Materials and Methods

Orestis Loukas & Ho-Ryun Chung

Toy model

Integer solutions

The toy model with $L = 3$ binary features has $|\mathcal{A}| = 8$ microstates. Using pairwise marginal constraints as information for the estimation by IPF we have a coefficient matrix $\mathbf{C}$ with the $D = 12$ marginal constraints in the rows and the $|\mathcal{A}| = 8$ microstates in the columns

```
C <- matrix(
  c(
    1, 1, 0, 0, 0, 0, 0, 0,
    0, 0, 1, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 1, 1,
    1, 0, 1, 0, 0, 0, 0, 0,
    0, 1, 0, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 0,
    0, 0, 0, 0, 0, 1, 0, 1,
    1, 0, 0, 0, 1, 0, 0, 0,
    0, 1, 0, 0, 0, 1, 0, 0,
    0, 0, 1, 0, 0, 0, 1, 0,
    0, 0, 0, 1, 0, 0, 0, 1
  ),
  ncol = 8,
  byrow = TRUE
)
```

The marginal constraints $\{\widehat{m}_a\}_{a=1,\ldots,D}$ are given by

```
mVal = c(30, 14, 27, 29, 20, 24, 48, 8, 39, 18, 29, 14)
```

The integer solutions of the so-defined linear system are

```
sols = matrix(
    c(
        15, 15, 5, 9, 24, 3, 24, 5,
        14, 16, 6, 8, 25, 2, 23, 6,
        16, 14, 4, 10, 23, 4, 25, 4,
        13, 17, 7, 7, 26, 1, 22, 7,
        17, 13, 3, 11, 22, 5, 26, 3,
        18, 12, 2, 12, 21, 6, 27, 2,
        12, 18, 8, 6, 27, 0, 21, 8,
        19, 11, 1, 13, 20, 7, 28, 1,
        20, 10, 0, 14, 19, 8, 29, 0
    ),
    ncol = 8,
    byrow = TRUE
)
```

a matrix with the solutions in the rows and the microstates in the columns.

Using the solutions we calculate the multinomial probability mult$(N\mathfrak{p};\mathfrak{u})$

```
multinom <- apply(sols, 1, function(x) dmultinom(x, prob = rep(1, 8), log = TRUE))
```

the multinomial coefficients $W[N\mathfrak{p}]$

```
mult <- apply(sols, 1, function(x) lfactorial(sum(x)) - sum(lfactorial(x)))
```

and N times the entropy $NH[\mathfrak{p}]$

```
nEntropy <- apply(
    sols, 1,
    function(x){
        sum(x) * sum(ifelse(x > 0, -x / sum(x) * log(x / sum(x)), 0))
    }
)
```

We define the microstates

```
microstates <- t(sapply(0:7,function(x){ as.integer(intToBits(x))})[1:3, ])
microstates <- microstates[order(microstates[,1], microstates[,2], microstates[,3]), ]
```

and render **Figure 1**

```
## get color palette
source("../R/colors.R")
pdf("../figures/Figure1.pdf", width = 4.33, height = 1.4, pointsize = 6)
par(mar = c(0,0,0,0) + 0.1)
plot(NULL, xlim = c(0, 67), ylim = c(15, 0), axes = FALSE, ann = FALSE, frame = FALSE)

## marginals
cr <- c(cols$white, cols$black)
off = 0
for (i in 1:12){
  mI <- which(C[i, ] == 1)
  fI <- which(microstates[mI[1], ] == microstates[mI[2], ])
  st <- microstates[mI[1], fI]
  rect(
    (fI - 0.5), (i - 0.5) + off, (fI + 0.5), (i + 0.5) + off,
    col = cr[st + 1], border = cols$grey, lwd = 0.2
  )
  text(0.5, i + off, mVal[i], pos = 2)

  if (i %% 4 == 0){
      off <- off + 0.2
  }
}

## coefficient matrix C == bipartite graph
rr1 <- which(C != 0, arr.ind = TRUE)
segments(4, rr1[,1] + 0.2 * ((rr1[, 1] - 1) %/% 4), 9, rr1[,2] + 2, col = cols$red)
yy <- 3:10
for (i in 1:3){
  rect(
    9 + i - 0.5, (yy - 0.5), 9 + i + 0.5, (yy + 0.5),
    col = cr[microstates[, i] + 1], border = cols$grey, lwd = 0.2
  )
}

## integer solution vectors
base <- 13
text(40, 1.4, "Integer solution vectors")
segments(13, 2.2, 67, 2.2)
for (i in 1:nrow(sols)){
    text(base + 6, yy, sols[i,], pos = 2)
    base <- base + 6
}
segments(13, 10.8, 67, 10.8)

## multinomial probability
text(seq(19, 67, by = 6), 11.6, sprintf("%.2f", multinom), pos = 2, cex = 1)
## multinomial coefficient
text(seq(19, 67, by = 6), 12.6, sprintf("%.2f", mult), pos = 2, cex = 1)
## N times Entropy
text(seq(19, 67, by = 6), 13.6, sprintf("%.2f", nEntropy), pos = 2, cex = 1)
## labels
text(13, 11.6, "mult p", pos = 2)
text(13, 12.6, "mult c", pos = 2)
text(13, 13.6, "NH", pos = 2)
segments(13, 14.4, 67, 14.4)
dev.off()
```

ipf algorithm

To illustrate the evolution of the estimate $\hat{\mathrm{p}}$ during the iterations in IPF we use the function `Ipfp` from the package `mipfp`

```
if (! require("mipfp", quietly = TRUE))
  install.packages("mipfp")
library(mipfp)
```

`Ipfp` takes as arguments

- `seed` the initial multi-dimensional array to be updated
- `target.list` a list of dimensions correspond to the marginal constraints
- `target.data` the marginal constraints

We defined `seed` using the uniform distribution (all entries are 1/8) and `dim = c(2,2,2)`

```
seed = array(1/8, dim = c(2,2,2))
```

the `target.list`

```
target.list <- list(
  c(1,2),
  c(1,3),
  c(2,3)
)
```

and the `target.data`

```r
target.data <- list(
  array(c(30,27,14,29) / 100, dim = c(2,2)),
  array(c(20,48,24,8) / 100, dim = c(2,2)),
  array(c(39, 29, 18,14) / 100, dim = c(2,2))
)
```

We ran IPF

```r
ipf <- Ipfp(seed, target.list, target.data)
```

The results is in `ipf$p.hat`.

We recorded the evolution of $\hat{\mathfrak{p}}$ during the iterations of IPF

```r
## current result
result <- seed

## trajectory of p
traj <- list()
traj[[1]] = result
k <- 2

## look at the first 4 cycles
for (i in 1:4) {
  result.temp <- result
  for (j in 1:length(target.list)) {
    temp.sum <- apply(result, target.list[[j]], sum)
    update.factor <- ifelse(target.data[[j]] == 0 | temp.sum ==
                              0, 0, target.data[[j]]/temp.sum)
    result <- sweep(result, target.list[[j]], update.factor,
                    FUN = "*")
    traj[[k]] <- result
    k <- k + 1
  }
}
## add the final solution of IPF to the trajectory
traj[[k]] <- ipf$p.hat
```

We generated a matrix of indices to query the multidimensional arrays

```r
idx <- as.matrix(expand.grid(c(1,2), c(1,2), c(1,2)))
```

Using `idx` we get the values of $\hat{\mathfrak{p}}$ during the different phases

```r
gg <- t(sapply(traj, function(x) x[idx]))
```

and calculate the KL divergence from the final estimate $\hat{\mathfrak{p}}$ obtained by IPF

```r
dkl = sapply(traj[1:13], function(x) sum(gg[14, ] * log(gg[14,] / x)))
```

Finally we render **Figure 2**

```r
pdf(file = "../figures/Figure2.pdf", width = 4.33, height = 2.5, pointsize = 6)

layout(
  matrix(c(1,2), nrow = 2),
  height = c(2,1)
)
```

```
par(mar = c(0.1,4,0,8) + 0.1)
plot(NULL, ylim = c(0, 0.3), xlim = c(1,13),
     ylab = "IPF estimate", axes = FALSE, xlab = ""
)

rect(0.5, 0, 1.5, 0.3, col = cols$white, border = NA)
rect(4.5, 0, 7.5, 0.3, col = cols$white, border = NA)
rect(10.5, 0, 13.5, 0.3, col = cols$white, border = NA)
text(c(3,6,9, 12), rep(0.28, 4), paste0(1:4, ". cycle"))
abline(h = gg[14,], lty = 3, col = unlist(cols[3:10]))

for (i in 1:8){
  lines(gg[1:13,i], col = cols[[i + 2]])
  points(gg[1:13,i], col = cols[[i + 2]], pch = 16)
}
axis(2, las = 1)

par(xpd = TRUE)
text(13.5, gg[14, ], gg[14,], col = unlist(cols[3:10]), pos = 4)
par(xpd = FALSE)

par(mar = c(2,4,0,8) + 0.1)

plot(dkl, log = 'y', frame = FALSE, xaxt = "n", ylim = c(1e-7,1), type = 'l', xlab = "")
points(dkl, pch = 16)
axis(1, at = 1:13, labels = 0:12)

dev.off()
```

Iterative proportional fitting using the NHAMCS data

We will be using data from the public database of the *National Center for Health Statistics*. Specifically, we have considered all entries documented over the years 2003,-,2018 at the emergency department (ED) by the National Hospital Ambulatory Medical Care Survey (NHAMCS)[1].

We have concentrated on patients whose fate is known, excluding anyone who either left against medical advice or left without being seen or before treatment completion. In addition, anyone who arrived already dead at the ED was excluded. In this way, $N = 392\,454$ records remained from which 44 293 were admitted to the hospital. A patient is considered critical whenever the patient is admitted to an intensive/critical care unit (ICU) or died during the hospital stay resulting in 6 300 critical cases out of 44 293 hospitalized patients. Here, we encounter a structural zero, because non-hospitalized but critical patients are impossible.

In detail, we define $L = 5$ categorical features:

- `ageGroup`: age group of a patient with $q_1 = 6$ states: $[0, 15)$ $(\alpha_1 = 0)$, $[15, 25)$ $(\alpha_1 = 1)$, $[25, 45)$ $(\alpha_1 = 2)$, $[45, 65)$ $(\alpha_1 = 3)$, $[65, 75)$ $(\alpha_1 = 4)$, and $[76, 100)$ $(\alpha_1 = 5)$
- `sex`: sex of patient with $q_2 = 2$ states: `female` $(\alpha_2 = 0)$ or `male` $(\alpha_2 = 1)$
- `arrivalByAmbulance`: did the patient arrive by ambulance? with $q_3 = 2$ states: `no` $(\alpha_3 = 0)$ or `yes` $(\alpha_3 = 1)$
- `hospitalization`: was the patient hospitalized? with $q_4 = 2$ states: `no` $(\alpha_4 = 0)$ or `yes` $(\alpha_4 = 1)$
- `critical`: was the patient admitted to the intensive care unit or died? with $q_5 = 2$ states: ‘`no` $(\alpha_5 = 0)$ or `yes` $(\alpha_5 = 1)$

[1]Datasets and documentation can be downloaded from https://www.cdc.gov/nchs/ for public use.

There are $|\mathcal{A}| = 6 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 96$ microstates. However, from those only $|\mathcal{A}'| = 72$ are not associated with the aforementioned structural zero. We consider *all* marginal relative frequencies of orders $\ell = 1, 2, 3, 4$. In principle we could estimate also the MaxEnt distribution for $\ell = 5$, but due to the reduced state space $\mathcal{A}'$, the equivalence class $[f]_{\ell=4}$ incorporating all 4-order constraints is identical to $[f]_{\ell=5}$ induced by the empirical distribution $\mathfrak{f}$ itself.

Data preparations

The data with the empirical counts for the $|\mathcal{A}| = 96$ microstates is stored in the file `../data/empirical.csv`, which we load into R:

```
empirical <- read.csv("../data/empirical.csv")
```

Here, we show the first 5 rows of the table:

```
##   ageGroup sex arrivalByAmbulance hospitalization critical count
## 1        0   0                  0               0        0 32747
## 2        1   0                  0               0        0 31172
## 3        2   0                  0               0        0 54007
## 4        3   0                  0               0        0 32345
## 5        4   0                  0               0        0  8159
## 6        5   0                  0               0        0  8761
```

Some of the microstates have not been observed. These are associated with the structural zero between `hospitalization` = `no` and `critical` = `yes`:

```
##    ageGroup sex arrivalByAmbulance hospitalization critical count
## 49        0   0                  0               0        1     0
## 50        1   0                  0               0        1     0
## 51        2   0                  0               0        1     0
## 52        3   0                  0               0        1     0
## 53        4   0                  0               0        1     0
## 54        5   0                  0               0        1     0
```

We define the number of features L and the number of states per features $\mathbf{q}$

```
L <- 5
q <- c(6, 2, 2, 2, 2)
```

From the empirical data we construct a joint distribution $\mathfrak{f}$ as an `array` with `dim = c(6,2,2,2,2)` $= \mathbf{q}$. First, we extract a matrix $\mathcal{A}$ with the $|\mathcal{A}| = 96$ microstates in the rows and the $L = 5$ features in the columns:

```
A <- as.matrix(
  empirical[,
    c(
      "ageGroup",
      "sex",
      "arrivalByAmbulance",
      "hospitalization",
      "critical"
    )
  ]
)
```

As R uses 1-based indexing, we construct from this an index matrix by adding one to any entry in $\mathcal{A}$:

```
idx <- A + 1
```

The empirical count for each microstate is stored in the column `count` of the data frame `empirical`:

```
h <- empirical$count
```

The total number of records in the data set is given by the sum of the counts

```
N <- sum(h)
```

We define the aforementioned array $\mathfrak{f}$ with `dim` $=$ $\mathbf{q}$

```
f <- array(
  1,
  dim = q
)
```

and fill the array using `idx` with the counts in `h`

```
f[idx] = h
```

Estimation of maxent distributions with ipf

Computation of all ℓ-order marginal constraints from $\mathfrak{f}$

Using the empirical frequencies $\mathfrak{f}$ for all the $|\mathcal{A}| = 96$ we compute the marginal constraints of order ℓ. These correspond to all ℓ-way contingency tables. We define a function to calculate these constraints

```
marginalsByOrder <- function(f, ell, L){
  if (ell > 0 && ell < L){
    tuples <- combn(seq_len(L), ell)
    marginals <- apply(
      tuples, 2,
      function(tuple){
        apply(f, tuple, sum)
      }
    )
    list <- lapply(seq_len(ncol(tuples)), function(i) tuples[, i])

    list(marginals = marginals, list = list)
  } else{
    list(marginals = f, list = list(seq_len(L)))
  }
}
```

The function takes as argument

- `f` the empirical frequencies $\mathfrak{f}$ stored in a multidimensional `array` with `dim` $=$ $\mathbf{q}$
- `ell` the order ℓ of the marginal constraints to compute
- `L` the number of features L

As we want to compute *all* marginal constraints of order ℓ we generate all combinations of size ℓ using L features

```
tuples <- combn(seq_len(L), ell)
```

and then compute the marginals

```
marginals <- apply(
  tuples, 2,
  function(tuple){
    apply(f, tuple, sum)
  }
)
```

we store a list of feature indices that correspond to the marginal constraints

```
list <- lapply(seq_len(ncol(tuples)), function(i) tuples[, i])
```

Obviously, the marginal constraints are defined for orders $\ell = 1, \ldots, L$. However, marginal constraints with order $\ell = L$ correspond to the observed distribution $\mathfrak{f}$. Thus, in case $\ell = L$, we just return $\mathfrak{f}$.

Running ipf

We defined `seed` using the uniform distribution (all entries are 1) and `dim` $= \mathbf{q}$

```
seed <- array(
  1,
  dim = q
)
```

order $\ell = 1$ marginal constraints

```
m1 <- marginalsByOrder(
  f = f,
  ell = 1,
  L = L
)
phat1 <- Ipfp(
  seed = seed,
  target.list = m1$list,
  target.data = m1$marginals,
  ## verbose
  print = TRUE
)
```

```
## Margins consistency checked!
## ... ITER 1
##        stoping criterion: 45162.5
## ... ITER 2
##        stoping criterion: 7.275958e-12
## Convergence reached after 2 iterations!
```

IPF stops after the first iteration. Trivially, first order constraints assume independence, i.e. the joint distribution can be calculated in one step. `Ipfp` returns

```
## List of 7
##  $ x.hat        : num [1:6, 1:2, 1:2, 1:2, 1:2] 30734 24333 45164 33460 9975 ...
##  $ p.hat        : num [1:6, 1:2, 1:2, 1:2, 1:2] 0.0783 0.062 0.1151 0.0853 0.0254 ...
##  $ conv         : logi TRUE
##  $ error.margins: num [1:5] 3.64e-12 2.91e-11 0.00 0.00 0.00
##  $ evol.stp.crit: num [1:2] 4.52e+04 7.28e-12
##  $ method       : chr "ipfp"
##  $ call         : language Ipfp(seed = seed, target.list = m1$list, target.data = m1$marginals, prin
##  - attr(*, "class")= chr [1:2] "list" "mipfp"
```

a list of 7 elements, where `$ p.hat` contains the estimated joint probability distribution $\hat{\mathfrak{p}}^{(\ell=1)}$.

order $\ell = 2$ marginal constraints

```r
m2 <- marginalsByOrder(
  f = f,
  ell = 2,
  L = L
)
phat2 <- Ipfp(
  seed = seed,
  target.list = m2$list,
  target.data = m2$marginals
)
```

order $\ell = 3$ marginal constraints

```r
m3 <- marginalsByOrder(
  f = f,
  ell = 3,
  L = L
)
phat3 <- Ipfp(
  seed = seed,
  target.list = m3$list,
  target.data = m3$marginals
)
```

order $\ell = 4$ marginal constraints

```r
m4 <- marginalsByOrder(
  f = f,
  ell = 4,
  L = L
)
phat4 = Ipfp(
  seed = seed,
  target.list = m4$list,
  target.data = m4$marginals
)
```

As expect marginal constraints of order $\ell = 4$ give back the empirical distribution $\mathfrak{f}$

```r
all.equal(f, phat4$x.hat)
```

```
## [1] TRUE
```

We store the four MaxEnt distributions in a `data.frame` and save it to file `../data/TableS2.tsv`

```r
maxentFile <- "../data/TableS2.tsv"
maxentP <- data.frame(
  empirical[, 1:5],
  one = phat1$p.hat,
  two = phat2$p.hat,
  three = phat3$p.hat,
  four = phat4$p.hat,
  empirical = empirical$count / sum(empirical$count)
)
write.table(maxentP, file = maxentFile, sep = '\t', quote = TRUE, row.names = FALSE)
```

Subsampling

We assume that the empirical distribution $\mathfrak{f}$ describes a "population" of infinite size, i.e. it corresponds to a fictitious asymptotic distribution. From this "asymptotic" distribution we sample data sets of size N_s with replacement. These data sets are conveniently summarized by the frequencies of the 72 possible microstates (`fs`). To prevent infinities during the calculation of the KL divergence we add a pseudocount of unity to all 72 admissible microstates.

ℓ-uniform order marginal constraints

We estimate model probability distributions $\mathfrak{p}^{(s,\ell)}$ for sets of marginal constraints encompassing *all* marginal constraints of a given order $\ell = 1, 2, 3, 4$. Model distributions $\hat{\mathfrak{p}}_s^{(\ell)}$ obtained with $\ell = 4, 5$ coincide due to the structural zero between hospitalization and criticality.

We will subsample (with replacement) from the empirical distribution $\mathfrak{f}$. For this we define a function

```r
subSampleUniform <- function(subSampleSize, ells = c(1,2,3)){

  ## define the multi-dimensional "empirical" distribution
  fs <- array(
    1,
    dim = q
  )
  ## fill it with multinomial samples according to the empirical
  ## multinomial probabilitites stored in f
  fs[idx] <- rmultinom(
    n = 1,
    size = subSampleSize,
    prob = h
  )

  ## regularize all entries, where the original f was non-zero
  fs[f > 0] = fs[f > 0] + 1

  ## compute the IPF models for the orders stored in ells
  res <- sapply(
    ells,
    function(ell){
      ## order ell marginal constraints
      m <- marginalsByOrder(fs, ell, L)

      ## define the seed
      seed <- array(
        1,
        dim = q
      )

      ## run IPF
      p.hat = Ipfp(
        seed = seed,
        target.list = m$list,
        target.data = m$marginals,
        iter = 10000
      )$p.hat
      ## calculate KL divergence
      sum(ifelse(f > 0, f * (log(f) - log(p.hat)), 0))
    }
  )
  ## "empirical"
  c(
    res,
    sum(ifelse(f > 0, f * (log(f) - log(fs / sum(fs))), 0))
  )

}
```

Every "model" uses the complete set of possibly redundant marginal constraints.

To get reproducible results we set a `seed`

```
seed = 20220308
set.seed(seed)
```

and sample for each of the subsample sizes 20000 subsamples

```
Ns <- c(5000, seq(10000, 390000, by = 10000), N)
nSamples <- 20000
```

For each of these subsamples we compute the KL divergence from the empirical $\mathfrak{f}$ for the four different orders $\ell = 1, 2, 3, 4$ of marginal constrains. We perform the subsampling in parallel using the function `mclapply` of the package `parallel`

```
if (! require("parallel", quietly = TRUE))
  install.packages("parallel")
library(parallel)
```

using `mc.cores` = 256 cores:

```
dklUniform <- lapply(
  Ns,
  function(subSampleSize){

    res <- mclapply(
      1:nSamples,
      function(i){
        subSampleUniform(subSampleSize)
      },
      mc.cores = 256
    )

    matrix(unlist(res), ncol = 4, byrow = TRUE)
  }
)
```

The results are saved in a file `../data/dklUniform.RData`

```
dklFile <- "../data/dklUniform.RData"
save(seed, dklUniform, file = dklFile)
```

For convenience, we provide a R-script in `../R/uniformOrderConstraints.R` that performs the subsampling, estimation of the models, and calculation of the KL divergence. The results can be loaded by sourcing the corresponding R-script.

Figure 3 This script produces Figure 3. We generate/load the subsample data stored in list `dklUniform` by sourcing the file `../R/subSample.R`

```
source("../R/uniformOrderConstraints.R")
```

In addition we source our color palette

```
source("../R/colors.R")
```

Compute the average KL-divergences per subsample size and per $\hat{\mathfrak{p}}_s^{(\ell)}$ model

```
KLavg <- sapply(
  dklUniform,
  colMeans
)
```

$\mathfrak{p}^{(s,\ell)}$

Render the Figure

```
pdf("../figures/Figure3.pdf", width = 7.08, height = 2.5, pointsize = 8)
par(mar = c(3,3,1,0) + 0.1)
plot(
  1:ncol(KLavg), KLavg[2, ],
  ylim = c(1e-4, 1e-2), log = "y",
  col = cols$yellow, frame = FALSE,
  xlim = c(0, ncol(KLavg) + 1),
  pch = 16, cex = 0.5,
  axes = FALSE, ann = FALSE, xaxs = "i"
)
lines(1:ncol(KLavg), KLavg[2,], col = cols$yellow)
points(1:ncol(KLavg), KLavg[3, ], col = cols$red, pch = 16, cex = 0.5) ## IPF-3
lines(1:ncol(KLavg), KLavg[3, ], col = cols$red)
points(1:ncol(KLavg), KLavg[4,], col = cols$purple, pch = 16, cex = 0.5) ## IPF-4
lines(1:ncol(KLavg), KLavg[4,], col = cols$purple)

axis(2, at = c(1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2), las = 1)
axis(1, at = 1:ncol(KLavg), labels = as.integer(Ns) / 1000)
par(xpd = TRUE)
legend(
  ncol(KLavg) - 1,
  max(KLavg[-1,]),
  c("$\\maxentP^{(2)}$", "$\\maxentP^{(3)}$", "$\\maxentP^{(4)}$"),
  col = c(cols$yellow, cols$red, cols$purple),
  lty = 1, pch = 16, bty = "n", cex = 0.5
)
par(xpd = FALSE)
dev.off()
```

Mixed-order marginal constraints

We estimate model probability distributions $\hat{\mathfrak{p}}^{s,k}$ for all possible summary statistics, where each of the $L = 5$ features belongs to at least one cluster of features. We generate all combinations of marginal constraints of mixed order encompassing at least one constraint for each of the $L = 5$ features using the function `getCombinations` provided in `../R/getCombinations.R`. This function takes two arguments:

- `L` the number of features
- `upTo` the maximal order of constraints to be considered

We evaluate `getCombinations` this by setting $\texttt{L} = 5$ and $\texttt{upTo} = 4$

```
source("../R/getCombinations.R")
combinations <- getCombinations(L = L, upTo = 4)
nn <- combinations$nn
combinations <- combinations$combinations
```

Each combination is a `list` with four sublists, each enumerating the indices of the marginal constraints to be considered.

Again, we will subsample (with replacement) from the empirical distribution $\mathfrak{f}$. For this we define a function

```
subSampleMixed <- function(subSampleSize){
  ## define the multi-dimensional "empirical" distribution
  fs <- array(
    1,
    dim = q
  )

  ## fill it with multinomial samples according to the empirical
  ## multinomial probabilities stored in f
  fs[idx] <- rmultinom(
    n = 1,
    size = subSampleSize,
    prob = h
  )

  ## regularize all entries, where the original f was non-zero
  fs[f > 0] <- fs[f > 0] + 1
  margByOrder <- lapply(
    c(1,2,3,4),
    function(ell){
      marginalsByOrder(
        f = fs,
        ell = ell,
        L = L
      )
    }
  )

  ## estimate the MaxEnt distribution
  ## for each set of marginal constraints
  PP <- mclapply(
    combinations,
    function(comb){
      seed <- array(
        1,
        dim = q
      )
      ## run IPF
      target.list <- c(
        margByOrder[[1]]$list[comb[[1]]],
        margByOrder[[2]]$list[comb[[2]]],
        margByOrder[[3]]$list[comb[[3]]],
        margByOrder[[4]]$list[comb[[4]]]
      )
      target.data <- c(
        margByOrder[[1]]$marginals[comb[[1]]],
        margByOrder[[2]]$marginals[comb[[2]]],
        margByOrder[[3]]$marginals[comb[[3]]],
        margByOrder[[4]]$marginals[comb[[4]]]
      )

      Ipfp(
        seed = seed,
        target.list = target.list,
        target.data = target.data,
        iter = 10000
      )$p.hat

    },
    mc.cores = 256
  )

  ## return KL Divergence
  sapply(
    PP,
    function(p){
      sum(ifelse(f > 0, f * (log(f) - log(p)), 0))
    }
  )
}
```

We compute the KL divergence from the empirical $\mathfrak{f}$ for each $k = 1, \ldots, 6\,893$ sets of marginal constraints for each subsample s.

To get reproducible results we set a `seed`

```
seed = 20220309
set.seed(seed)
```

For this analysis we chose a single subsample size $N_s = N$

```
## subSampleSize
subSampleSize = N
```

We determined the KL divergence for each of the 6 893 sets of marginal constraints in chunks of `nSamples` = 100 subsamples. The KL divergences for each set of marginal constraints were saved in a `RDS` file `../data/mixedOrder.<chunk #>.rds`. In total we ran 200 chunks resulting in 20 000 subsamples.

```
## number of samples
nSamples = 100

## subsample in chunks of 200 subsamples
## at a time
for (fold in seq_len(200)){
  cat("\n", sprintf("%03d", fold), "\n")
  dkl <- lapply(
    1:nSamples,
    function(i){
      cat(".")
      if (i %% 50 == 0){
        cat(" ", i, "\n")
      }
      subSampleMixed(subSampleSize)
    }
  )

  saveRDS(dkl, file = paste0("../data/mixedOrder.", sprintf("%03d", fold), ".rds"))
}
```

For convenience, we provide a R-script in `../R/mixedOrderConstraints.R` that performs the subsampling, estimation of the models, and calculation of the KL divergence.

Figure 3 We calculated the rank of the reduced coefficient matrix $\mathbf{C}'$ using the function `Rank` from the `pracma` package

```
if (! require("pracma", quietly = TRUE))
  install.packages("pracma")
library(pracma)
```

We loaded the KL divergences for the 20 000 subsamples for each of the $6,893$ sets of marginal constraints from the 200 chunks generated by the R-script in `../R/mixedOrderConstraints.R`. This resulted in a matrix `dklMixed` with $20,000$ subsamples in the rows and $6,893$ sets of marginal constraints in the columns

```
## load DKLs
dklMixed <- sapply(
    seq_len(200),
    function(fold){
        file <- paste0("../data/mixedOrder.", sprintf("%03d", fold), ".rds")

        readRDS(file)

    }
)
dklMixed <- matrix(unlist(dklMixed), ncol = length(combinations), byrow = TRUE)
```

We determined the index of the all first, second, third, and forth order marginal constraints. For this we generated all combinations using the function `getCombinations` in `../R/getCombinations.R` and the information about

```
## get the data
source("../R/getData.R")
## getCombinations
source("../R/getCombinations.R")

combinations <- getCombinations(L, L - 1)
nn <- combinations$nn
combinations <- combinations$combinations
```

The index is determined by requiring the the combinations contains all 5 first order, 10 second order, 10 third order, and 5 fourth order constraints:

```
## determine the all ell order constraints
allOne <- which(sapply(combinations, function(x) length(x[[1]]) == 5))
allTwo <- which(sapply(combinations, function(x) length(x[[2]]) == 10))
allThree <- which(sapply(combinations, function(x) length(x[[3]]) == 10))
allFour <- which(sapply(combinations, function(x) length(x[[4]]) == 5))
```

We calculate the difference $\Delta_{s,k} = D_{\text{KL}}(\mathfrak{f} \parallel \hat{\mathfrak{p}}^{(s,k)}) - D_{\text{KL}}(\mathfrak{f} \parallel \hat{\mathfrak{p}}^{(s,\ell=4)})$ for each of the subsamples $s = 1, \ldots, 20\,000$

```
deltas <- dklMixed - dklMixed[, allFour]
```

and the median difference for each set $k = 1, \ldots, 6\,893$ of marginal constraints

```
deltasMedian <- apply(deltas, 2, median)
```

and order them in ascending order

```
deltasMedianOrder <- order(deltasMedian)
```

There are several sets of marginal constraints that give the *identical* results (see below). Thus we determined the top 10 median KL divergences

```
top10 <- tapply(
    deltasMedianOrder[1:1000],
    round(deltasMedian[deltasMedianOrder[1:1000]], 10),
    function(i) i
)
top10 <- sum(sapply(top10[1:10], length))
```

The groups of sets of marginal constraints with *identical* results can be understood on the level of the linear system, i.e. they all have the same reduce row echelon form. We determined the reduced coefficient matrix $\mathbf{C}'$ for each set of marginal constraints using the function `getCoefficientMatrix` in `../R/getCoefficientMatrix.R`

```
source("../R/getCoefficientMatrix.R")
```

The function needs the empirical marginals to determine those marginals that are zero

```
## marginals
source("../R/marginalsByOrder.R")
margByOrder <- lapply(
    c(1,2,3,4),
    function(ell){
        marginalsByOrder(
            f = f,
            ell = ell,
            L = L
        )
    }
)
```

Furthermore, it requires the states α_i for each of the features

```
alphas <- list(
    ageGroup = 0:5,
    sex = 0:1,
    arrivalByAmbulance = 0:1,
    hospitalization = 0:1,
    critical = 0:1
)
```

We calculated the reduced coefficient matrix $\mathbf{C}'$ for each of the $6\,893$ sets of constraints

```
CReduced <- lapply(
    combinations,
    getCoefficientMatrix,
    alphas = alphas,
    nn = nn,
    margByOrder = margByOrder

)
```

Finally, we determined the rank of each reduced coefficient matrix $\mathbf{C}'$

```
ranks <- sapply(
    CReduced,
    Rank
)
```

We aggregate the data for the top 10 median KL divergences

```
dataMixed <- list(
    deltaDkl = deltas[,deltasMedianOrder[seq_len(top10)]],
    rank = ranks[deltasMedianOrder[seq_len(top10)]],
    combinations = combinations[deltasMedianOrder[seq_len(top10)]],
    coefficientMatrix = CReduced[deltasMedianOrder[seq_len(top10)]],
    allOne = deltasMedian[allOne],
    allTwo = deltasMedian[allTwo],
    allThree = deltasMedian[allThree],
    allFour = deltasMedian[allFour]
)
```

and saved the result in the file `../data/mixedOrder.RData`

```
save(dataMixed, file = "../data/mixedOrder.RData")
```

For convenience, we provide a R-script in `../R/combineMixed.R` that generates/loads the list `dataMixed`.

Render Figure 4

Load the data

```
source("../R/combineMixed.R")
```

```
pdf("../figures/Figure4.pdf", width = 7.08, height = 5.7, pointsize = 8)
layout(
  matrix(c(1, 2, 3, 3), ncol = 2),
  widths = c(41, 59)
)
```

Figure 3a:

```
ylim <- range(dataMixed$deltaDkl)
medians <- apply(dataMixed$deltaDkl, 2, median)
tab <- table(round(medians, 10))
cs <- c(0, cumsum(tab))
gs <- (cs[-1] - cs[-length(cs)]) / 2 + cs[-length(cs)] + 0.5
breaks <- seq(-max(abs(ylim)) * 1.01, max(abs(ylim)) * 1.01, length.out = 102)
yy <- apply(dataMixed$deltaDkl, 2, function(x)  as.numeric(table(cut(x, breaks))))

plot(0, ylim = ylim, xlim = c(0, ncol(yy) + 1), type = "n",
     axes = FALSE, ann = FALSE, frame = FALSE
)

## violins
k = 1
```

```r
for (i in seq_len(ncol(yy))){

  if (k < 11){
    if (i > gs[k]){
      tmp = yy[, i]
      tmp = (tmp / max(tmp)) * 3
      sel = tmp > 0
      tmp = tmp[tmp > 0]
      rect(gs[k] - tmp, breaks[-length(breaks)][sel],
           gs[k]+ tmp, breaks[-1][sel],
           col = cols$grey, border = NA
      )

      k <- k + 1
    }
  }

}
abline(h = dataMixed$allThree, col = cols$red)
abline(h = dataMixed$allFour, col = cols$purple)

## medians
for (i in seq_len(ncol(yy))){
  ss = round(medians[i], 10)
  #points(i - 0.3, ss, col = cols$white, cex = 0.5, pch = 16)
  if (round(ss, 10) == round(dataMixed$allThree, 10)){
    points(i, ss, pch = 16, cex = 0.8, col = cols$red)
  } else{
    if (round(ss, 10) == round(dataMixed$allFour, 10)){
      points(i, ss, pch = 16, cex = 0.8, col = cols$purple)
    } else{
      points(i, ss, pch = 16, cex = 0.8)

    }
  }

}
axis(2, at = seq(-4, 2, by = 2) * 1e-5, labels = seq(-4, 2, by = 2), las = 1)
axis(1, at = gs, 72 - dataMixed$rank[gs])

mtext(tab, 1, line = 2, at = gs)

mtext("df", 1, line = 1, at = -11, adj = 0)
mtext("# sets", 1, line = 2, at = -11, adj = 0)
```

Figure 3b:

We will need the package `HyperG`

```r
if (! require("HyperG", quietly = TRUE))
  install.packages("HyperG")
library(HyperG)
```

Render the figure

```r
## colors for the cardinality of the constraint
colors = c(NA, cols$yellow, cols$red, cols$purple)

## convert the combinations of constraints
## to a hypergraph in feature space
nams = c("age", "sex", "amb", "hosp", "crit")
toHypergraph = function(comb){
  edges = sapply(1:4, function(i) matrix(nn[[i]]$c[, comb[[i]]], nrow = i))
  edges = lapply(edges, function(e) lapply(seq_len(ncol(e)), function(i) nams[e[, i]]))
  edges = unlist(edges, recursive = FALSE)
  cc = colors[sapply(edges, length)]
  list(gr = hypergraph_from_edgelist(edges, nams), cols = cc, edges = edges)
}

## the same coordinates for the five
## features (a regular pentagon)
ang = 2 * pi / 5 * 1:5
LO = cbind(
  sin(ang) * 0.8,
  cos(ang) * 0.8
)

## there are 13 sets of constraints
## with the same reduced row echelon form
## we define the offsets on the plot
offs = cbind(
  c(2,4,6, 1,3,5,7,1,3,5,7, 3,5),
  c(rep(7, 3), rep(5, 4), rep(3,4), rep(1, 2))
)

## render the graphs
par(mar = c(0,0,3,3) + 0.1)
for (i in 1:13){
  gg = toHypergraph(dataMixed$combinations[[i]])
  LOtmp = LO
  LOtmp[,1] = LOtmp[,1] + offs[i,1]
  LOtmp[,2] = LOtmp[,2] + offs[i,2]
  plot.hypergraph(gg$gr, mark.col = adjustcolor(gg$cols, 0.3), mark.border = gg$cols,
                  vertex.size = 30, vertex.label.cex = 1, layout = LOtmp,
                  xlim = c(0, 8), ylim = c(0,8), add = i != 1, rescale = FALSE,
                  vertex.color = c(cols$white, cols$black, cols$yellow, cols$lightblue, cols$buff),
                  vertex.label = NA
  )

}
## add the legend
par(xpd = TRUE)
points(c(0, 2, 3,5,7.5),
       rep(9, 5), pch = 21,
       bg = c(cols$white, cols$black, cols$yellow, cols$lightblue, cols$buff),
       cex = 1.5
)
text(c(0, 2, 3,5,7.5), rep(9, 5),
     labels = c("age group", "sex", "ambulance", "hospitalization", "critical"),
     pos = 4
)
par(xpd = FALSE)
```

Figure 3c:

Compute the reduced row echelon form **R** and the transformation matrix **T** from the reduced coefficient matrix $\mathbf{C}'$ for the top scoring set of constraints

```
i <- 1
T <- rref(
  cbind(
    dataMixed$coefficientMatrix[[i]],
    diag(nrow(dataMixed$coefficientMatrix[[i]]))
  )
)
R <- T[, 1:72]
T <- T[, -(1:72)]
```

Get the marginal states and the number of rows

```
source("../R/getMarginals.R")
m <- getMarginals(dataMixed$combinations[[i]], alphas, nn, margByOrder)
marginalStates <- getMarginalsState(dataMixed$combinations[[i]], alphas, nn, margByOrder, L)
rows <- marginalStates$rows
marginalStates <- marginalStates$m
```

Select nonzero rows in **R**

```
sel <- which(rowSums(R != 0) > 0)
R <- R[sel, ]
```

All admissible microstates

```
microstates <- empirical[, 1:5][empirical$count != 0, ]
```

Render the figure

```
par(mar = c(0,0,0,0) + 0.1)
plot(NULL, ylim = c(0, nrow(T) + 2), xlim = c(0,nrow(T) + 1),
     type = "n", frame = FALSE, axes = FALSE, ann = FALSE)

## marginal constraints
cr <- colorRampPalette(c(cols$white, cols$black))(6)
yy <- 1:nrow(marginalStates)
ss <- which(!is.na(marginalStates[, 1]))
points(
  10:14, rep(100.5,5), pch = 21,
  bg = c(cols$white, cols$black, cols$yellow, cols$lightblue, cols$buff),
  cex = 0.7, lwd = 0.5
)
rect(
  10 - 0.5, yy[ss] - 0.5, 10 + 0.5, yy[ss] + 0.5,
  col = cr[marginalStates[ss, 1] + 1], border = cols$grey,
  lwd = 0.2
)
```

```r
cr <- c(cols$white, cols$black)
for (x in 2:5){
  ss <- which(!is.na(marginalStates[, x]))
  rect(
    9 + x - 0.5, yy[ss] - 0.5, 9+x + 0.5, yy[ss] + 0.5,
    col = cr[marginalStates[ss, x] + 1], border = cols$grey,
    lwd = 0.2
  )
}

## labels for the sets of constraints
y1 <- c(0, cumsum(rows)[-length(rows)]) + 0.5
y2 <- cumsum(rows) + 0.5

rect(
  8.5 - 0.25 * (seq_along(y1) - 1), y1, 9 - 0.25 * (seq_along(y1) - 1), y2,
  col = cols$purple, border = NA
)
k <- 0
cr <- c(cols$white, cols$black, cols$yellow, cols$lightblue, cols$buff)
marginalLabel <-
  lapply(
    1:length(dataMixed$combinations[[i]]),
    function(ell){
      m <- NULL
      for(j in dataMixed$combinations[[i]][[ell]]){
        m <- cbind(m, nn[[ell]]$c[, j])
        points(
          8.5 - 0.25 * k - (1:ell) * 1.5, rep((y1 + y2)[k + 1] / 2, ell),
          bg = cr[rev(nn[[ell]]$c[, j])], pch = 21, lwd = 0.5
        )
        k <<- k + 1
      }
      m
    }
  )

## links between marginals and generalized marginal constraints
rr1 <- which(T != 0, arr.ind = TRUE)
segments(
  15 + 0.5, rr1[,2], 55 - 0.5, rr1[,1],
  col = ifelse(T[rr1] > 0, cols$red, cols$blue),
  lwd = 0.2
)

## generalized marginal constraints
yy2 <- 1:length(sel)
rect(
  56 - 0.5, yy2 - 0.5, 56 + 0.5, yy2 + 0.5,
  col = cols$black, border = cols$grey,
  lwd = 0.2
)
yy2 <- (length(sel) + 1):nrow(T)
```

```r
rect(
  56 - 0.5, yy2 - 0.5, 56 + 0.5, yy2 + 0.5,
  col = cols$white, border = cols$grey,
  lwd = 0.2
)

## links between generalized marginal constraints and joint distribution
rr1 <- which(R != 0, arr.ind = TRUE)
segments(
  57 + 0.5, rr1[,1], 96 + 0.5, rr1[,2],
  col = ifelse(R[rr1] > 0, cols$red, cols$blue),
  lwd = 0.2
)

## joint distribution
cr <- colorRampPalette(c(cols$white, cols$black))(6)
yy <- 1:72
rect(
  98 - 0.5, yy - 0.5, 98 + 0.5, yy + 0.5,
  col = cr[microstates[, 1] + 1], border = cols$grey,
  lwd = 0.2
)
cr <- c(cols$white, cols$black)
for (x in 2:5)
  rect(
    97 + x - 0.5, yy - 0.5, 97 + x + 0.5, yy + 0.5,
    col = cr[microstates[, x] + 1], border = cols$grey,
    lwd = 0.2
  )

points(
  98:102, rep(73.5,5), pch = 21,
  bg = c(cols$white, cols$black, cols$yellow, cols$lightblue, cols$buff),
  cex = 0.7, lwd = 0.5
)

## legend
segments(95, 99, 97, 99, col = cols$red)
segments(95, 96, 97, 96, col = cols$blue)
text(97,99, "+1", pos = 4)
text(97,96, "-1", pos = 4)
```