left <- cells
    # cells.right <- cells
    cols.cells <- rep(0, length(cells))
    for (i.sp in 1:length(cells)) {
        cells.left[i.sp] <- which(sort.peaks.left == cells[i.sp])
        # cells.right[i.sp] <- which(sort.peaks.right == cells[i.sp])
        cols.cells[i.sp] <- rainbow(N.cells.active.left, end=0.7)[which(sort.peaks.left == cells[i.sp])]
    }


    # plot(spw[,1], cells.left, pch=16, col=4)
    # abline(v=t.spw[i.spw], col=grey(0.75))
    title <- paste('t:', round(t.spw[i.spw], 1), 'x:', round(x.spw[i.spw], 2))
    plot(spw[,1], cells.left, pch=16, col=cols.cells, axes=F, main=title, xlim=c(t.start, t.end), ylim=c
    abline(v=t.spw[i.spw], col=grey(0.75))
    box()
    # readline(i.spw)
}
```




```
#dev.off()
```

Homework 2

- Come up with a metric that can classify these events into three categories: forward, backward or unrelated.

- Calculate this metric for the replay events shown above. Do you observe more forward or reverse events?
- Calculate the frequency of the different categories at the left and right end of the linear track. Does the frequency of the forward and backward events depend on the position of the animal?