How to use CMetaMap

Christina Chou

2024-03-28

Custom code has been written to provide users with a GUI running in IgorPro to analyze the optomapping dataset. The raw data can also be exported in a table format, which can then be copied into a spreadsheet software then imported into other database tools.

- Note that the Igor language is not case sensitive

Install Igor Pro and custom procedure files

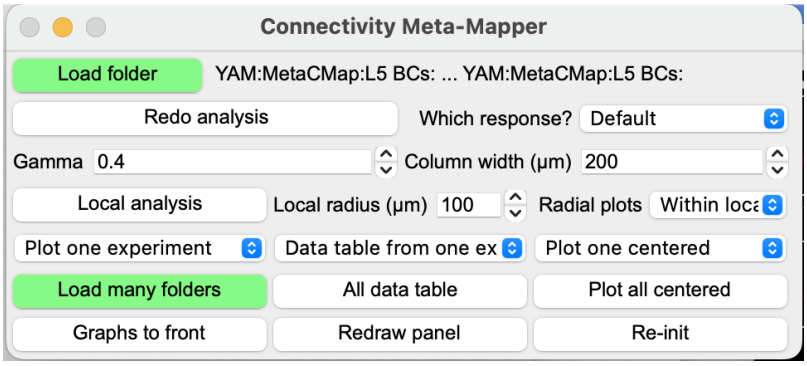
- Download demo from this address: https://www.wavemetrics.com/downloads/current/Igor%20Pro%209 (https://www.wavemetrics.com/downloads/current/Igor%20Pro%209)
- Move the file `JespersTools_V03` into DocumentsProcedures
- Unzip the folder `User Procedures` and move those files into DocumentsProcedures
- Unzip the folder `Igor Stuff` , create a new folder in your user folder named “Igor Stuff”, and move the contents in.
    - This `Igor Stuff` folder should be in the same layer as your Documents folder.
    - It might be helpful to create a shortcut in your finder sidebar for the Igor Stuff folder.

Dataset structure:

- Folders in the MetaCMap folder are organized by the postsynaptic cell type and layer location.
- In each folder, individual .itx files represent individual postsynaptic cells and contain a data frame with numerical data
    - They are named as CMap_DateOfExperiment_cellNumber.
    - Note that the date of the experiment also serves as a unique animal identifier as only 1 animal was used per experiment day.
    - Each row of the data frame represents each tested presynaptic input for that postsynaptic cell

How to use CMetaMap_v02

- Open `CMetaMap_v02.ipf` from the `Igor Stuff` folder. This should automatically open Igor.
    - Hit `Compile` on the procedure window if it does not automatically compile.
- In the top bar, select `Macros > Init Connectivity Meta-Mapper` to load the Connectivity Meta-Mapper GUI.



- Click `Load folder`, and select one of the loaded data folders, eg. “L5BCs”.
    - This will generate the ensemble heat map as well as graphs for layer-specific connectivity, EPSP amplitude, PPR, path strength, radial connectivity, radial amplitude graphs, and EPSP amplitude frequency distribution graphs.

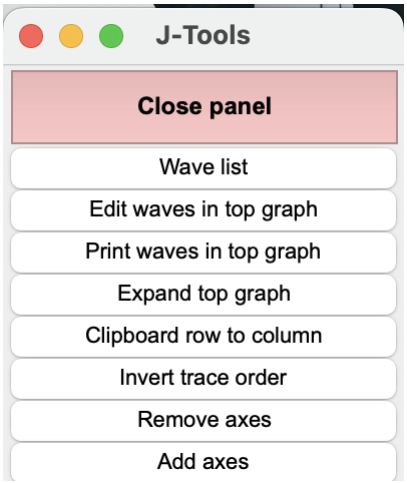
Load figures for individual postsynaptic cells

- Click the `Local analysis` button to see the connectivity within the specified `Local radius` for each postsynaptic neuron
- Click the `Plot one experiment` dropdown menu to bring up a popup window showing the connectivity map from individual postsynaptic neurons.
- In the popup window, you can click `next` or `prev` to cycle through the individual connectivity maps
- Clicking `data` will bring up additional information from that experiment such as animal age and sex

Generate color-coded matrices

- Open up the Jesper’s Tools GUI. In the command window:

```
JT()
```



- Select a bar graph and hit `Edit waves in top graph` in the Jesper’s Tools panel.
    - This will generate a table with the mean and SEM values shown in the selected graph.
    - Data from the “Layer connectivity within column”, “Amplitude within column”, and “Pathway strength” graphs were used to create the color-coded matrices in the publication. However, statistical analysis was done separately in RStudio using extracted data (see below).
- Connectivity, EPSP amplitude, and path strength matrices were first made into a spreadsheet table and an identical table was made with normalized values.

BC	post				Normalized	post		
pre	L2/3	L5	L6		pre	L2/3	L5	L6
L2/3	0.0194514	0.00184491	0.00030159		L2/3	31.9248531	3.02798796	0.49499412
L4	0.0172601	0.00731432	0.00264178		L4	28.3283526	12.0047191	4.33585338
L5	0.01135961	0.0609287	0.01174625		L5	18.6441053	99.9999984	19.2786798
L6	3.86E-05	0.0003361	0.00911022		L6	0.06330034	0.55162913	14.952259

In the Igor command window:

```
Make/O/N=(4,3) Heatmap
```

- Creates a matrix with 3 columns and 4 rows in table format

```
Edit Heatmap
```

- Brings up this table, populate it with the values from your spreadsheet

```
NewImage Heatmap
```

- Creates a color coded matrix using the values indicated in the Heatmap table. Note that the rows and columns are swapped in the color matrix.

```
ModifyGraph swapXY=1
```

- Optionally, this will swap the XY axis
- Right click on the color coded matrix and select `Modify Image` to change the color code.

Getting data for analysis in RStudio

- In the top bar, select Macros > Export data for stats in R Studio
    - This will generate a data table containing selected metadata and measurements for each tested connection within the 200-µm-wide vertical column surrounding the postsynaptic cell:

Name	Data type
Postsyn Layer	Layer location of postsynaptic cell (zero is layer 1, one is L2/3, etc)
Date	Date of experiment (YYYYMMDD)
PostSyn	Experimental cell ID of the postsynaptic cell (starts from 1 on each experiment day)
PostCellID	Overall Cell ID of the postsynaptic cell (starts from 1 for each MetaMap category)
Connected	Whether this input was connected (0 vs 1)
LayerLoc	Layer location of presynaptic cell (zero is layer 1, one is L2/3, etc)
EPSPAmpli	Measured EPSP1 amplitude
PPR	Measured paired pulse ratio EPSP2+EPSP1
TPR	Measured triple pulse ratio (mean(EPSP2+EPSP3)/EPSP1)
Sex	Sex of the animal
Age	Age of the animal
SampleID	ID number for this tested input (starts from 1 on each postsynaptic cell)
MaxDepol	Peak depolarization amplitude (not accounting for temporal summation)
MaxDepolLoc	Latency of MaxDepol (in ms)

- This table can be copied into a spreadsheet software then imported into other database tools.
- Note that you may have to change some table headings if you want to run our specific R code:

Change from	Change to
Date	Animal
Postsyn	Postsyn.Cell
PostCellID	Cell.ID.Within
LayerLoc	Layer.Loc
EPSPAmpli	EPSP.ampli
SampleID	Sample.ID.Within
MaxDepolLoc	Max.Depol.Latency

- To use our specific R code, save each table as a separate sheet in Excel

Getting other data from MetaMap files

*First, print the list of postsynaptic cell files in this analysis folder In the Igor command window

```
print CMM_explist
```

*To access any source data type, for example:

```
edit CMap_20220118_01_1_MaxDepolLoc, CMap_20220118_01_1_UseAmp, CMap_20220118_01_1_Colum
nLoc
```

The edit function will generate a table of the specified data types Please see the CMetaMap_v02 procedure file (line 83) for the types of source data and their names. Using the edit function, one can create a customized spreadsheet with one's own data of interest.

Statistical analysis in RStudio

Loading data into RStudio

```
library(dplyr)
library(lme4)
library(car)
library(nlme)
library(lmerTest)
library(afex)
library(MASS)
library(emmeans)
library(MCMCglmm)
library(pbkrtest)
library(purrr)
library(readxl)
```

Load the data from the previous steps in Igor using the read_excel() function

```
# your data should have the column headers: "Type", "Postsyn.Layer", "Animal",
#   "Postsyn.Cell", "Cell.ID.Within", "Connected", "Layer.Loc", "EPSP.ampli",
#   "PPR", "TPR", "Sex", "Age", "Sample.ID.Within", "MaxDepol", "Max.Depol.Latency"
# if getting data tables from Igor, add a column for "Type" and add " Within" to
#   the column names "Cell ID" and "Sample ID", and change "MaxDepolLoc" to
#   "Max.Depol.Latency"
# Load each sheet as a separate dataframe
L23PCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L23PCs")
L23BCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L23BCs")
L23MCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L23MCs")
L5PCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L5PCs")
L5BCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L5BCs")
L5MCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L5MCs")
L6PCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L6PCs")
L6BCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L6BCs")
L6MCs <- read_excel("YourFilePath/Folder/Folder/File.xlsx", sheet = "L6MCs")
```

The data frames require further cleaning up

```
# create a combined list of data frames (df's) to clean up together
data.list <- list(L23PCs=L23PCs, L5PCs=L5PCs, L6PCs=L6PCs, L23BCs=L23BCs,
                  L5BCs=L5BCs, L6BCs=L6BCs, L23MCs=L23MCs, L5MCs=L5MCs, L6MCs=L6MCs)

# function to add columns for area, prevent repeats for Cell ID and Sample ID,
# log EPSP amplitude, and log PPR
add_columns <- function(df) {
  df <- df %>%
    mutate("area" = "V1") %>%
    mutate("Cell.ID" = Cell.ID.Within + x) %>%
    mutate("Sample.ID" = Sample.ID.Within + y) %>%
    mutate("log.EPSP.ampli" = log10(EPSP.ampli)) %>%
    mutate("log.PPR" = log10(PPR)) %>%
    mutate("log.TPR" = log10(TPR))
}
# Since Cell.ID was counted up within each category and Sample.ID was counted up
# within each postsynaptic cell, use a for loop to apply add_columns function to
# count up Cell.ID and Sample.ID across df's
# Cell.ID will be included as a random effect in our models
for (i in 1:length(data.list)) {
  if(i == 1){
    x = 0
    y = 0
  } else {
    j = i-1
    x = max(data.list[[j]]$Cell.ID)
    y = max(data.list[[j]]$Sample.ID)
  }
   data.list[[i]] = add_columns(data.list[[i]])
}

# Change datatype to factor for: Type, Postsyn.Layer, Animal, Layer.Loc, Sex, area,
#   Cell.ID, Sample.ID; and change to numeric for Connected; change Date to Animal
change_datatype <- function(df){
  df <- df %>% mutate(across(c("Type","Postsyn.Layer","Animal","Layer.Loc",
                               "Sex","area","Cell.ID","Sample.ID"),
                             as.factor))%>%
    mutate(across(Connected, as.numeric))
}

# apply change_datatype on df's in the data.list
# note that these changes do not apply to the original CSV loaded df's
data.list <- lapply(data.list, change_datatype)

# concatenate df's by postsyn layer
L23 <- do.call("rbind", data.list[c(grep("L23",names(data.list)))])
L5 <- do.call("rbind", data.list[c(grep("L5",names(data.list)))])
L6 <- do.call("rbind", data.list[c(grep("L6",names(data.list)))])
# concatenate the entire list and save as CSV
allData <- bind_rows(data.list)
write.csv(allData, "/Users/User/YourFilePath/Folder/Folder/File.csv", row.names = FALSE)
```

```
# separate list back into individual df's
list2env(data.list,.GlobalEnv)
```

Analyzing connectivity in RStudio

Generalized linear mixed models were used to analyze connectivity as the data had a binomial error structure (connected vs unconnected). The entire dataset could be analyzed at the same time, but given the 3x4x3 structure, the results were quite unwieldy. Therefore, we separated data by postsynaptic cell layer location and analyzed via a 3x4 structure.

```
L23glmer<-afex::mixed(Connected ~ Layer.Loc * Type + (1|Animal/Cell.ID),
                      data= L23, family= binomial, method="LRT", control=
                      glmerControl(optimizer="bobyqa", optCtrl=list(maxfun=2e5)))
summary(L23glmer)
anova(L23glmer)
L23emm<- emmeans(L23glmer,~Layer.Loc+Type:Layer.Loc)
pairs(L23emm, simple="each")

L5glmer<-afex::mixed(Connected ~ Layer.Loc * Type + (1|Cell.ID),
                     data= L5, family= binomial, method="LRT", control=
                     glmerControl(optimizer="bobyqa", optCtrl=list(maxfun=2e5)))
summary(L5glmer)
anova(L5glmer)
L5emm<- emmeans(L5glmer,~Layer.Loc+Type:Layer.Loc)
pairs(L5emm, simple="each")

L6glmer<-afex::mixed(Connected ~ Layer.Loc * Type + (1|Animal/Cell.ID),
                     data= L6, family= binomial, method="LRT", control=
                       glmerControl(optimizer="bobyqa", optCtrl=list(maxfun=2e5)))
summary(L6glmer)
anova(L6glmer)
L6emm<- emmeans(L6glmer,~Layer.Loc+Type:Layer.Loc)
pairs(L6emm, simple="each")
```

Analyzing EPSP amplitude in RStudio

Linear mixed models were used to analyze connectivity. Since EPSP amplitudes distributed log normally, we used the logged EPSP amplitudes in our models.

```
L23nlme<-lme(log.EPSP.ampli~Layer.Loc*Type, random=~1|Animal/Cell.ID,
             data=L23 %>% filter(Connected=="1", Layer.Loc !="4"))
summary(L23nlme)
anova(L23nlme)
emmeans(L23nlme, pairwise~ Type)
L23AmpEmm <- emmeans(L23nlme, ~Type:Layer.Loc)
pairs(L23AmpEmm, simple = "each")

L5nlme<-lme(log.EPSP.ampli~Layer.Loc*Type, random=~1|Animal/Cell.ID,
             data=L5 %>% filter(Connected=="1"))
summary(L5nlme)
anova(L5nlme)
L5AmpEmm <- emmeans(L5nlme, ~Type:Layer.Loc)
pairs(L5AmpEmm, simple = "each")

L6nlme<-lme(log.EPSP.ampli~Layer.Loc*Type, random=~1|Animal/Cell.ID,
            data=L6 %>% filter(Connected=="1", Layer.Loc !="1"))
summary(L6nlme)
anova(L6nlme)
emmeans(L6nlme, pairwise~ Type)
L6AmpEmm <- emmeans(L6nlme, ~Type:Layer.Loc)
pairs(L6AmpEmm, simple = "each")
```

In the L2/3 and L6 analyses, we had to exclude Layer.Loc == “4” and Layer.Loc == “1”, respectively. This was because there were not a sufficient number of PC->MCs data points for these connection types.

One could break down the data by postsynaptic layer and cell type to include L2/3->L6 and L6->L2/3 connections for PC->PC and PC->BC connections

```
L23PCnlme<-lme(log.EPSP.ampli~Layer.Loc, random=~1|Animal/Cell.ID, data=L23PCs %>% filte
r(Connected =="1"))
summary(L23PCnlme)
anova(L23PCnlme)
#anova was not significant so pairwise comparisons were not conducted
```

Analyzing PPR in RStudio

PPR measurements were found to differ based on cell type and presynaptic layer location, but not postsynaptic layer location.

```
#model on presyn layer and type
PPRnlme<-lme(log.PPR~Type*Layer.Loc, random=~1|Animal/Cell.ID,
                data=allData %>% filter(!is.na(allData$TPR)))
summary(PPRnlme)
anova(PPRnlme)
PPREmm <- emmeans(PPRnlme, ~Type:Layer.Loc)
pairs(PPREmm, simple = "each")

#model on postsyn layer and type
PPRnlme<-lme(log.PPR~Type*Postsyn.Layer, random=~1|Animal/Cell.ID,
             data=allData %>% filter(!is.na(allData$TPR)))
summary(PPRnlme)
anova(PPRnlme)
PPREmm <- emmeans(PPRnlme, ~Type:Postsyn.Layer)
pairs(PPREmm, simple = "each")
```

To display PPR based on presyn layer and type:

```
#Make data frame for PPR based on type and presyn layer only
pre <- c(1,2,3,4,1,2,3,4,1,2,3,4) #each presyn layer inputs to 3 postsyn layers
type <- c(rep("PC", length.out = 4), rep("BC", length.out = 4), rep("MC",
      length.out =4)) #add column for type, repeat 12 times for each pre/post combo

PPRDF <- data.frame(
  Pre = pre,
  Type = type
)
vector_PPR2 <- numeric(0)
vector_PPR.SEM2 <- numeric (0)
vector_PPR.N2 <- numeric (0)

for (i in 1:nrow(PPRDF)) {
  toFilterPre <- PPRDF[i,"Pre"]
  toFilterType <- PPRDF[i,"Type"]
  #subset data of current ensemble in for loop
  pprSub2 <- c(allData$PPR[!is.na(allData$PPR)& allData$Layer.Loc == toFilterPre
                           & allData$Type == toFilterType])
  pprN2 <- length(pprSub2)
  ppr2 <- mean(pprSub2)
  #to calculate sem, divide SD by sqrt(n observations)
  pprsem2 <- sd(pprSub2) / sqrt(pprN2)
  vector_PPR2 <- c(vector_PPR2, ppr2)
  vector_PPR.SEM2 <- c(vector_PPR.SEM2, pprsem2)
  vector_PPR.N2 <- c(vector_PPR.N2, pprN2)
  print(vector_PPR2)
}
PPRDF$PPR <- vector_PPR2
PPRDF$PPR.SEM <- vector_PPR.SEM2
PPRDF$PPR.N <- vector_PPR.N2
write.csv(PPRDF,"FilePath.csv", row.names = FALSE)
```