concordance=TRUE

Applications of GFPOP

Toby Dylan Hocking

February 18, 2020

1 Multi-state model for QRS complex detection in electrocardiogram (ECG) data

```
data(ECG, package="gfpop")
arg.list <- list(
  list("b", "Q",      "down", 8000000, gap=0),
  list("Q",       "R",       "up",    0,         gap=2000),
  list("R",       "S",       "down", 0,         gap=5000),
  list("S",       "x1",      "up",    0,         gap=2000),
  list("x1",      "x2",      "up",    0,         gap=1000),
  list("x2",      "x3",      "up",    0,         gap=0),
  list("x3",      "x4",      "up",    0,         gap=0,  model=1),
  list("x4",      "y1",    "up",    0,         gap=0,  model=1),
  list("y1",    "y2",  "down", 0,         gap=0),
  list("y2",   "y3", "down", 0,         gap=0),
  list("y3",  "y4", "down", 0,         gap=0, model=1),
  list("y4",  "y5",   "down",   0,        gap=0, model=1),
  list("y5",    "b","up",   0,        gap=0))
library(data.table)
m <- function(x){
  factor(x, c("previous", "changepoint"), c(
    "Previous model", "Proposed models"))
}
model.dt.list <- list(
  previous=ECG$PanTompkins[, data.table(
    n.states=NA,
    states=NA,
    model=m("previous"),
    time,
    millivolts,
    state=letter
  )])
mean.dt.list <- list()
select.vec.list <- list(
  less=sapply(arg.list, function(L)is.null(L$model)),
  more=rep(TRUE, length(arg.list)))
for(states in names(select.vec.list)){
  select.vec <- select.vec.list[[states]]
  edge.list <- lapply(arg.list[select.vec], function(L){
    do.call(gfpop::Edge, L[names(L)!="model"])
  })
  prev <- edge.list[[1]]$state1
  for(edge.i in seq_along(edge.list)){
    if(edge.list[[edge.i]]$state1 != prev){
      edge.list[[edge.i]]$state1 <- prev
    }
    prev <- edge.list[[edge.i]]$state2
  }
  myGraph <- do.call(gfpop::graph, edge.list)
  fit <- gfpop::gfpop(ECG$data$millivolts, mygraph = myGraph, type = "mean")
  end.i <- fit$changepoints
  start.i <- c(1, end.i[-length(end.i)]+1)
  n.states <- length(unique(fit$states))
  segments.dt <- with(ECG$data, data.table(
    timeStart=time[start.i],
    timeEnd=time[end.i],
    state=fit$states,
    mean=fit$parameters))
  ## to show vertical lines connecting the horizontal segment means:
  mean.dt.list[[states]] <- segments.dt[, data.table(
    n.states,
    states,
    time=as.numeric(rbind(timeStart-0.5, timeEnd+0.5)),
    mean=as.numeric(rbind(mean, mean)))]
  model.dt.list[[states]] <- segments.dt[, data.table(
    n.states,
    states,
    model=m("changepoint"),
    time=ifelse(state=="Q", timeEnd, (timeStart+timeEnd)/2),
    millivolts=mean,
    state)]
}
```

```
#> Error in explore(mynewgraph):  Not all path have a recursive edge
```

```
cat(tikz.tex <- data.table(myGraph)[type != "null", paste(c("
\\definecolor{deepskyblue}{RGB}{0,191,255}
\\begin{tikzpicture}[->,>=latex,shorten >=1pt,auto,node distance=1.4cm,
      thick,main node/.style={circle,draw}]
", sprintf(
  "\\node[main node, fill=%s, text=blue] (%s) %s {%s};\n",
  ifelse(select.vec.list$less, "white!50!deepskyblue", "white"),
  state1,
  ifelse(state2=="Q", "", paste0("[right of=", c(NA, state1[-.N]), "]")),
  state1), "
\\path[every node/.style={font=\\sffamily\\small}]
", sprintf(
  "(%s) edge [bend left%s] node [above] {%s} node [below] {\\scriptsize %s} (%s)\n",
  state1, ifelse(
    state2=="b", ", looseness=0.4", ""),
  sprintf(
    "$%s, %s$",
    ifelse(
      penalty==0, 0, "\\lambda"),
    ifelse(
      type=="up", "\\uparrow", "\\downarrow")),
  ifelse(parameter==0, "", parameter/1e3),
  state2), ";
\\end{tikzpicture}
"), sep="\n")])

#> Error in sprintf("\\node[main node, fill=%s, text=blue] (%s) %s {%s};\n", :  arguments
cannot be recycled to the same length
```

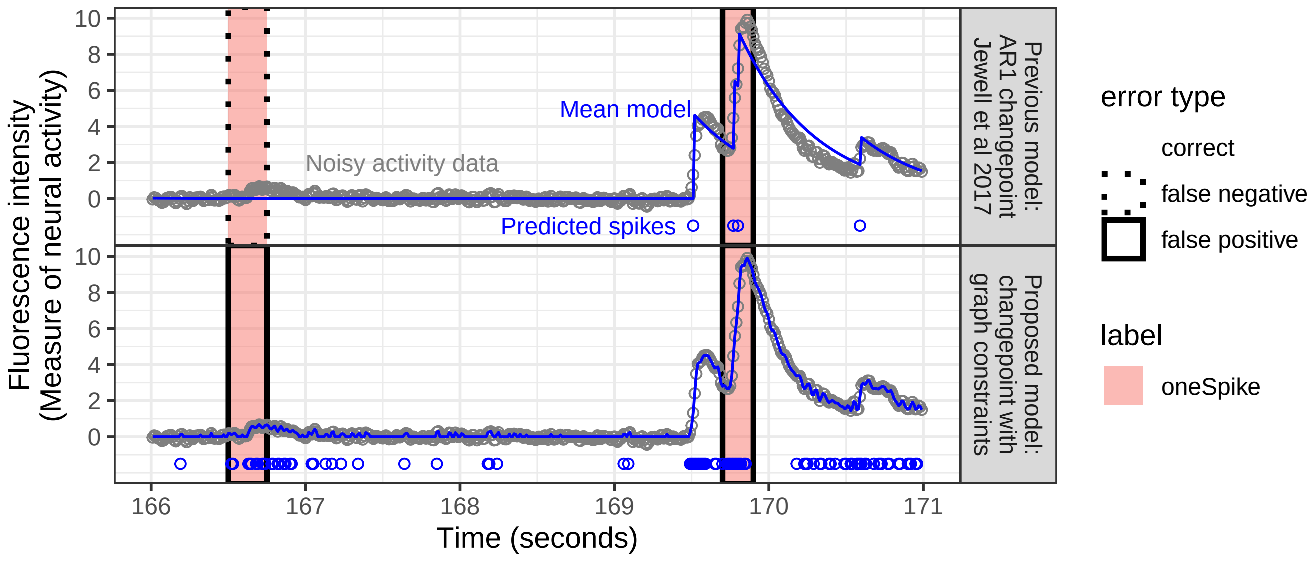
```
(model.dt <- do.call(rbind, model.dt.list)[state %in% c("Q", "R", "S")])

#>     n.states states          model  time millivolts state
#>  1:       NA     NA Previous model 52125       2002     Q
#>  2:       NA     NA Previous model 52335       2464     Q
#>  3:       NA     NA Previous model 52551       2002     Q
#>  4:       NA     NA Previous model 52766       3035     Q
#>  5:       NA     NA Previous model 52974       3383     Q
#>  6:       NA     NA Previous model 52128       3566     R
#>  7:       NA     NA Previous model 52340       4045     R
#>  8:       NA     NA Previous model 52553       3696     R
#>  9:       NA     NA Previous model 52769       4774     R
#> 10:       NA     NA Previous model 52976       4266     R
#> 11:       NA     NA Previous model 52130       5895     S
#> 12:       NA     NA Previous model 52349      -3080     S
#> 13:       NA     NA Previous model 52555       5851     S
#> 14:       NA     NA Previous model 52778      -2533     S
#> 15:       NA     NA Previous model 52978       6523     S

mean.dt <- do.call(rbind, mean.dt.list)
mean.dt[, States := factor(n.states, c("9", "13"))]
```

```
#> Error in `:=`(States, factor(n.states, c("9", "13"))):  Check that is.data.table(DT)
== TRUE. Otherwise, := and `:=`(...)  are defined for use in j, once only and in particular
ways.  See help(":=").

samples.per.second <- 250
truth.dt <- segments.dt[state=="R", list(time=(timeStart+timeEnd)/2)]

#> Error in eval(expr, envir, enclos):  object 'segments.dt' not found

library(ggplot2)
gg <- ggplot()+
  geom_vline(aes(
    xintercept=time/samples.per.second),
    color="red",
    data=truth.dt)+
  geom_text(aes(
    x, y/1e3, hjust=hjust, label="True R"),
    color="red",
    size=3,
    data=data.table(
      x=208.5, y=6500, hjust=1, label="True R", model=m("changepoint")))+
  theme_bw()+
  theme(panel.spacing=grid::unit(0, "lines"))+
  facet_grid(model ~ .)+
  geom_line(aes(
    time/samples.per.second, millivolts/1e3),
    color="grey50",
    data=ECG$data)+
  geom_line(aes(
    time/samples.per.second, mean/1e3, color=States, size=States),
    data=data.table(model=m("changepoint"), mean.dt),
    alpha=0.5
    )+
  scale_color_manual(values=c(
    "13"="blue",
    "9"="deepskyblue"))+
  scale_size_manual(values=c(
    "13"=0.8,
    "9"=1.5))+
  geom_label(aes(
    time/samples.per.second, millivolts/1e3,
    label=state),
    color="blue",
    size=3,
    label.padding=grid::unit(0.1, "lines"),
    alpha=0.6,
    data=model.dt)+
```

```r
  coord_cartesian(xlim=c(52000, 52900)/samples.per.second, expand=FALSE)+
  xlab("Time (seconds)")+
  ylab("Electrocardiogram activity (Volts)")
#> Error in fortify(data):  object 'truth.dt' not found
##png("figure-ecg.png", 7, 2.5, res=300, units="in")
print(gg)
#> Error in print(gg):  object 'gg' not found
##dev.off()
```

2 Multi-modal regression for neuro spike train data set

```r
library(data.table)
data(neuroSpike, package="gfpop")
fps <- 100
seconds.between.data <- 1/fps
sec.w <- seconds.between.data/3
myGraph <- gfpop::graph(
  gfpop::Edge("goingDown", "goingDown", "down", 0),
  gfpop::Edge("goingUp", "goingUp", "up",   0),
  gfpop::Edge("goingDown", "goingUp", "up",   5),
  gfpop::Edge("goingUp", "goingDown", "down", 0))
fit <- gfpop::gfpop(
  neuroSpike$calcium, mygraph = myGraph, type = "mean")
end.i <- fit$changepoints
start.i <- c(1, end.i[-length(end.i)]+1)
res.dt <- with(neuroSpike, data.table(
  start.seconds=seconds[start.i],
  end.seconds=seconds[end.i],
  state=fit$states,
  mean=fit$parameters))
res.dt[, Multimodal := ifelse(mean<0.1, 0, mean) ]
over.dt <- res.dt[neuroSpike, on=list(
  start.seconds <= seconds, end.seconds >= seconds)]
tall.dt <- melt(
  over.dt,
  measure.vars=c("Multimodal", "AR1"),
  variable.name="model")
tall.dt[, change.after := c(diff(value), NA), by=model]
tall.dt[, seconds.after := c(start.seconds[-1], NA), by=model]
tall.dt[, spike.i := cumsum(change.after > 0), by=model]
tall.dt[, thresh := ifelse(model=="Multimodal", 0, 0)]
m <- function(model){
  factor(
    model,
    c("AR1", "Multimodal"),
    c("Previous model:
AR1 changepoint
Jewell et al 2017", "Proposed model:
changepoint with
graph constraints"))
}
tall.dt[, model.fac := m(model)]
spike.dt <- tall.dt[0 < change.after & thresh < value, list(
  start.seconds=min(start.seconds),
  end.seconds=max(start.seconds)
), by=list(spike.i, model.fac)]
spike.dt[, mid.seconds := (start.seconds+end.seconds)/2]
library(ggplot2)
lab <- function(xmin, xmax){
  data.table(xmin, xmax, label="oneSpike", annotation="1change", problem=1)
}
label.dt <- rbind(
  lab(166.5, 166.75),
  lab(169.7, 169.9))
ann.colors <- c(oneSpike="#ffafaf")
xmin <- 166
xmax <- 171
spike.dt[, problem := 1]
models.dt <- spike.dt[, list(
  models=.N
  ), by=list(model.fac, problem)]
err.list <- penaltyLearning::labelError(
  models.dt, label.dt, spike.dt,
  change.var="mid.seconds",
  problem.var="problem",
  label.vars=c("xmin", "xmax"),
  model.vars="model.fac")
type.colors <- c(
  data="grey50",
  model="blue")
show.spikes <- spike.dt[xmin < mid.seconds & mid.seconds < xmax]
show.data <- neuroSpike[xmin < seconds & seconds < xmax]
spike.y <- -1.5
gg.out <- ggplot()+
  theme_bw()+
  theme(panel.spacing=grid::unit(0, "lines"))+
  facet_grid(model.fac ~ .)+
  penaltyLearning::geom_tallrect(aes(
    xmin=xmin, xmax=xmax, fill=label),
    color=NA,
    data=label.dt)+
  penaltyLearning::geom_tallrect(aes(
    xmin=xmin, xmax=xmax, linetype=status),
    fill=NA,
    color="black",
    data=err.list$label.errors)+
  scale_linetype_manual(
    "error type",
    limits=c("correct",
             "false negative",
             "false positive"),
    values=c(correct=0,
             "false negative"=3,
             "false positive"=1))+
  geom_point(aes(
    seconds, calcium, color=type),
    shape=1,
    data=data.table(type="data", show.data))+
  geom_line(aes(
    start.seconds, value, color=type),
    data=data.table(
      type="model",
      tall.dt[xmin < start.seconds & start.seconds < xmax]),
    size=0.5)+
  scale_y_continuous(
    "Fluorescence intensity
(Measure of neural activity)",
    breaks=seq(0, 10, by=2),
    limits=c(-2, 10)
  )+
  scale_x_continuous("Time (seconds)")+
  guides(color="none")+
  scale_color_manual(values=type.colors)+
  geom_text(aes(
    x,y,label=label, color=type, hjust=hjust),
    size=3,
    data=data.table(model.fac=m("AR1"), rbind(
      data.table(
        hjust=0, x=167, y=2, label="Noisy activity data", type="data"),
      data.table(
        hjust=1, x=169.5, y=5, label="Mean model", type="model"),
      data.table(
        hjust=1, x=169.4, y=spike.y, label="Predicted spikes", type="model"))))
if(nrow(show.spikes)){
  gg.out <- gg.out+
    geom_point(aes(
      mid.seconds, spike.y, color=type),
      shape=1,
      data=data.table(type="model", show.spikes))
}
##png("figure-AR1-multimodal.png", 7, 2.5, units="in", res=300)
print(gg.out)
```



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

3 Absolute value constraint for DNA copy number data set

```
library(data.table)
data(profile614chr2, package="gfpop")
profile614chr2$probes[, change.after := floor(
  c(diff(position)/2+position[-.N], NA)) ]
sngraph <- function(n.segs, type, gap){
  stopifnot(is.integer(n.segs), length(n.segs)==1, n.segs >= 1)
  s <- n.segs-1
  seg.vec <- 1:s
  gfpop::graph(
    gfpop::StartEnd(start=0, end=s),
    gfpop::Edge(seg.vec-1, seg.vec, type, gap=gap))
}
```

```
select.dt <- rbind(
  data.table(n.segments=c(3, 7, 13), graph.name="std", gap=0),
  data.table(n.segments=13, graph.name="abs", gap=1))
seg.dt.list <- list()
for(model.i in 1:nrow(select.dt)){
  model.info <- select.dt[model.i]
  cat(sprintf("%4d / %4d models\n", model.i, nrow(select.dt)))
  g <- model.info[, sngraph(as.integer(n.segments), graph.name, gap=gap)]
  fit <- gfpop::gfpop(
    profile614chr2$probes$logratio,
    mygraph = g, type = "mean")
  end.i <- fit$changepoints
  change.i <- end.i[-length(end.i)]
  change.pos <- profile614chr2$probes$change.after[change.i]
  seg.dt.list[[model.i]] <- data.table(
    model.info,
    segStart=c(profile614chr2$probes[1, position], change.pos),
    segEnd=c(change.pos, profile614chr2$probes[.N, position]),
    mean=fit$parameters)
}

#>    1 /    4 models

#> Error in explore(mynewgraph):  Not all path have a recursive edge

seg.dt <- do.call(rbind, seg.dt.list)

some.segs <- seg.dt[select.dt, on=list(n.segments, graph.name)]

#> Error in eval(expr, envir, enclos):  object 'n.segments' not found

some.change <- some.segs[min(segStart) < segStart]

#> Error in eval(expr, envir, enclos):  object 'some.segs' not found

some.change[, change := segStart]

#> Error in eval(expr, envir, enclos):  object 'some.change' not found


err.dt <- some.segs[, {
  change.dt <- .SD[min(segStart) < segStart]
  change.dt[, change := segStart]
  change.dt[, prob := 1]
  change.dt[, nseg := n.segments]
  model.dt <- data.table(nseg=n.segments, prob=1)
  penaltyLearning::labelError(
    model.dt,
    data.table(prob=1, profile614chr2$labels),
    change.dt,
    change.var="change",
    model.vars="nseg",
    label.vars=c("labelStart", "labelEnd"),
    problem.vars="prob")$label.errors
}, by=list(graph.name, n.segments)]
```

```
#> Error in eval(expr, envir, enclos):  object 'some.segs' not found
```

```
err.dt[, list(
  fp=sum(fp),
  fn=sum(fn),
  errors=sum(fp+fn)
  ), by=list(graph.name, n.segments)]
```

```
#> Error in eval(expr, envir, enclos):  object 'err.dt' not found
```

```
win <- function(min,max)data.table(windowStart=min*1e5,windowEnd=max*1e5)
windows <- rbind(
  win(  65,  71),
  win( 148, 171),
  win( 354, 361),
  win(1059,1065))
mb.fac <- 1e6
wfac <- function(x){
  factor(x, c("6.5-7.1", "14.8-17.1", "35.4-36.1", "105.9-106.5"))
}
windows[, window := wfac(sprintf(
  "%.1f-%.1f", windowStart/mb.fac, windowEnd/mb.fac))]
setkey(windows, windowStart, windowEnd)
f <- function(dt, key.vec){
  setkeyv(dt, key.vec)
  dt
}
profile614chr2$probes[, pos0 := position]
over.list <- list(
  changes=f(some.change, c("change", "segStart")),
  segments=f(some.segs, c("segStart", "segEnd")),
  labels=f(profile614chr2$labels, c("labelStart", "labelEnd")),
  errors=f(err.dt, c("labelStart", "labelEnd")),
  probes=f(profile614chr2$probes, c("position", "pos0")))
```

```
#> Error in is.data.table(x):  object 'some.change' not found
```

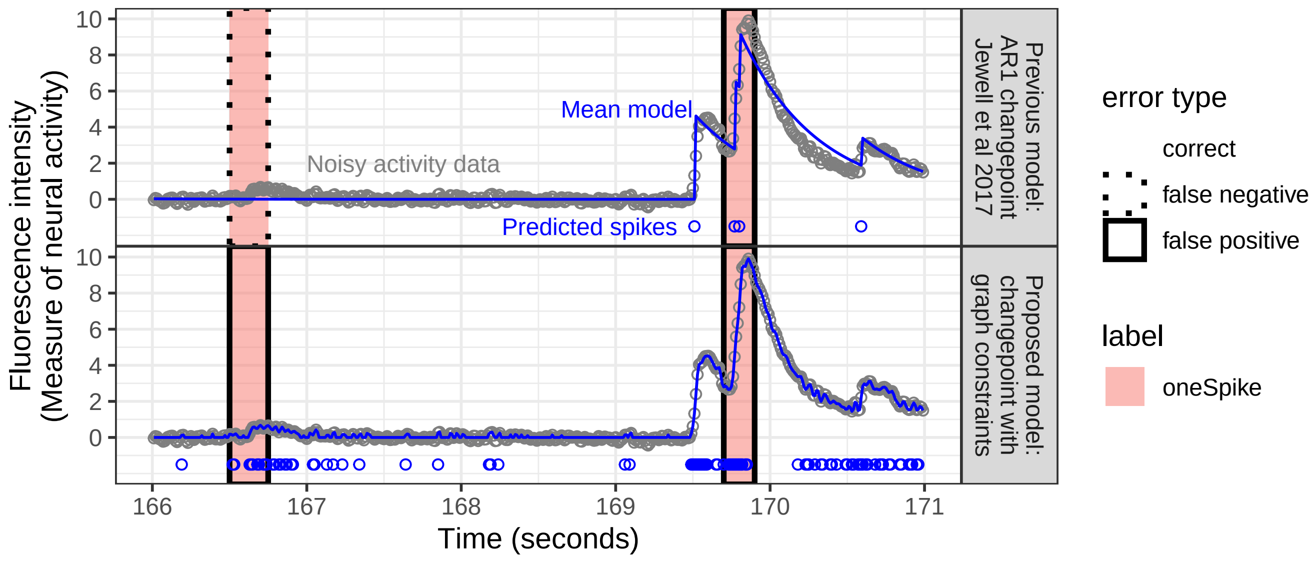
```
join.list <- lapply(over.list, foverlaps, windows, nomatch=0L)
```

```
#> Error in lapply(over.list, foverlaps, windows, nomatch = 0L): object 'over.list' not
found
```

```
show.err <- join.list$errors[, list(
  fp=sum(fp),
  fn=sum(fn),
  errors=sum(fp+fn)
  ), by=list(graph.name, n.segments)]
```

```
#> Error in eval(expr, envir, enclos):  object 'join.list' not found
```

```
library(ggplot2)
br <- c(6.5,7.0,seq(15,17,by=0.5),35.5,36,106,106.5)
names(br) <- as.character(br)
gg.out <- ggplot()+
  theme_bw()+
  theme(panel.spacing=grid::unit(0, "lines"))+
  coord_cartesian(expand=FALSE)+
  facet_grid(
    graph.name + n.segments ~ window,
    ##labeller=label_both,
    scales="free",space="free")+
  penaltyLearning::geom_tallrect(aes(
    xmin=labelStart/mb.fac, xmax=labelEnd/mb.fac, fill=annotation),
    color="grey",
    data=join.list$labels)+
  scale_fill_manual(
    "label",
    values=penaltyLearning::change.colors)+
  penaltyLearning::geom_tallrect(aes(
    xmin=labelStart/mb.fac, xmax=labelEnd/mb.fac, linetype=status),
    fill=NA,
    size=1,
    color="black",
    data=join.list$errors)+
  scale_linetype_manual(
    "error type",
    limits=c("correct",
             "false negative",
             "false positive"),
    values=c(correct=0,
             "false negative"=3,
             "false positive"=1))+
  geom_point(aes(
    position/mb.fac, logratio),
    shape=1,
    color="grey50",
    data=join.list$probes)+
  scale_y_continuous(
    "Logratio (Measure of DNA copy number)",
    breaks=seq(-2, 2, by=2)
  )+
  scale_x_continuous("Position on chr2 (mega bases)", breaks=br)+
  scale_color_manual(values=c(
    std="green",
    absSup="deepskyblue"))+
  geom_segment(aes(
    ifelse(segStart < windowStart, windowStart, segStart)/mb.fac, mean,
    color=graph.name,
    xend=ifelse(windowEnd < segEnd, windowEnd, segEnd)/mb.fac, yend=mean),
    data=join.list$segments,
    size=1)+
  geom_vline(aes(
    xintercept=change/mb.fac, color=graph.name),
    linetype="dashed",
    size=0.75,
    data=join.list$changes)+
  geom_text(aes(
    14.8, -3, label=sprintf(
      " %d label error%s for %d segment %s model",
      errors, ifelse(errors==1, "", "s"), n.segments, graph.name)),
    hjust=0,
    vjust=0,
    data=data.table(show.err, window=wfac("14.8-17.1")))

#> Error in fortify(data):  object 'join.list' not found

##png("figure-relevant-changes-copy-number.png", 8, 4, units="in", res=300, type="cairo")
print(gg.out)
```



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