Day-2 Practical Session, 26 May 2021

Part 2: BIGS-IWE strategy for Line Intercept Sampling (LIS)

Li-Chun Zhang[1,2,3] and *Melike Oguz-Alper*[2]

[1]*University of Southampton (L.Zhang@soton.ac.uk)*, [2]*Statistics Norway*, [3]*University of Oslo*

In this illustration, we will apply BIGS-IWE strategy to line-intercept sampling (LIS) which is a method of habitat sampling in a given area, where a habitat is sampled if a chosen line segment transects it. An example of LIS is given by Becker (1991). The sketch of the observed tracks can be obtained by compiling R-function **skthLISBecker**. Visualisation of the BIG constructed based on the observed line-intercept samples can be obtained by using R-function **skthLISBeckerBIG**.

Description of the population and sampling strategies

- Population BIG: $\mathcal{B} = (F, \Omega; H)$, where Ω consists of all wolverine tracks in the region of interest, F contains the corresponding projection segments, and H consist of edges between the tracks and the segments, i.e. $(i\kappa) \in H$ for $i \in F$ and $\kappa \in \Omega$
  - **NB**. $\mathcal{B}$ cannot be consructed unless the whole area observed
- Observed BIG: $\mathcal{B}^* = (F^*, \Omega_s; H^*)$, where Ω_s contains the observed wolverine tracks, F^* contain the projection segments constructed based on the actual samples s and Ω_s, and H^* consists of the incident observational links from F^* to Ω_s
- β^*_κ: *ancestry* set of $\kappa \in \Omega_s$ in $\mathcal{B}^*$ and α^*_i: *successors* of $i \in F^*$ in $\mathcal{B}^*$
- Four systematic samples, A, B, C and D, each containing three positions drawn on the baseline. Equal distances of 12 miles between each position in a given draw
- Four wolverine tracks observed: $\Omega_s = \{\kappa_1, \kappa_2, \kappa_3, \kappa_4\}$; $y_\kappa = (1, 2, 2, 1)^\top$ and $L_\kappa = (5.25, 7.5, 2.4, 7.05)^\top$
- The baseline divided into seven projection segments given the tracks: $F^* = \{i_1, i_2, i_3, i_4, i_5, i_6, i_7\}$
- The probability that the ith segment selected under systematic sampling: $p_i = x_i/12$, where $\mathbf{x} = (5.25, 2.25, x_3, 2.4, x_5, 7.05, x_7)^\top$
- We have $s_1 = s_2 = \{i_1, i_5, i_6\}$, yielding $\Omega_s = \{\kappa_1, \kappa_2, \kappa_4\}$, and $s_3 = s_4 = \{i_4, i_6, i_7\}$, yielding $\Omega_s = \{\kappa_3, \kappa_4\}$

Formula sheet

- The parameter of interest: total number of wolverines in the region of interest

  $\theta = \sum_{\kappa \in \Omega} y_\kappa$, where y_κ number of wolverines in track κ

- Hansen-Hurwitz (HH) type estimators on the rth draw

  $\hat\theta_r = \sum_{i \in s_r} \frac{z_i}{p_i}$, where $z_i = \sum_{\kappa \in \alpha^*_i} w_{i\kappa} y_\kappa$

  - *Multiplicity* estimator; equal weights

    $w_{i\kappa} \equiv \frac{1}{|\beta^*_\kappa|}$

  - HH-type estimator with *unequal* weights: *probability and inverse degree-adjusted (PIDA) weights*

    $w_{i\kappa} = \frac{p_i}{|\alpha^*_i|^\gamma} \left(\sum_{i \in \beta^*_\kappa} \frac{p_i}{|\alpha^*_i|^\gamma} \right)^{-1}, \gamma \geq 0$

NB. When $\gamma = 0$, PIDA weights reduce to $w_{i\kappa} = p_i / p_{(\kappa)}$, where $p_{(\kappa)} = \sum_{i \in \beta^*_\kappa} p_i$

- HH-type estimators over all the draws

  $\hat\theta = \frac{1}{4} \sum_{r=1}^4 \hat\theta_r$

- Variance estimator of the HH-type estimators over all the draws

  $\hat{\mathrm{V}}(\hat\theta) = \frac{1}{4} \frac{\sum_{r=1}^4 (\hat\theta_r - \hat\theta)^2}{r-1}$

NB. R-package **igraph** has to be installed before running R-functions below that generates bipartite graphs.

Description of R-function **skthLISBeckerBIG**

1. Function parameters

- **showplot**: Use **TRUE** to get the skecth of BIG; default **FALSE**

2. Main steps of the function

- A bipartite graph constructed based on the observed wolverine tracks and the sample line segments transecting them. R-package **igraph** used to generate the graph

3. Main outputs of the function

- BIG plot shown if **showplot**= **TRUE**
- The bipartite graph generated is returned as a graph object. It shall be called via **$G**

Description of R-function **zLISBecker**

1. Function parameters

- **graphstar**: graph to be used: the output of **skthLISBeckerBIG**
- **coefgamma**: coefficient to be used in the HH-type estimator with PIDA weights; default value 0. No effect of the choice if **multiplicity**= **TRUE**
- **probi**: a vector of the selection probabilities of the constructed projection segments
- **multiplicity**: Use **TRUE** to get z_i values based on equal weights, i.e. $w_{i\kappa} =| \beta^*_\kappa|^{-1}$; default **FALSE**

2. Main steps of the function

- Edge set derived from the input graph, as well as the labels of the vertices in F^* and Ω_s
- $|\alpha^*_i|$ and $|\beta^*_\kappa|$ calculated based on the edge set
- z_i values calculated for all $i \in F^*$ for chosen values of γ

3. Main outputs of the function

- z_i values returned

Description of R-function **mainLISBecker**

1. Function parameters

- **graphstar**: graph to be used: the output of **skthLISBeckerBIG**
- **coefgamma**: coefficient to be used in the HH-type estimator with PIDA weights; default value 0. No effect of the choice if **multiplicity**= **TRUE**
- **probi**: a vector of the selection probabilities of the constructed projection segments
- **multiplicity**: Use **TRUE** to get z_i values based on equal weights, i.e. $w_{i\kappa} =| \beta^*_\kappa|^{-1}$; default **FALSE**
- **showcat**: Use **FALSE** to avoid printing outputs from the *cat* function in R; default **TRUE**

2. Main steps of the function

- Edge set derived from the input graph
- $|\alpha^*_i|$ and $|\beta^*_\kappa|$ calculated based on the edge set
- z_i values obtained by calling function **zLISBecker**
- For each draw under systematic sampling, estimates obtained by using the HH-type estimator.
- The HH-type estimator over all draws applied by taking the average of the estimates. Variance of the estimator calculated

3. Main outputs of the function

- An estimate for the total number of wolverine tracks in the given area and its estimated variance. Variance estimate can be called via **$varest** for further analysis.

In [1]:

```
# load R-package igraph
library(igraph)
```

```
Warning message:
"package 'igraph' was built under R version 3.6.3"
Attaching package: 'igraph'

The following objects are masked from 'package:stats':

    decompose, spectrum

The following object is masked from 'package:base':

    union
```

In [2]:

```
# Skecth of the wolverine tracks, Becker (1991)
skthLISBecker <- function()
{
  plot(0,xaxt="n",yaxt="n",type="l",ylab="",xlab="Baseline",xlim=c(0,120),ylim=c(0,60)
  lines(c(0,0),c(0,60)); lines(c(0,120),c(0,0)); lines(c(0,120),c(60,60)); lines(c(12
  lines(c(0,10,12,19),c(40,40,36,42),lty=2); text(19,45,labels="k1")
  lines(c(0,5,15,25),c(15,18,14,18),lty=2); text(20,12,labels="k2")
  lines(c(31,34,38),c(25,30,20),lty=2); text(29,30,labels="k3")
  lines(c(75,68,85,90),c(10,15,20,15),lty=2); text(86,23,labels="k4")
  abline(v=c(2,42,82)); text(2,4,label="A1"); text(42,4,label="A2"); text(82,4,label='
  abline(v=c(10,50,90)); text(10,1,label="B1"); text(50,1,label="B2"); text(90,1,label
  abline(v=c(35,75,115)); text(35,3,label="C1"); text(75,3,label="C2"); text(115,3,lab
  abline(v=c(38,78,118)); text(38,-1,label="D1"); text(78,-1,label="D2"); text(118,-1,

}
```

In [3]:

```
# Skecth of BIG representation of LIS, Becker (1991)
skthLISBeckerBIG <- function(showplot=FALSE)
{
  # Projection of wolverine tracks on the baseline;
  # Areas without tracks assigned length of 1
  idx_F <- paste('i',1:7,sep='')
  idx_omega <- paste('k',1:4,sep='')
  edgeik <- data.frame(i=c('i1','i1','i2','i4','i6'),k=c('k1','k2','k2','k3','k4'))
  g <- graph_from_data_frame(edgeik, directed = TRUE)
  g <- add_vertices(g,3,attr=list(name=c('i3','i5','i7')))

  # Apply bipartite layout
  LO_bipart <- layout_as_bipartite(g,types=bipartite_mapping(g)$type)
  LO_bipart[bipartite_mapping(g)$type==FALSE,2] <- 0
  LO_bipart[bipartite_mapping(g)$type==TRUE,2] <- 1

  nodecolor <- rep("yellow",length(V(g)))
  nodecolor[bipartite_mapping(g)$type==TRUE] <- "orange"

  # Plot BIG
  if(showplot){
    plot(g, vertex.label=V(g)$name, vertex.size=10,vertex.label.dist=0,vertex.label.ce
         vertex.color=nodecolor, layout=LO_bipart[,2:1]) }

return(list(G=g))
}
```

In [4]:

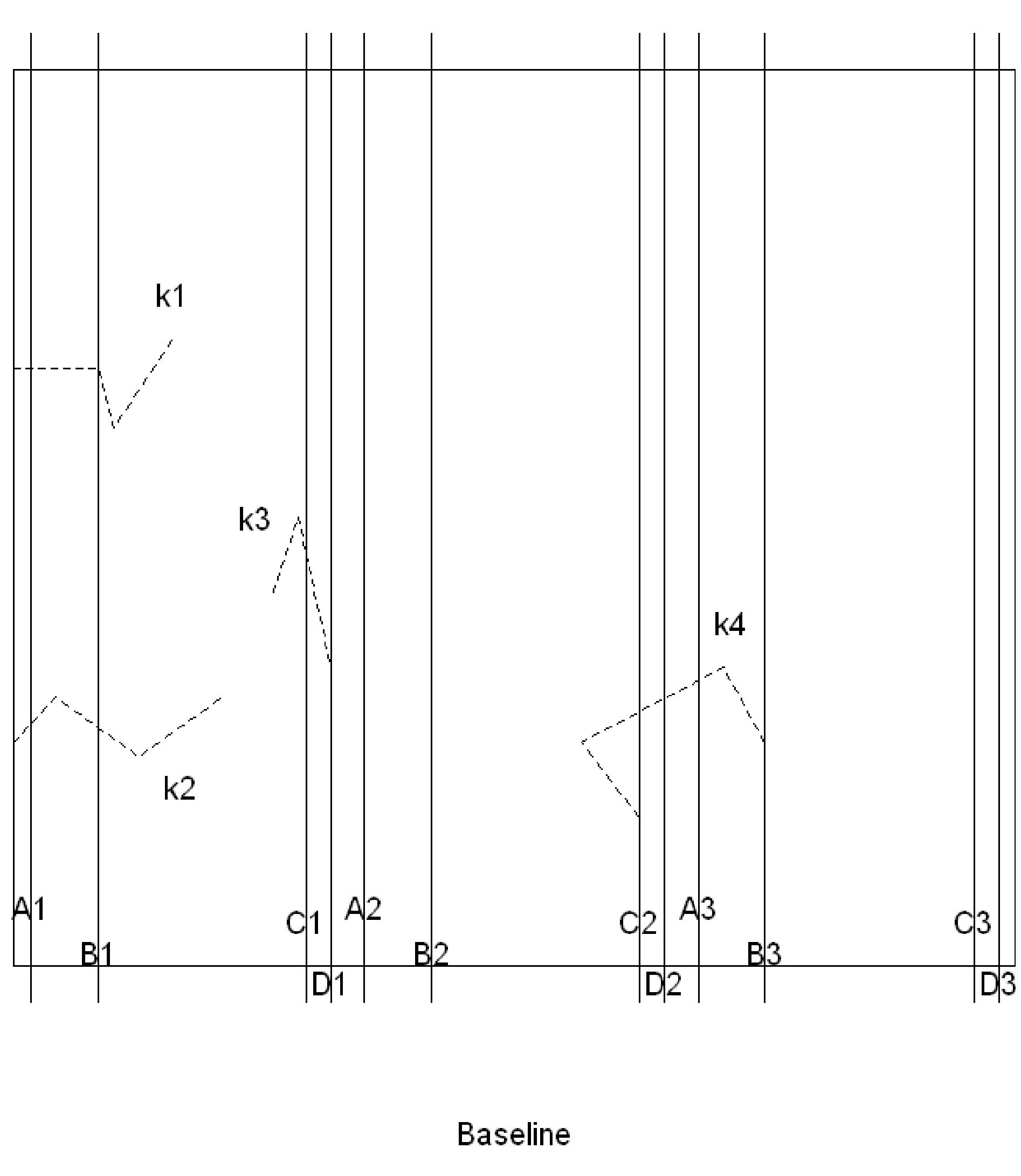
```
# zi-values
zLISBecker <- function(graphstar,coefgamma=0,probi,multiplicity=FALSE){
  edgeik <- data.frame(as_edgelist(graphstar))
  colnames(edgeik) <- c('i','k')
  idx_F <- as_ids(V(graphstar)[bipartite.mapping(graphstar)$type==FALSE])
  idx_F <- idx_F[order(idx_F)]
  idx_omega <- as_ids(V(graphstar)[bipartite.mapping(graphstar)$type==TRUE])
  card_alphai <- NULL
  for(i in idx_F){
    card_alphai <- c(card_alphai,sum(edgeik$i %in% i))
  }

  card_betak <- NULL
  for(k in idx_omega){
    card_betak <- c(card_betak,sum(edgeik$k %in% k))
  }
  yk <- c(1,2,2,1)
  zi <-  NULL
  for(i in idx_F){
    if(i %in% edgeik$i){
      tmp.k <- edgeik$k[edgeik$i %in% i]
      if(multiplicity){tmp.zi <- sum(yk[idx_omega %in% tmp.k]/card_betak[idx_omega %ir
      if(!multiplicity){
        tmp.zi <- 0
        for(k in tmp.k){
          betak <- edgeik$i[edgeik$k %in% k]
          wik <- (probi[idx_F==i]*(1/(card_alphai[idx_F==i])^coefgamma)/(sum(probi[idx
          tmp.zi <- tmp.zi + yk[idx_omega==k]*wik
        }
      }
    }
    if(!(i %in% edgeik$i)){
      tmp.zi <- 0}
    zi <- c(zi,tmp.zi)
  }
  return(zi)
}
```

In [5]:

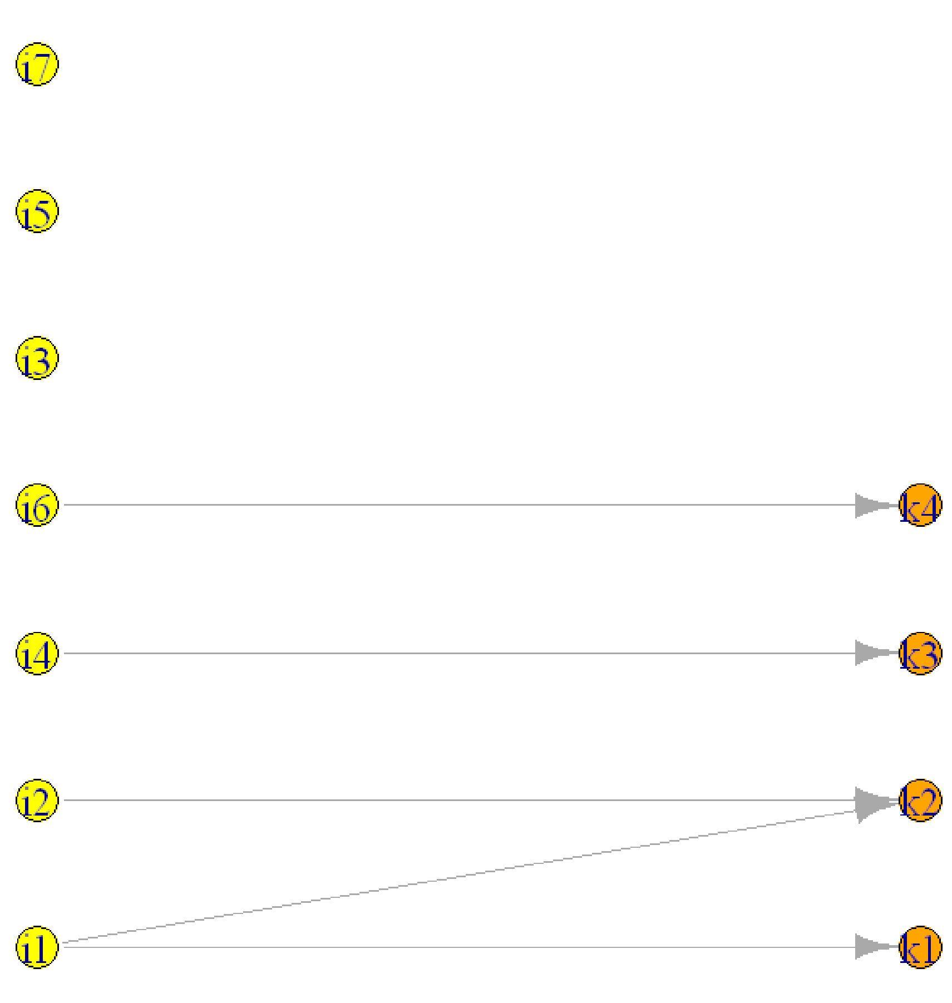
```
# HH-estimators over all draws
mainLISBecker <- function(graphstar,coefgamma=0,probi,multiplicity=FALSE,showcat=TRUE)
  edgeik <- data.frame(as_edgelist(graphstar))
  colnames(edgeik) <- c('i','k')
  idx_F <- as_ids(V(graphstar)[bipartite.mapping(graphstar)$type==FALSE])
  idx_F <- idx_F[order(idx_F)]
  idx_omega <- as_ids(V(graphstar)[bipartite.mapping(graphstar)$type==TRUE])

all.subsets <- list(c(1,5,6),c(1,5,6),c(4,6,7),c(4,6,7))
B <- length(all.subsets)

# Estimates over random samples
  YhatHH_alpha <- NULL
  for(b in 1:B){
    s0 <- idx_F[all.subsets[[b]]]
    s1 <- unique(edgeik$k[edgeik$i %in% s0])
    zi_alpha <- zLISBecker(graphstar,coefgamma,probi,multiplicity = multiplicity)
    YhatHH_alpha <- c(YhatHH_alpha,sum(zi_alpha[idx_F %in% s0]/probi[idx_F %in% s0]))
  }

if(showcat){
  cat('g:',coefgamma,'\t','zi:',zi_alpha,"\n")

  cat("Estimate per draw: ",YhatHH_alpha, "\n")
  cat("(Estimate over all draws,VarEst): ",mean(YhatHH_alpha), '\t',var(YhatHH_alpha),
}
  return(list(varest=var(YhatHH_alpha)/B))
}
```

In [6]:

```
skthLISBecker()
skthLISBeckerBIG(showplot=TRUE)
```



```
$G
IGRAPH d2aeb8b DN-- 11 5 --
+ attr: name (v/c)
+ edges from d2aeb8b (vertex names):
[1] i1->k1 i1->k2 i2->k2 i4->k3 i6->k4
```



In [11]:

```
gstar <- skthLISBeckerBIG()$G
```

FALSE

In [8]:

```
xi <- c(5.25,7.5-5.25,1,2.4,1,7.05,1)
pi <- xi/12
coefg <- 0
cat('g:',coefg,'\t','zi_alpha:',zLISBecker(gstar,coefg,pi),"\n",'\t','zi_beta:',zLISBe
```

```
g: 0     zi_alpha: 2.4 0.6 0 2 0 1 0
         zi_beta: 2 1 0 2 0 1 0
```

In [9]:

```
mainLISBecker(gstar,probi=pi,multiplicity=TRUE)
```

```
g: 0     zi: 2 1 0 2 0 1 0
Estimate per draw:  6.273556 6.273556 11.70213 11.70213
(Estimate over all draws,VarEst):  8.987842      2.455782
```

$varest = 2.45578231292517

In [10]:

```
mainLISBecker(gstar,probi=pi)
```

```
g: 0     zi: 2.4 0.6 0 2 0 1 0
Estimate per draw:  7.187842 7.187842 11.70213 11.70213
(Estimate over all draws,VarEst):  9.444985      1.698231
```

$varest = 1.69823129251701

In [11]:

```
mainLISBecker(gstar,0.5,probi=pi)
```

```
g: 0.5   zi: 2.245259 0.754741 0 2 0 1 0
Estimate per draw:  6.834148 6.834148 11.70213 11.70213
(Estimate over all draws,VarEst):  9.268138      1.974769
```

$varest = 1.97476856050939

In [12]:

```
mainLISBecker(gstar,1,probi=pi)
```

```
g: 1     zi: 2.076923 0.9230769 0 2 0 1 0
Estimate per draw:  6.44938 6.44938 11.70213 11.70213
(Estimate over all draws,VarEst):  9.075754      2.299279
```

$varest = 2.29927947510365

In [10]:

```
# Variance estimates of zi_alpha for different choices of gamma
max.gamma <- 15
range.gamma <- seq(0,max.gamma,by=0.1)
varest_gamma <- NULL
for(tmp.gamma in range.gamma){tmp <- mainLISBecker(gstar, coefgamma=tmp.gamma,probi=p
varest_gamma <- c(varest_gamma,tmp)                          }
# Gamma value which gives minimum variance
par(mfrow=c(1,1),xpd=FALSE)
plot(range.gamma,varest_gamma,xlab='gamma',ylab='varest')
varest_multiplicity <- mainLISBecker(gstar,coefgamma=tmp.gamma,probi=pi,multiplicity =
abline(h=varest_multiplicity)
text(max.gamma*0.90,varest_multiplicity *0.98,label='varest(zi_beta)',pos=1)
gamma.multiplicity <- range.gamma[which(abs(varest_gamma-varest_multiplicity)==min(abs
abline(v=gamma.multiplicity,lty=2)
text(gamma.multiplicity*1.1,max(varest_gamma)*0.99,label=paste('gamma.multiplicity=',g
mainLISBecker(gstar,gamma.multiplicity,probi=pi)
mainLISBecker(gstar,probi=pi,multiplicity=TRUE)
```

```
g: 1.2   zi: 2.00776 0.9922395 0 2 0 1 0
Estimate per draw:  6.291294 6.291294 11.70213 11.70213
(Estimate over all draws,VarEst):  8.996711      2.43976
```

$varest = 2.43975967827738

```
g: 0     zi: 2 1 0 2 0 1 0
Estimate per draw:  6.273556 6.273556 11.70213 11.70213
(Estimate over all draws,VarEst):  8.987842      2.455782
```

$varest = 2.45578231292517