Single_cells_simulation_with_LIMsolve.R

martijn

Thu Nov 28 13:51:17 2019

```
options(stringsAsFactors = FALSE)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(spdep)
```

```
## Loading required package: sp
```

```
## Loading required package: spData
```

```
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source'))`
```

```
## Loading required package: sf
```

```
## Linking to GEOS 3.6.2, GDAL 2.3.0, PROJ 5.0.1
```

```
library(limSolve)
```

```
##
## Attaching package: 'limSolve'
```

```
## The following object is masked from 'package:ggplot2':
##
##     resolution
```

```
library(reshape2)
library(gpclib)
```

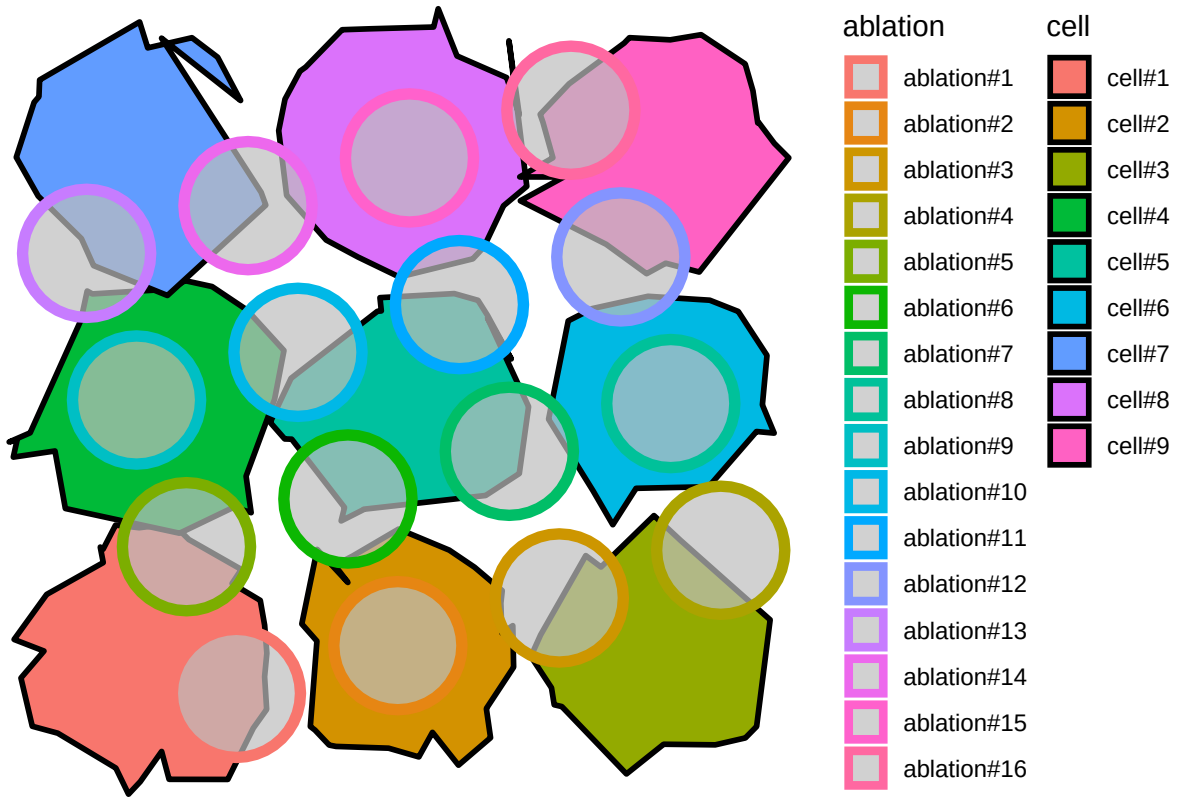
```
## General Polygon Clipper Library for R (version 1.5-5)
##  Type 'class ? gpc.poly' for help
```

```
### functions
rotate_xy <- function(xy, angle, center = c(0, 0)) {
  xRot = center[1] + cos(angle) * (xy[, 1] - center[1]) - sin(angle) * (xy[, 2] - center[2])
  yRot = center[2] + sin(angle) * (xy[, 1] - center[1]) + cos(angle) * (xy[, 2] - center[2])
  cbind(xRot, yRot)
}


convex.poly <- function(nSides, area)
```

```
{
  radius <- sqrt((2 * area) / (nSides * sin((2 * pi) / nSides)))
  angle <- (2 * pi) / nSides

  radii <- rnorm(nSides, radius, radius / 10)
  angles <- rnorm(nSides, angle, angle / 10) * 1:nSides
  angles <- sort(angles)

  points <- list(x = NULL, y = NULL)
  points$x <- cos(angles) * radii
  points$y <- sin(angles) * radii

  m <- matrix(unlist(points), ncol = 2)
  m <- rbind(m, m[1, ])
  current.area <-
    0.5 * (sum(m[1:nSides, 1] * m[2:(nSides + 1), 2]) - sum(m[1:nSides, 2] *
                                                              m[2:(nSides + 1), 1]))

  points$x <- points$x * sqrt(area / current.area)
  points$y <- points$y * sqrt(area / current.area)

  return (points)
}

regular.poly <- function(nSides, area)
{
  radius <- sqrt((2 * area) / (nSides * sin((2 * pi) / nSides)))

  points <- list(x = NULL, y = NULL)
  angles <- (2 * pi) / nSides * 1:nSides

  points$x <- cos(angles) * radius
  points$y <- sin(angles) * radius

  return (points)

}


### generate 9 random cells
coordinates <- rep(seq(-2, 2, length.out = 3), 3)
x <- matrix(coordinates, ncol = 3, byrow = F)
y <- matrix(coordinates, ncol = 3, byrow = T)

cells <-
  sapply(1:9, function(i) {
    xy <- c(x[i], y[i])
    cell <- data.frame(cell = paste("cell#", i, sep = ""),
                       convex.poly(
                         nSides = round(runif(
                           n = 1, min = 15, max = 30
                         ), 0),
                         area = rnorm(n = 1, mean = 2.5, sd = .3)
```

```
                        ))
    cell[2:3] <- t(apply(cell[2:3], 1, function(row) {
      row + xy
    }))

    return(cell)
  }, simplify = FALSE) %>% do.call("rbind", .)

cells$cell <- factor(cells$cell, levels = paste("cell#", 1:9, sep = ""))


##### generate 16 ablations

coordinates <- rep(seq(-2, 2, length.out = 4), 4)
x <- as.vector(matrix(coordinates, ncol = 4, byrow = F))
y <- as.vector(matrix(coordinates, ncol = 4, byrow = T))

x <- rotate_xy(cbind(x, y), angle = 0.3)[, 1]
y <- rotate_xy(cbind(x, y), angle = 0.3)[, 2]

ablations <-
  sapply(1:16, function(i) {
    xy <- c(x[i], y[i])
    cell <- data.frame(ablation = paste("ablation#", i, sep = ""),
                       regular.poly(nSides = 40, area = .8))
    cell[2:3] <- t(apply(cell[2:3], 1, function(row) {
      row + xy
    }))

    return(cell)
  }, simplify = FALSE) %>% do.call("rbind", .)

ablations$ablation <-
  factor(ablations$ablation, levels = paste("ablation#", 1:16, sep = ""))

### plot simulation

ggplot() +
  coord_fixed() +
  geom_polygon(
    data = cells,
    aes(x = x, y = y, fill = cell),
    color = "black",
    size = 1
  ) +
  geom_polygon(
    data = ablations,
    aes(x = x, y = y, color = ablation),
    fill = "gray",
    alpha = .7,
    size = 2
  ) +
  theme_void()+
```

```
  theme(legend.direction = "vertical",
        legend.position = "right",
        legend.box = "horizontal"
  )
```

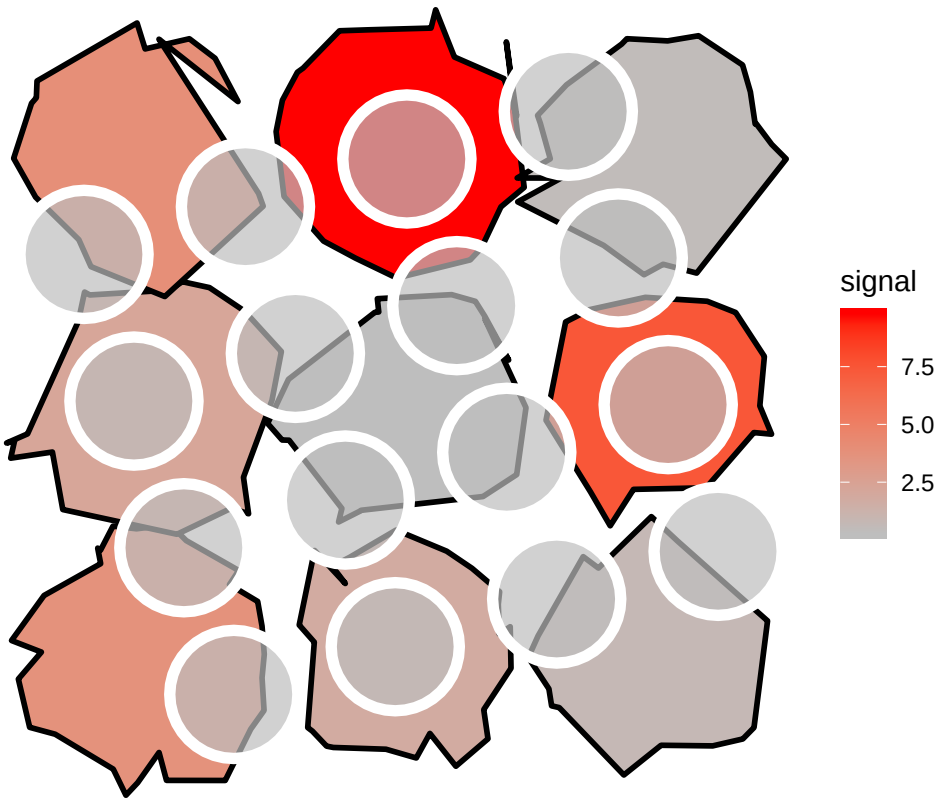



```
#### assign random metabolite X signals

cell_signals <- data.frame(cell = paste("cell#", 1:9, sep = ""),
                           signal = runif(9, min = 0, max = 10))

cell_signals
```

```
##     cell    signal
## 1 cell#1 3.7434571
## 2 cell#2 1.7867082
## 3 cell#3 0.7811720
## 4 cell#4 2.2119831
## 5 cell#5 0.2996752
## 6 cell#6 7.4388045
## 7 cell#7 3.9458130
## 8 cell#8 9.7728409
## 9 cell#9 0.4978408
```

```
cells$signal <-
  cell_signals$signal[match(cells$cell, cell_signals$cell)]

### plot signals
```

```
ggplot() +
  coord_fixed() +
  scale_fill_gradient(low = "gray", high = "red") +
  geom_polygon(
    data = cells,
    aes(
      x = x,
      y = y,
      group = cell,
      fill = signal
    ),
    color = "black",
    size = 1
  ) +
  geom_polygon(
    data = ablations,
    aes(x = x, y = y, group = ablation),
    color = "white",
    fill = "gray",
    alpha = .7,
    size = 2,
    show.legend = F
  ) +
  theme_void()
```



```
###### calculate expected signal in the ablations

## first, formalize polys to measure them
cells_polys <-
  cells %>% group_split(cell) %>% sapply(function(cell) {
    shape <- cbind(cell$x, cell$y)
    shape <- shape[chull(shape),]
    shape <- as(shape, "gpc.poly")
    shape
  })

ablations_polys <-
  ablations %>% group_split(ablation) %>% sapply(function(ablation) {
    shape <- cbind(ablation$x, ablation$y)
    shape <- shape[chull(shape),]
    shape <- as(shape, "gpc.poly")
    shape
  })

## calculate area% of cell_i in abl_i
overlap_matrix <-
  sapply(1:length(ablations_polys), function(abl_i) {
    sapply(1:length(cells_polys), function(cell_i) {
      area.poly(intersect(ablations_polys[[abl_i]], cells_polys[[cell_i]]))
    }) / area.poly(ablations_polys[[abl_i]])
  })

overlap_matrix <- data.frame(overlap_matrix)
rownames(overlap_matrix) <- paste("cell#", 1:9, sep = "")
colnames(overlap_matrix) <- paste("ablation#", 1:16, sep = "")

## matrix looks like:
head(overlap_matrix)
```

```
##        ablation#1 ablation#2 ablation#3   ablation#4 ablation#5 ablation#6
## cell#1   0.745518          0 0.00000000 0.0000000000  0.5721564  0.0000000
## cell#2   0.000000          1 0.03797893 0.0000000000  0.0000000  0.1162700
## cell#3   0.000000          0 0.47335142 0.3159384843  0.0000000  0.0000000
## cell#4   0.000000          0 0.00000000 0.0000000000  0.2930858  0.0000000
## cell#5   0.000000          0 0.00000000 0.0000000000  0.0000000  0.5295149
## cell#6   0.000000          0 0.00000000 0.0003947399  0.0000000  0.0000000
##        ablation#7 ablation#8 ablation#9 ablation#10 ablation#11
## cell#1 0.00000000          0          0   0.0000000   0.0000000
## cell#2 0.00000000          0          0   0.0000000   0.0000000
## cell#3 0.00000000          0          0   0.0000000   0.0000000
## cell#4 0.00000000          0          1   0.2414207   0.0000000
## cell#5 0.54146993          0          0   0.3335744   0.4741416
## cell#6 0.04267561          1          0   0.0000000   0.0000000
##        ablation#12 ablation#13 ablation#14 ablation#15 ablation#16
## cell#1  0.00000000  0.00000000           0           0           0
## cell#2  0.00000000  0.00000000           0           0           0
## cell#3  0.00000000  0.00000000           0           0           0
## cell#4  0.00000000  0.09804151           0           0           0
## cell#5  0.00000000  0.00000000           0           0           0
```

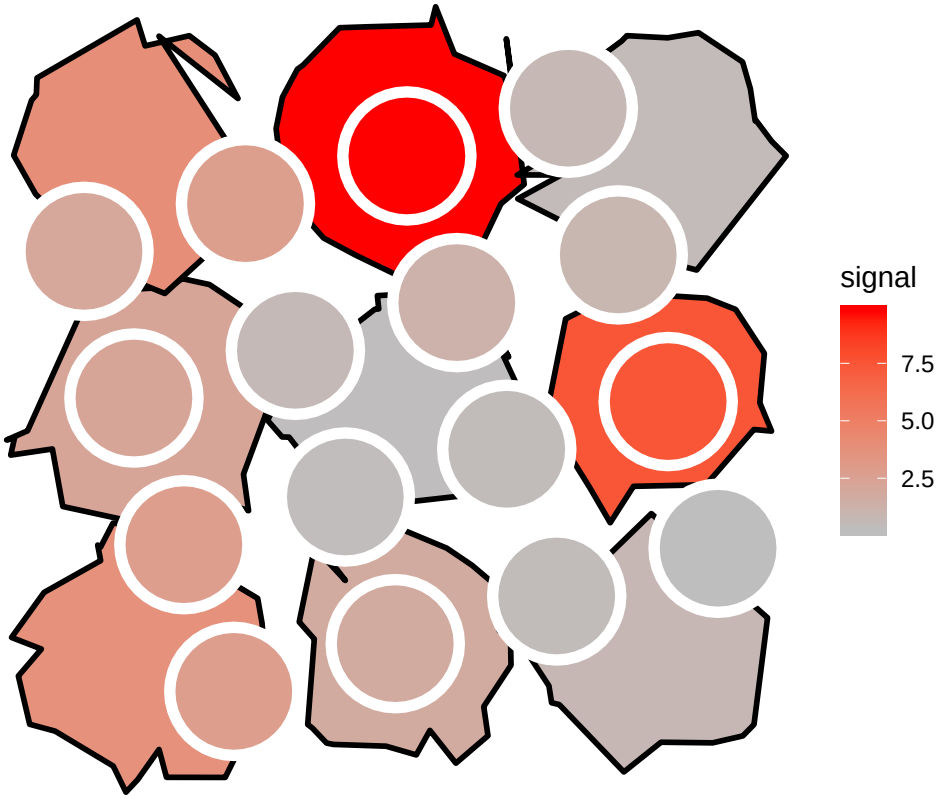
```
## cell#6  0.09015843  0.00000000           0           0           0
```

```
## now calculate anticipated signals in ablations
ablations_signals <-
  data.frame(
    ablation = paste("ablation#", 1:16, sep = ""),
    signal = apply(overlap_matrix, 2, function(abl_i) {
      sum(abl_i * cell_signals$signal)
    })
  )

ablations_signals
```

```
##                ablation    signal
## ablation#1   ablation#1 2.7908145
## ablation#2   ablation#2 1.7867082
## ablation#3   ablation#3 0.4376261
## ablation#4   ablation#4 0.2497387
## ablation#5   ablation#5 2.7901437
## ablation#6   ablation#6 0.3664230
## ablation#7   ablation#7 0.4797206
## ablation#8   ablation#8 7.4388045
## ablation#9   ablation#9 2.2119831
## ablation#10 ablation#10 0.6339824
## ablation#11 ablation#11 1.2230068
## ablation#12 ablation#12 0.9151484
## ablation#13 ablation#13 2.0719167
## ablation#14 ablation#14 2.7430365
## ablation#15 ablation#15 9.7728409
## ablation#16 ablation#16 0.7768438
```

```
ablations$signal <-
  ablations_signals$signal[match(ablations$ablation, ablations_signals$ablation)]


## plot signals to check, seems to be ok
ggplot() +
  coord_fixed() +
  scale_fill_gradient(low = "gray", high = "red") +
  geom_polygon(
    data = cells,
    aes(
      x = x,
      y = y,
      group = cell,
      fill = signal
    ),
    color = "black",
    size = 1
  ) +
  geom_polygon(
    data = ablations,
    aes(
      x = x,
      y = y,
```

```
      group = ablation,
      fill = signal
    ),
    color = "white",
    size = 2,
    show.legend = F
  ) +
  theme_void()
```

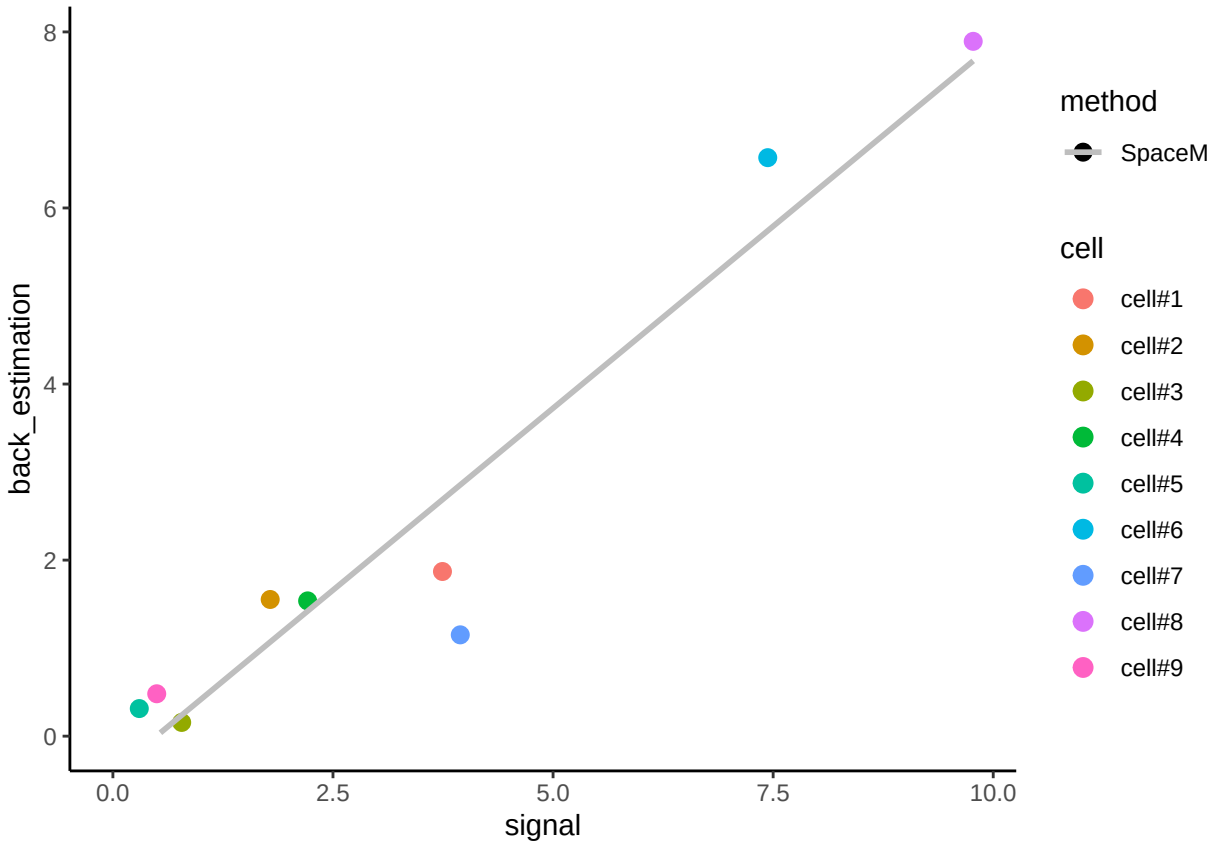



```
### now calculate back with SpaceM method
### the metabolite concentrations per cell
cell_signals$SpaceM <-
  sapply(1:9, function(cell_i) {
    top_bottom <- rowSums(sapply(1:16, function(abl_i) {
      top <-
        ablations_signals$signal[abl_i] * overlap_matrix[cell_i, abl_i] *
        area.poly(intersect(ablations_polys[[abl_i]], cells_polys[[cell_i]]))
      bottom <-
        area.poly(intersect(ablations_polys[[abl_i]], cells_polys[[cell_i]]))
      c(top, bottom)
    }))
    top_bottom[1] / top_bottom[2]
  })

cell_signals
```

```
##     cell    signal    SpaceM
## 1 cell#1 3.7434571 1.8703529
## 2 cell#2 1.7867082 1.5527786
## 3 cell#3 0.7811720 0.1558152
## 4 cell#4 2.2119831 1.5365680
## 5 cell#5 0.2996752 0.3134495
## 6 cell#6 7.4388045 6.5715919
## 7 cell#7 3.9458130 1.1508654
## 8 cell#8 9.7728409 7.8928818
## 9 cell#9 0.4978408 0.4814599
```

```
cell_signals_long <-
  melt(
    cell_signals,
    id.vars = 1:2 ,
    variable.name = "method",
    value.name = "back_estimation"
  )

## plot comparison
ggplot(data = cell_signals_long,
       aes(
         x = signal,
         y = back_estimation,
         color = cell,
         shape = method
       )) +
  scale_x_continuous(limits = c(0, NA)) +
  scale_y_continuous(limits = c(0, NA)) +
  coord_fixed() +
  geom_point(size = 3) +
  geom_smooth(
    aes(linetype = method),
    method = "lm",
    se = F,
    color = "gray"
  ) +
  theme_classic()
```

```
## Warning: Removed 2 rows containing missing values (geom_smooth).
```




```
### in fact, it is a linear inverse problem, with 16
### knowns (ablations) and 10 unknowns (9 cells + background)
### A*x = B

B <- ablations_signals$signal
B
```

```
##  [1] 2.7908145 1.7867082 0.4376261 0.2497387 2.7901437 0.3664230 0.4797206
##  [8] 7.4388045 2.2119831 0.6339824 1.2230068 0.9151484 2.0719167 2.7430365
## [15] 9.7728409 0.7768438
```

```
A <- overlap_matrix
A <- t(as.matrix(rbind(A, 1 - colSums(A)))) ## last row is background
colnames(A)[10] <- "background"
A
```

```
##                cell#1     cell#2    cell#3     cell#4    cell#5
## ablation#1  0.7455180 0.00000000 0.0000000 0.00000000 0.0000000
## ablation#2  0.0000000 1.00000000 0.0000000 0.00000000 0.0000000
## ablation#3  0.0000000 0.03797893 0.4733514 0.00000000 0.0000000
## ablation#4  0.0000000 0.00000000 0.3159385 0.00000000 0.0000000
## ablation#5  0.5721564 0.00000000 0.0000000 0.29308578 0.0000000
## ablation#6  0.0000000 0.11626997 0.0000000 0.00000000 0.5295149
## ablation#7  0.0000000 0.00000000 0.0000000 0.00000000 0.5414699
## ablation#8  0.0000000 0.00000000 0.0000000 0.00000000 0.0000000
## ablation#9  0.0000000 0.00000000 0.0000000 1.00000000 0.0000000
## ablation#10 0.0000000 0.00000000 0.0000000 0.24142067 0.3335744
## ablation#11 0.0000000 0.00000000 0.0000000 0.00000000 0.4741416
```

```
## ablation#12 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000
## ablation#13 0.0000000 0.00000000 0.0000000 0.09804151 0.0000000
## ablation#14 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000
## ablation#15 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000
## ablation#16 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000
##                   cell#6    cell#7     cell#8    cell#9 background
## ablation#1  0.0000000000 0.0000000 0.00000000 0.0000000  0.2544820
## ablation#2  0.0000000000 0.0000000 0.00000000 0.0000000  0.0000000
## ablation#3  0.0000000000 0.0000000 0.00000000 0.0000000  0.4886697
## ablation#4  0.0003947399 0.0000000 0.00000000 0.0000000  0.6836668
## ablation#5  0.0000000000 0.0000000 0.00000000 0.0000000  0.1347578
## ablation#6  0.0000000000 0.0000000 0.00000000 0.0000000  0.3542151
## ablation#7  0.0426756050 0.0000000 0.00000000 0.0000000  0.4158545
## ablation#8  1.0000000000 0.0000000 0.00000000 0.0000000  0.0000000
## ablation#9  0.0000000000 0.0000000 0.00000000 0.0000000  0.0000000
## ablation#10 0.0000000000 0.0000000 0.00000000 0.0000000  0.4250050
## ablation#11 0.0000000000 0.0000000 0.11060431 0.0000000  0.4152541
## ablation#12 0.0901584338 0.0000000 0.00000000 0.4910755  0.4187660
## ablation#13 0.0000000000 0.4701314 0.00000000 0.0000000  0.4318271
## ablation#14 0.0000000000 0.4823744 0.08591947 0.0000000  0.4317061
## ablation#15 0.0000000000 0.0000000 1.00000000 0.0000000  0.0000000
## ablation#16 0.0000000000 0.0000000 0.04633273 0.6508936  0.3027737
```

```
### Solve system:
cell_signals$LIMSolve <- Solve(A, B)[1:9] ##10th element is background
cell_signals
```

```
##     cell    signal    SpaceM  LIMSolve
## 1 cell#1 3.7434571 1.8703529 3.7434571
## 2 cell#2 1.7867082 1.5527786 1.7867082
## 3 cell#3 0.7811720 0.1558152 0.7811720
## 4 cell#4 2.2119831 1.5365680 2.2119831
## 5 cell#5 0.2996752 0.3134495 0.2996752
## 6 cell#6 7.4388045 6.5715919 7.4388045
## 7 cell#7 3.9458130 1.1508654 3.9458130
## 8 cell#8 9.7728409 7.8928818 9.7728409
## 9 cell#9 0.4978408 0.4814599 0.4978408
```

```
cell_signals_long <-
  melt(
    cell_signals,
    id.vars = 1:2 ,
    variable.name = "method",
    value.name = "back_estimation"
  )

### plot comparisons, LIMSolve seems to be more exact
ggplot(data = cell_signals_long,
       aes(
         x = signal,
         y = back_estimation,
         color = cell,
         shape = method
       )) +
  scale_x_continuous(limits = c(0, NA)) +
```

```
  scale_y_continuous(limits = c(0, NA)) +
  coord_fixed() +
  geom_point(size = 3) +
  geom_smooth(
    aes(linetype = method),
    method = "lm",
    se = F,
    color = "gray"
  ) +
  theme_classic()
```

```
## Warning: Removed 2 rows containing missing values (geom_smooth).
```