```
### ------------------------------------------------------------------------------
### A package for CAR priors: CARBayes
###  https://CRAN.R-project.org/package=CARBayes
### ------------------------------------------------------------------------------
library(CARBayes)
```

```
library(spdep)
```

```
library(maptools)
```

```
library(sp)
library(CARBayesdata)
```

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Example 1 - Scottish lip cancer data
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
# Scottish lip cancer data set, which is included purely to
# illustrate how to combine a dataframe and shapefile together
# into a "SpatialPolygonsDataFrame" object.
# The creation of this object allows spatial maps to be produced
# of variables of interest, as well as allowing the neighbourhood
# matrix W to be created
data(lipdata) # data you wish to model (lipdata is included in the CARbayesdata package)
head(lipdata)
```

```
##   observed expected pcaff latitude longitude
## 1        9      1.4    16    57.29      5.50
## 2       39      8.7    16    57.56      2.36
## 3       11      3.0    10    58.44      3.90
## 4        9      2.5    24    55.76      2.40
## 5       15      4.3    10    57.71      5.09
## 6        8      2.4    24    59.13      3.25
```

```
rownames(lipdata)
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15"
## [16] "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30"
## [31] "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42" "43" "44" "45"
## [46] "46" "47" "48" "49" "50" "51" "52" "53" "54" "55" "56"
```

```
# -- Observed: The number of recorded lip cancer cases in each district (a total of 56 districts)
# -- Expected: the expected number of lip cancer cases in each
#    district based on the population size, its age and sex structure in each district.
# -- pcaff: percentage of the workforce in each district employed in agriculture, fishing and forestry.
```

```
# The shapefile format is a popular geospatial vector data format
# for geographic information system (GIS) software.
# The shapefile format can spatially describe vector features:
# points, lines, and polygons, representing,
# for example, water wells, rivers, and lakes.
# Each item usually has attributes that describe it, such as name or temperature.

data(lipdbf)  # shapefile (.shp, .dbf)
data(lipshp)
names(lipdbf)
```

```
## [1] "dbf"    "header"
```

```
names(lipshp)
```

```
## [1] "shp"    "header"
```

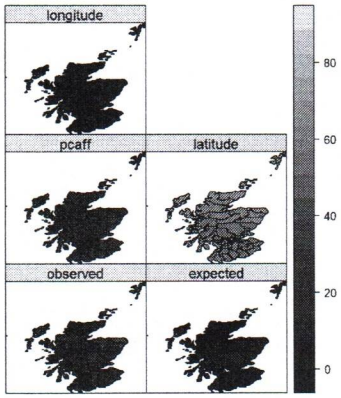
```
# These three data sets can be combined together to create a
# "SpatialPolygonsDataFrame" object:
names(lipdbf$dbf)
```

```
## [1] "NAME" "ID"
```

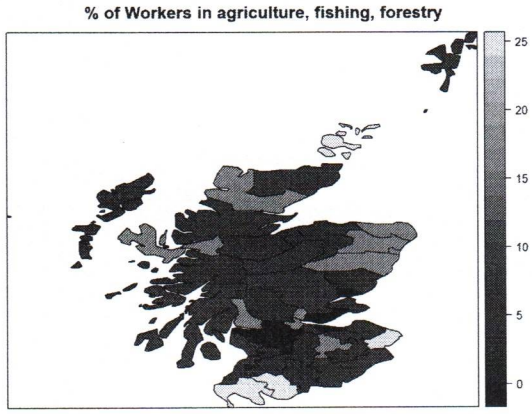
```
lipdbf$dbf    <- lipdbf$dbf[ , c(2,1)] # invert the columns for convenience
data.combined <- combine.data.shapefile(data=lipdata, shp=lipshp, dbf=lipdbf)
is(data.combined)
```

```
## [1] "SpatialPolygonsDataFrame" "SpatialPolygons"
## [3] "Spatial"                  "SpatialVector"
```

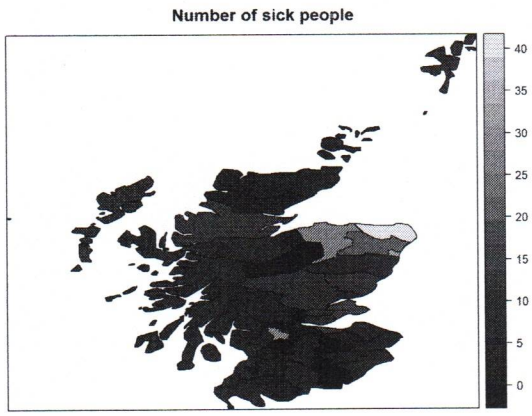
```
# For this function to work the rownames of the dataframe (lipdata)
# must be contained in the first column of the .dbf (lipdbf$dbf)
# object, which is the reason for reordering the columns in
# the first line of the above code.
# The data.combined object is a "SpatialPolygonsDataFrame" object
x11()
spplot(data.combined) # Note that the colors follow a unique scale
```




```
spplot(data.combined[, "pcaff"], main = "% of Workers in agriculture, fishing, forestry" )
```




```
spplot(data.combined[, "observed"], main = "Number of sick people" )
```




```
# A binary neighbourhood matrix W can be constructed for the
# data from this object using the code:
W.nb  <- poly2nb(data.combined, row.names = rownames(lipdata)) # neighbourhood
is(W.nb)
```

```
## [1] "nb"
```

```
W.mat <- nb2mat(W.nb, style = "B")
is(W.mat)
```

```
## [1] "matrix"    "array"     "mMatrix"   "structure" "vector"
```

```
dim(W.mat)     # matrix of dimension 56 x 56
```

```
## [1] 56 56
```

```
W.mat[1:9,1:9] # proximity matrix to be used for fitting CARBayes models to the dataset
```

```
##   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## 1    0    0    0    0    1    0    0    0    1
## 2    0    0    0    0    0    0    1    0    0
## 3    0    0    0    0    0    1    0    0    1
## 4    0    0    0    0    0    0    0    0    0
## 5    1    0    0    0    0    0    0    0    0
## 6    0    0    1    0    0    0    0    1    0
## 7    0    1    0    0    0    0    0    0    0
## 8    0    0    0    0    0    1    0    0    0
## 9    1    0    1    0    0    0    0    0    0
```

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Example 2 - property prices in Greater Glasgow
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
# Modelling the spatial pattern in average property prices
# across Greater Glasgow, Scotland, in 2008.
# This is a regression analysis for areal data, whose aim is to
# identify the factors that affect property prices and quantify
# their effects.
```

```
# The study region is the Greater Glasgow and Clyde health board
# (part of Scotland), which is split into 270 intermediate geographies (IG)
# These IGs are small areas that have a median area of 124 hectares
# and a median population of 4,239
data(GGHB.IG)
data(pricedata)
summary(pricedata)
```

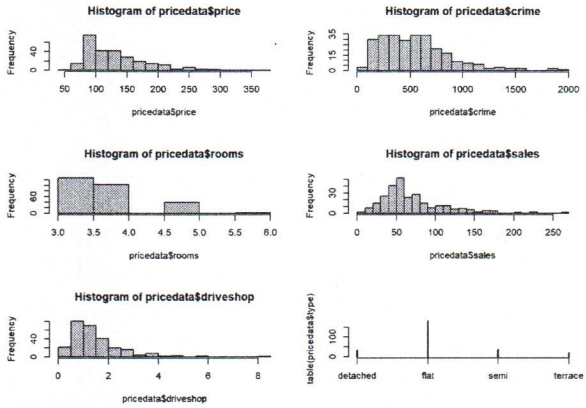
```
##          IG          price           crime            rooms
##  S02000260:  1   Min.   : 50.0   Min.   :  85.0   Min.   :3.000
##  S02000261:  1   1st Qu.: 95.0   1st Qu.: 303.2   1st Qu.:3.000
##  S02000262:  1   Median :121.8   Median : 517.0   Median :4.000
##  S02000263:  1   Mean   :134.8   Mean   : 555.1   Mean   :3.681
##  S02000264:  1   3rd Qu.:159.2   3rd Qu.: 728.0   3rd Qu.:4.000
##  S02000265:  1   Max.   :372.8   Max.   :1994.0   Max.   :6.000
##  (Other)  :264
##      sales          driveshop           type
##  Min.   :  4.00   Min.   :0.300   detached: 31
##  1st Qu.: 46.00   1st Qu.:0.850   flat    :183
##  Median : 58.00   Median :1.250   semi    : 36
##  Mean   : 70.88   Mean   :1.534   terrace : 20
##  3rd Qu.: 84.75   3rd Qu.:1.887
##  Max.   :266.00   Max.   :8.500
##
```

```
head(pricedata)
```

```
##          IG   price crime rooms sales driveshop     type
## 1 S02000260 112.250   390     3    68       1.2     flat
## 2 S02000261 156.875   116     5    26       2.0     semi
## 3 S02000262 178.111   196     5    34       1.7     semi
## 4 S02000263 249.725   146     5    80       1.5 detached
## 5 S02000264 174.500   288     4    60       0.8     semi
## 6 S02000265 163.521   342     4    24       2.5     semi
```

```
# price: median property price in each IG
# crime: the crime rate (number of crimes per 10,000 people) in each IG
# rooms: the median number of rooms in a property in each IG
# sales: the percentage of properties that sold in each IG in a year
# driveshop: the average time taken to drive to a shopping centre in minutes
# type: the predominant property type in each IG with levels:
# detached, semi, terraced, flat
# casa "singola", "bifamiliare", case a schiera, appartamento
```

```
x11()
par(mfrow = c(3,2))
hist(pricedata$price, nclass = "fd")
hist(pricedata$crime, nclass = "fd")
hist(pricedata$rooms, nclass = "fd")
hist(pricedata$sales, nclass = "fd")
hist(pricedata$driveshop, nclass = "fd")
plot(table(pricedata$type), type = "h")
```

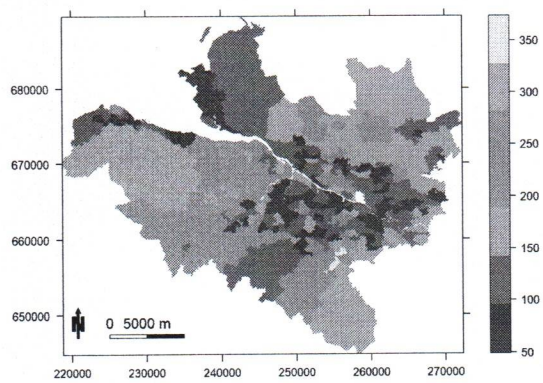



```
# The response variable in this study is "price", i.e.
# the median price (in thousands) of all properties sold in 2008 in each IG
graphics.off()
```

```
# The pricedata object is a data.frame containing the property price data,
# but has data for 270 of the 271 IGs in GGHB, as one area had
# outlying values and was hence removed.
# Thus GGHB.IG needs to have the appropriate row removed and then
# be combined with the pricedata data.frame. This is achieved
# using the code below:
```

```
missing.IG                <- setdiff(rownames(GGHB.IG@data), pricedata$IG)
missing.IG.row            <- which(missing.IG == rownames(GGHB.IG@data))
propertydata.spatial      <- GGHB.IG[-missing.IG.row, ]
propertydata.spatial@data <- data.frame(propertydata.spatial@data, pricedata)
# The first two lines identify the missing row and its row number, while the third line subsets
# the GGHB.IG object by removing that row to create the propertydata.spatial object. Finally,
# the last line adds the data contained in pricedata to the propertydata.spatial object.
```

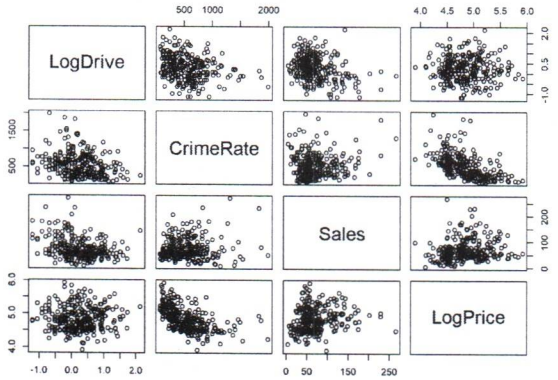
```
### Spatial map:
northarrow  <- list("SpatialPolygonsRescale", layout.north.arrow(),
                    offset = c(220000,647000), scale = 4000)
scalebar    <- list("SpatialPolygonsRescale", layout.scale.bar(),
                    offset = c(225000,647000), scale = 10000, fill=c("transparent","black"))
text1       <- list("sp.text", c(225000,649000), "0")
text2       <- list("sp.text", c(230000,649000), "5000 m")
breakpoints <- seq(min(pricedata$price)-1, max(pricedata$price)+1, length.out=8)
x11()
spplot(propertydata.spatial, c("price"), sp.layout=list(northarrow, scalebar, text1, text2),
       scales=list(draw = TRUE), at=breakpoints,
       col.regions = terrain.colors(n=length(breakpoints)-1), col="transparent")
```




```
# The figure suggests that Glasgow has a number of property
# sub-markets, whose prices are not related to those in  neighbouring areas.
```

```
# Non-spatial modelling
# Initial plots of the data using the pairs() command suggest
# that the log of drive time to a shopping centre is linearly
# related to the response, and
# that crime rate
# has a non-linear relationship to the response
```

```
propertydata.spatial@data$logprice     <- log(propertydata.spatial@data$price)
propertydata.spatial@data$logdriveshop <- log(propertydata.spatial@data$driveshop)

x11()
pairs(cbind(propertydata.spatial@data$logdriveshop, propertydata.spatial@data$crime,
            propertydata.spatial@data$sales, propertydata.spatial@data$logprice),
      labels = c("LogDrive", "CrimeRate", "Sales", "LogPrice"))
```




```
# A frequentist model with all the covariates is fitted to the data,
# where the crime rate variable is modelled
# as non-linear using a natural cubic spline with 3 degrees of freedom.
library(splines)
form  <- logprice ~ ns(crime, 3) + rooms + sales + factor(type) + logdriveshop
model <- lm(formula=form, data=propertydata.spatial@data)
summary(model)
```

```
##
## Call:
## lm(formula = form, data = propertydata.spatial@data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91319 -0.15992  0.00136  0.15647  0.81675
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.436135   0.157971  28.082  < 2e-16 ***
## ns(crime, 3)1        -0.358967   0.089006  -4.033 7.24e-05 ***
## ns(crime, 3)2        -0.617084   0.165152  -3.736 0.000229 ***
## ns(crime, 3)3        -0.299454   0.126516  -2.367 0.018670 *
## rooms                 0.193827   0.029268   6.623 2.02e-10 ***
## sales                 0.002034   0.000362   5.619 4.93e-08 ***
## factor(type)flat     -0.215967   0.066412  -3.252 0.001298 **
## factor(type)semi     -0.153610   0.057750  -2.660 0.008301 **
## factor(type)terrace -0.280023   0.072634  -3.855 0.000146 ***
## logdriveshop         -0.089084   0.025588  -3.482 0.000585 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2243 on 260 degrees of freedom
## Multiple R-squared:  0.6206, Adjusted R-squared:  0.6075
## F-statistic: 47.26 on 9 and 260 DF,  p-value: < 2.2e-16
```

```
# A Moran's I permutation test for spatial autocorrelation was
# then applied to the residuals from this model based on 10,000
# random permutations, using the functionality of the spdep package.
# The first two lines turn the "SpatialPolygonsDataFrame" object
# spatialhousedata into an "nb" and then a "listw"
W.nb        <- poly2nb(propertydata.spatial, row.names = rownames(propertydata.spatial@data))
W.list      <- nb2listw(W.nb, style = "B")
resid.model <- residuals(model)
moran.mc(x = resid.model, listw = W.list, nsim=1000)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  resid.model
## weights: W.list
## number of simulations + 1: 1001
##
## statistic = 0.2733, observed rank = 1001, p-value = 0.000999
## alternative hypothesis: greater
```

```
# H_0: no spatial autocorrelation, H_1: positive spatial autocorrelation
# The Moran's I statistic equals 0.2733, with p-value=0.000099,
# which suggests that the residuals contain substantial positive
# spatial autocorrelation
```

```
# NOTE: Like a correlation coefficient, the values of Moran's I
# range from +1 (strong positive spatial autocorrelation) to 0
# (random pattern) to -1 (strong negative spatial autocorrelation).
# The test is based on MC sampling from the set of all
# permutations of the positions at random
```

```
### Spatial modelling with CARBayes
W <- nb2mat(W.nb, style="B")
W[1:8,1:8]
```

```
##           [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## S02000260    0    1    1    0    1    0    0    0
## S02000261    1    0    1    0    0    0    0    0
## S02000262    1    1    0    0    1    0    0    0
## S02000263    0    0    0    0    0    1    1    0
## S02000264    1    0    1    0    0    0    0    0
## S02000265    0    0    0    1    0    0    0    0
## S02000266    0    0    0    1    0    0    0    0
## S02000267    0    0    0    0    0    0    0    0
```

```
model.spatial <- S.CARleroux(formula=form,
                  data=propertydata.spatial@data,
                  family="gaussian", W = W,
                  burnin = 20000,
                  n.sample = 120000, thin = 10)
```

```
print(model.spatial)
```

```
##
## #################
## #### Model fitted
## #################
## Likelihood model - Gaussian (identity link function)
## Random effects model - Leroux CAR
## Regression equation - logprice ~ ns(crime, 3) + rooms + sales + factor(type) + logdriveshop
## Number of missing observations - 0
##
## #############
## #### Results
## #############
## Posterior quantities and DIC
##
##                       Median    2.5%   97.5% n.effective Geweke.diag
## (Intercept)           4.2388  3.9655  4.5228      9035.5        -1.0
## ns(crime, 3)1        -0.2463 -0.4003 -0.0941      8883.9         0.7
## ns(crime, 3)2        -0.4091 -0.7069 -0.1058      8353.5         0.0
## ns(crime, 3)3        -0.2005 -0.4105  0.0028     10000.0        -0.5
## rooms                 0.2203  0.1690  0.2706     10000.0         1.0
## sales                 0.0023  0.0016  0.0029     10000.0         0.8
## factor(type)flat     -0.2466 -0.3669 -0.1274     10000.0         0.1
## factor(type)semi     -0.1606 -0.2640 -0.0601     10000.0         0.5
## factor(type)terrace -0.2914 -0.4216 -0.1654     10000.0         0.0
## logdriveshop         -0.0051 -0.0621  0.0509      6721.1         1.5
## nu2                   0.0249  0.0145  0.0344      2203.1         0.1
## tau2                  0.0410  0.0187  0.0824      1752.8        -0.5
## rho                   0.9418  0.7634  0.9927      3419.9         1.2
##
## DIC =  -144.4427       p.d =  92.24352       LMPL =  57.74
```

```
# NOTE: for the hyperparameters' specification, see the help of the function.
# We are considering default values.
```

```
# The first part of the output is a description of the model that
# was fitted, including the likelihood and random effects specifications,
# as well as the covariates included in the linear predictor.
# The second part summarises selected  parameters, including
# posterior medians and 95 percent credible intervals, the number of samples,
# the acceptance rate, the effective number of independent samples
# and the Geweke convergence diagnostic in the form of a Z-score.
```

```
model.spatial$model
```

```
## [1] "Likelihood model - Gaussian (identity link function)"
## [2] "\nRandom effects model - Leroux CAR\n"
```

```
names(model.spatial)
```

```
##  [1] "summary.results"     "samples"             "fitted.values"
##  [4] "residuals"           "modelfit"            "accept"
##  [7] "localised.structure" "formula"             "model"
## [10] "X"
```

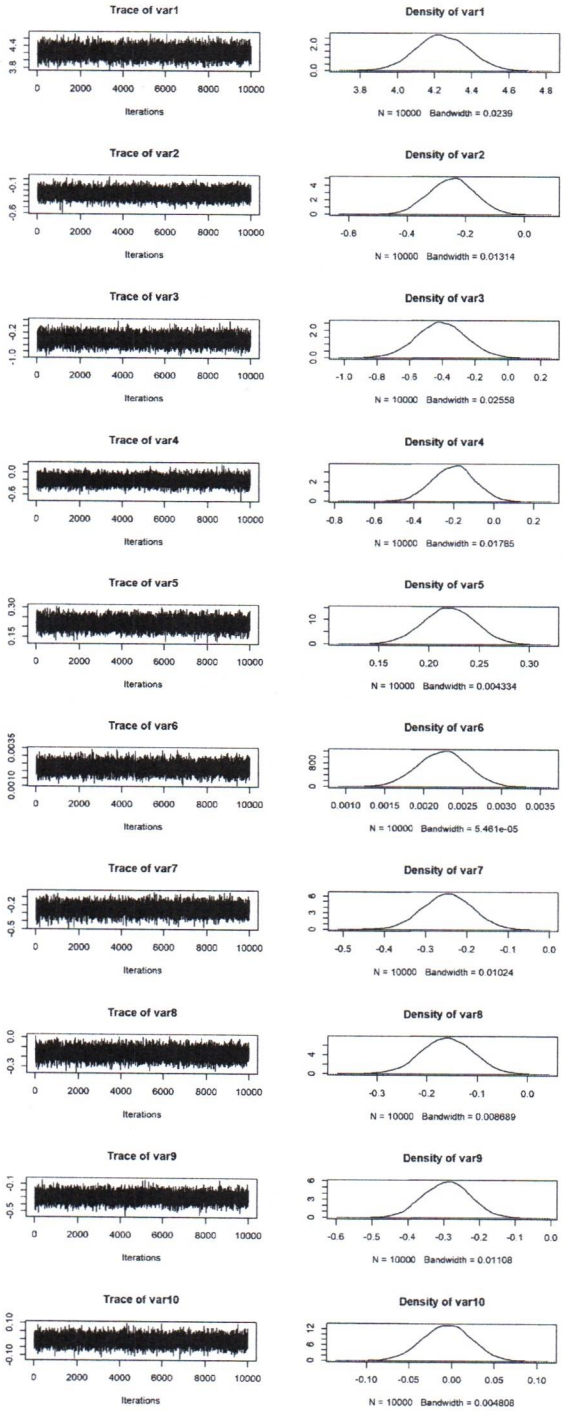
```
model.spatial$modelfit  # Goodness of fit indices
```

```
##           DIC           p.d          WAIC           p.w          LMPL
##   -144.44273      92.24352    -138.22766      80.96328      57.74263
## loglikelihood
##      164.46488
```

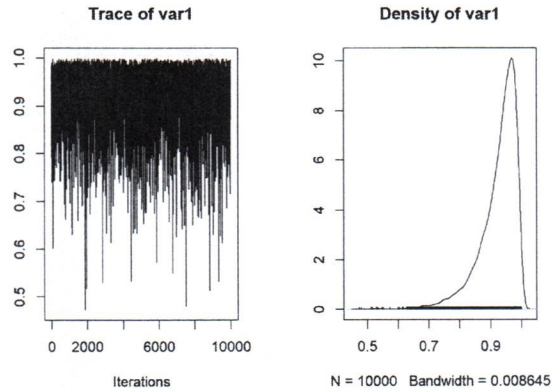
```
names(model.spatial$samples)
```

```
## [1] "beta"   "phi"    "tau2"   "nu2"    "rho"    "fitted" "Y"
```

```
x11()
plot(model.spatial$samples$beta)
```



```
plot(model.spatial$samples$rho)
```




```
# posterior inference suggests a strong spatial correlation

summarise.samples(model.spatial$samples$beta, quantiles=c(0.5, 0.025, 0.975))
```

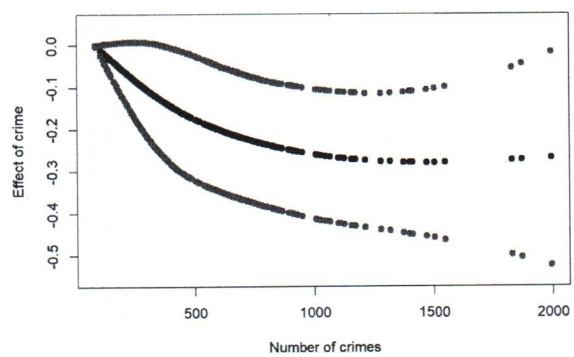
```
## $quantiles
##                0.5        0.025        0.975
##  [1,]  4.238848206  3.965540813  4.522840657
##  [2,] -0.246330648 -0.400312816 -0.094108078
##  [3,] -0.409140173 -0.706935907 -0.105801251
##  [4,] -0.200468710 -0.410511850  0.002815251
##  [5,]  0.220304334  0.169001564  0.270640051
##  [6,]  0.002251494  0.001614575  0.002884780
##  [7,] -0.246585730 -0.366855805 -0.127412065
##  [8,] -0.160587083 -0.264022789 -0.060087731
##  [9,] -0.291351836 -0.421611163 -0.165405674
## [10,] -0.005132780 -0.062067707  0.050936867
##
## $exceedences
## NULL
```

```
# For the crime variable its relationship is non-linear and summarised by the
# results for all 3 basis functions ns(crime, 3)1, ns(crime, 3)2, ns(crime, 3)3.
crime.effect <- summarise.lincomb(model=model.spatial, columns=c(2,3,4),
                                  quantiles=c(0.5, 0.025, 0.975), distribution=FALSE)
names(crime.effect)
```

```
## [1] "quantiles" "posterior"
```

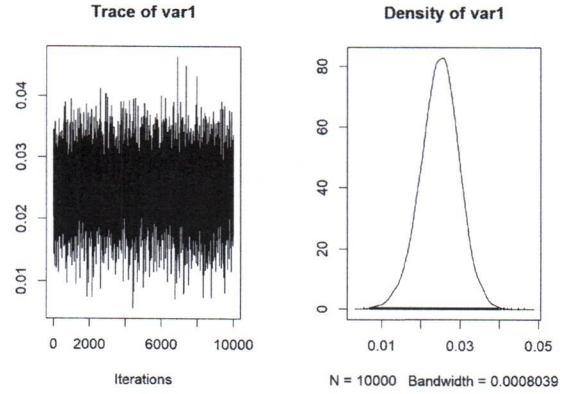
```
# This function computes the posterior distribution and
# posterior quantiles of a linear combination of the covariates from the linear predictor
```

```
x11()
plot(propertydata.spatial@data$crime, crime.effect$quantiles[ ,1], pch=19,
     ylim=c(-0.55,0.05), xlab="Number of crimes", ylab="Effect of crime")
points(propertydata.spatial@data$crime, crime.effect$quantiles[ ,2], pch=19, col="red")
points(propertydata.spatial@data$crime, crime.effect$quantiles[ ,3], pch=19, col="red")
```

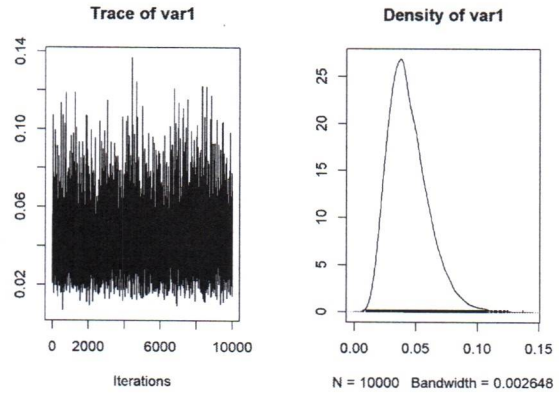



```
# Plot showing the estimated non-linear relationship between
# crime rate and log-price.
# --> Obviously, if the crime rate increases, the log-price decreases!
```

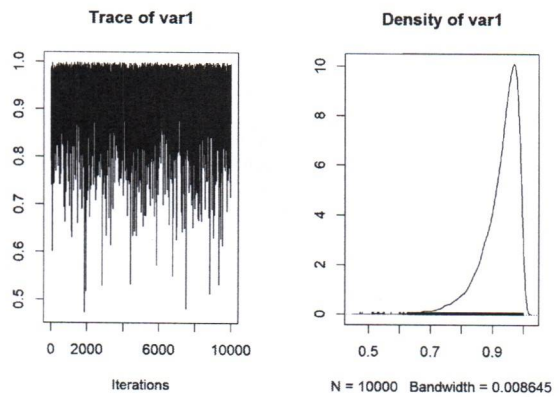
```
x11()
# Posterior of the variance nu^2
plot(model.spatial$samples$nu2)
```



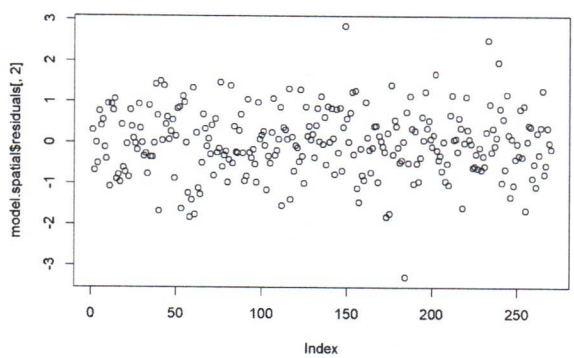

```
plot(model.spatial$samples$tau2)
```



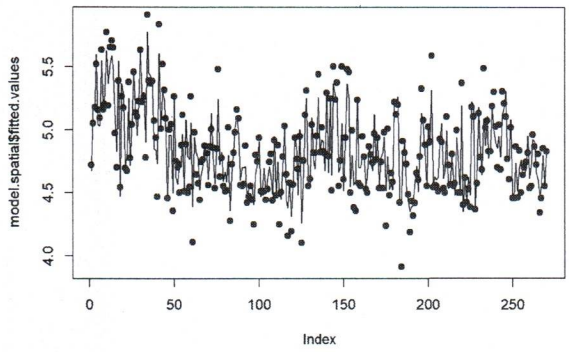

```
plot(model.spatial$samples$rho)
```




```
x11()
plot(model.spatial$residuals[,2]) # nice residuals :) --> it seems that no correlation is left
```




```
plot(model.spatial$fitted.values, type = 'l',ylim=c(3.9,5.9)) # fitted values
points(propertydata.spatial@data$logprice, pch = 19, col = "darkred")
```




```
graphics.off()
```

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Example 3 - identifying high-risk disease clusters
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
# Localised spatial autocorrelation model by Lee and Mitchell (2012):
# it can identify boundaries that represent step changes
# in the (random effects) response surface between geographically adjacent areal units.
# The aim of this analysis is to identify boundaries in the risk
# surface of respiratory disease in Greater Glasgow, Scotland
# in 2010, so that the spatial extent of high-risk clusters can be identified.
```

```
library(CARBayesdata)
data(respiratorydata)
missing.IG     <- setdiff(rownames(GGHB.IG@data), respiratorydata$IG)
missing.IG.row <- rep(NA, length(missing.IG))
for(i in 1:length(missing.IG)){
  missing.IG.row[i] <- which(missing.IG[i] == rownames(GGHB.IG@data))
}
respiratorydata.spatial      <- GGHB.IG[ -missing.IG.row, ]
respiratorydata.spatial@data <- data.frame(respiratorydata.spatial@data, respiratorydata)
head(respiratorydata.spatial@data)
```

```
##                  IG                        name  easting northing      IG.1
## S02000260 S02000260                  Auchinairn 261624.5 669657.4 S02000618
## S02000261 S02000261               Woodhill East 262927.1 670027.8 S02000613
## S02000262 S02000262               Woodhill West 262142.9 670428.0 S02000623
## S02000263 S02000263              Westerton East 254570.5 670593.8 S02000626
## S02000264 S02000264 Bishopbriggs West and Cadder 261248.4 670928.0 S02000636
## S02000265 S02000265              Westerton West 253764.4 670982.6 S02000645
##           observed  expected incomedep       SMR
## S02000260      105 105.12944        15 0.9987687
## S02000261       85  69.41011        22 1.2246054
## S02000262       37  87.85767         8 0.4211357
## S02000263       90  89.41669        26 1.0065235
## S02000264       41  97.55097         8 0.4202931
## S02000265       47  84.86336         8 0.5538315
```

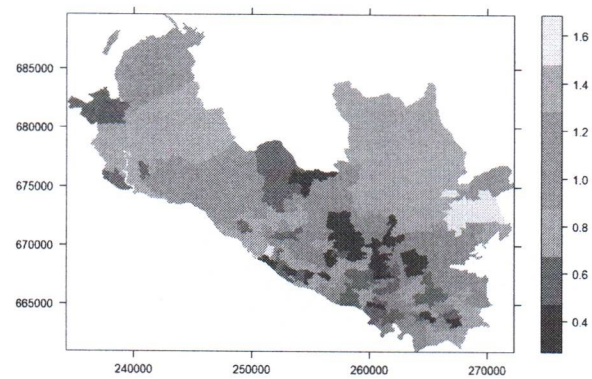
```
dim(respiratorydata.spatial@data)
```

```
## [1] 134   9
```

```
###### Some data are (partially) and we remove them!
# The data set contains the numbers of hospital admissions in 2010
# in each IG due to respiratory disease (International Classification of Disease tenth
# revision codes J00-J99), which is stored in the column "observed".
# However, these observed numbers will depend on the size and demographic
# structure of the populations living in each IG; these factors
# have to be adjusted for, before estimating disease risk. This
# is typically achieved by computing the EXPECTED numbers of
# hospital admissions in each IG based on this demographic
# information, using either internal or external standardisation.
# For these data external standardisation has been used, based on
# age and sex standardised rates for the whole Scotland.
```

```
# IG
# The unique identifier for each IG.
#
# observed
# The observed numbers of hospitalisations due to respiratory disease in each IG.
#
# expected
# The expected numbers of hospitalisations due to respiratory
# disease in each IG computed using indirect standardisation
# from Scotland-wide respiratory hospitalisation rates.
#
# incomedep
# The percentage of people in each IG who are defined to be
# income deprived.
#
# SMR
# The standardised morbidity ratio
# (SMR, observed / expected) for respiratory
# hospitalisation in each IG.
```

```
northarrow  <- list("SpatialPolygonsRescale", layout.north.arrow(), offset =
                    c(220000,647000), scale = 4000)
scalebar    <- list("SpatialPolygonsRescale", layout.scale.bar(), offset =
                    c(225000,647000), scale = 10000, fill=c("transparent","black"))
text1       <- list("sp.text", c(225000,649000), "0")
text2       <- list("sp.text", c(230000,649000), "5000 m")
breakpoints <- seq(min(respiratorydata.spatial@data$SMR)-0.05,
                   max(respiratorydata.spatial@data$SMR)+0.05, length.out=8)
# SMR ---> The standardised morbidity ratio (SMR, observed / expected)
# for respiratory hospitalisation in each IG.
x11()
spplot(respiratorydata.spatial, c("SMR"), sp.layout=list(northarrow, scalebar, text1, text2),
       scales=list(draw = TRUE), at=breakpoints,
       col.regions=terrain.colors(n=length(breakpoints)-1), col="transparent")
```




```
# The simplest measure of disease risk is the Standardised Incidence Ratio (SIR), which is the
# ratio of the observed to the expected numbers of hospital admissions
```

```
W.nb        <- poly2nb(respiratorydata.spatial, row.names = rownames(respiratorydata.spatial@data))
W           <- nb2mat(W.nb, style="B")
income      <- respiratorydata.spatial@data$incomedep
Z.incomedep <- as.matrix(dist(cbind(income, income), method="maximum", diag=TRUE, upper = TRUE))
Z.incomedep[1:9,1:9]
```

```
##    1  2  3  4  5  6  7  8  9
## 1  0  7  7 11  7  7  3 20 19
## 2  7  0 14  4 14 14  4 13 12
## 3  7 14  0 18  0  0 10 27 26
## 4 11  4 18  0 18 18  8  9  8
## 5  7 14  0 18  0  0 10 27 26
## 6  7 14  0 18  0  0 10 27 26
## 7  3  4 10  8 10 10  0 17 16
## 8 20 13 27  9 27 27 17  0  1
## 9 19 12 26  8 26 26 16  1  0
```

```
# Z: matrix of dissimilarity between areal units
# In this example: the absolute difference in the percentage of
# people in each IG who are defined to be income deprived (incomedep)
# Spatial Map for SIR: it gives us an idea of the spatial
# variaton of the risk
```

```
# Income deprivation is a measure of poverty that takes into
# account not only the money, but also services, health, employment...
# The model here assumes that socioeconomic deprivation plays a
# role in determining people's health.
```

```
# The function to implement the localised CAR model is called
# S.CARdissimilarity(), and it takes the same arguments as the
# global CAR models except that it additionally requires
# the dissimilarity metrics. These are required in the form of a
# list of K Ã K matrices, and the
# model is run using the following code.
# MODEL fitted: Y_k \sim Poisson(E_k R_k),
#               where Rk is the disease risk in areal unit k and
#               ln(Rk) =beta_0 + \Phi_k, phi_k spatial random effect
```

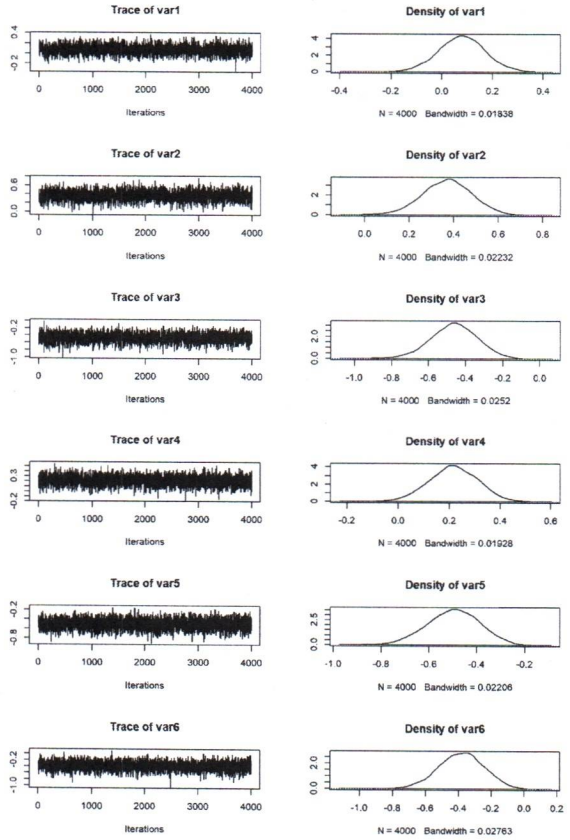
```
formula <- observed ~ offset(log(expected))
model.dissimilarity <- S.CARdissimilarity(formula = formula,
    data=respiratorydata.spatial@data,
    family="poisson", W=W,
    Z=list(Z.incomedep=Z.incomedep), burnin=10000, n.sample=50000,
    thin=10)
```

```
#save(model.dissimilarity, file = "mod_dis.Rdata")
#load("mod_dis.Rdata")
print(model.dissimilarity)
```

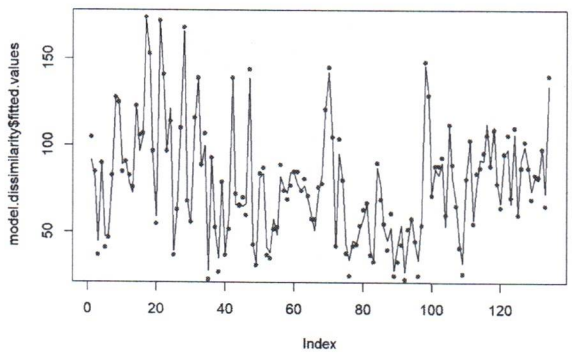
```
##
## #################
## #### Model fitted
## #################
## Likelihood model - Poisson (log link function)
## Random effects model - Binary dissimilarity CAR
## Dissimilarity metrics -  Z.incomedep
## Regression equation - observed ~ offset(log(expected))
## Number of missing observations - 0
##
## #############
## #### Results
## #############
## Posterior quantities and DIC
##
##              Median    2.5%   97.5% n.effective Geweke.diag alpha.min
## (Intercept) -0.2188 -0.2403 -0.1974      3763.5         0.2        NA
## tau2         0.1466  0.0968  0.2623       349.1        -1.1        NA
## Z.incomedep  0.0500  0.0363  0.0513       209.7         0.4    0.0139
##
## DIC =  1074.828       p.d =  105.3803       LMPL =  -584.08
##
## The number of stepchanges identified in the random effect surface
##      no stepchange stepchange
## [1,]           228        132
```

```
# The first line of the above code specifies the formula with an
# offset (the natural log of the expected numbers of cases)
# but no covariates; the extra column of covariates enter in the proximity matric
# Covariate is required so that boundaries
# identified in the random effects surface can also be interpreted as boundaries in the risk sur-
#   face (that is R = (R1 , . . . , Rn ))
```

```
# Some posterior chains:
x11()
plot(model.dissimilarity$samples$phi[,1:6], type='l')
```




```
# The spatial component phi differs substantially from one location to another

# Fitted points & observed points:
x11()
plot(model.dissimilarity$fitted.values, type='l')
points(respiratorydata.spatial@data$observed, pch = 20, col = 'blue')
```



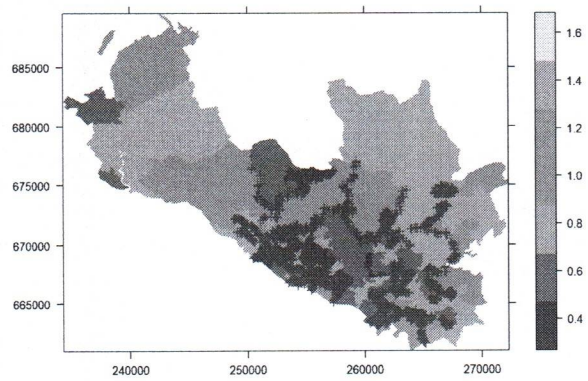

```
# The number and locations of these boundaries are summarised in
# the element of the output list called
# model.dissimilarity$localised.structure$W.posterior, which is
# a K \times K symmetric matrix containing the posterior median
# of each element w_{kj} in the proximity matrix W

# Estimated probability of having a border ----> exp(-z_kj \alpha)
# (NA if they are not adjacent)
model.dissimilarity$localised.structure$W.border.prob[1:9, 1:9]
```

```
##      [,1]    [,2]    [,3]    [,4]  [,5]    [,6] [,7] [,8]  [,9]
##  [1,]   NA 0.00000 0.00000      NA 0.000      NA   NA   NA    NA
##  [2,]    0      NA 0.67525      NA    NA      NA   NA   NA    NA
##  [3,]    0 0.67525      NA      NA 0.000      NA   NA   NA    NA
##  [4,]   NA      NA      NA      NA    NA 0.90875    0   NA    NA
##  [5,]    0      NA 0.00000      NA    NA      NA   NA   NA 0.999
##  [6,]   NA      NA      NA 0.90875    NA      NA   NA   NA    NA
##  [7,]   NA      NA      NA 0.00000    NA      NA   NA   NA    NA
##  [8,]   NA      NA      NA      NA    NA      NA   NA   NA    NA
##  [9,]   NA      NA      NA      NA 0.999      NA   NA   NA    NA
```

```
# If the probability is small, than a border is detected,
# otherwise no
# i.e. if exp(-\alpha z_{ji}) > 0.5, there is NO border
model.dissimilarity$localised.structure$W.posterior[1:9, 1:9]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
##  [1,]   NA    1    1   NA    1   NA   NA   NA   NA
##  [2,]    1   NA    0   NA   NA   NA   NA   NA   NA
##  [3,]    1    0   NA   NA    1   NA   NA   NA   NA
##  [4,]   NA   NA   NA   NA   NA    0    1   NA   NA
##  [5,]    1   NA    1   NA   NA   NA   NA   NA    0
##  [6,]   NA   NA   NA    0   NA   NA   NA   NA   NA
##  [7,]   NA   NA   NA    1   NA   NA   NA   NA   NA
##  [8,]   NA   NA   NA   NA   NA   NA   NA   NA   NA
##  [9,]   NA   NA   NA   NA    0   NA   NA   NA   NA
```

```
x11()
border.locations                  <- model.dissimilarity$localised.structure$W.posterior
respiratorydata.spatial@data$risk <- model.dissimilarity$fitted.values /
  respiratorydata.spatial@data$expected
boundary.final <- highlight.borders(border.locations=border.locations,
                                    spdata=respiratorydata.spatial)
spplot(respiratorydata.spatial, c("risk"),
       sp.layout=list(northarrow, scalebar,text1, text2, boundary.final),
       scales=list(draw = TRUE), at=breakpoints,
       col.regions=terrain.colors(n=length(breakpoints)-1), col="transparent")
```




```
# Map displaying estimated risk and locations of the boundaries for the northern part
# of Greater Glasgow.
graphics.off()
```

BAYESIAN NONPARAMETRIC

02/12/20

1

↳ where the random parameter is ∞-dimensional

$X_1, X_2, X_3, \ldots \mid \tilde{p} \overset{iid}{\sim} \tilde{p}$ = "true" distribution of each observation

$\tilde{p} \sim Q$ = prior distribution

$\tilde{p}$:= random probability measure (= random object)

$\mathcal{P}$:= space of all probabilities on $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$

Things are a little bit more complicated because before the randomness was in the parameters (finite dim. parameters). The randomness now it the whole distribution (infinite dimensional).

$\mathcal{E}_{\mathcal{P}}$:= smallest σ-algebra generated by open sets in the weak-topology

what does it mean? It's the σ-algebra where:

open neighborhood of $P_0 \in \mathcal{P} = \{ P \in \mathcal{P} : \left| \int f_1 dP - \int f_1 dP_0 \right| < \varepsilon,$

$\left| \int f_2 dP - \int f_2 dP_0 \right| < \varepsilon, \ldots,$

$\left| \int f_k dP - \int f_k dP_0 \right| < \varepsilon, \quad \forall \varepsilon > 0$

$\forall f_1, \ldots, f_k : f_j : \mathbb{R} \to \mathbb{R}$ bounded and continuous, $j \in \{1, \ldots, k\}$, $k < \infty \}$

What is a random probability measure?

By random probability measure: $P : (\Omega, \mathcal{F}) \to (\mathcal{P}, \mathcal{E}_{\mathcal{P}})$ = measurable function from Ω to $\mathcal{P}$

$(\tilde{p})$ alternative writing

It means that:

any $P(w)$ is a probability measure on $\mathbb{R}$. How can we assign this random prob. measure which represent our prior belief over the population distribution? We should assign:

$P \sim Q$ in such a way that: Q has full support

$(\tilde{p})$ ↳ prior (↳ it shouldn't be too restrictive since it must allow the posterior to adapt well. If the support is restriced then data may not update freely the prior)

How can we assign a random probability measure $P : (\Omega, \mathcal{F}) \to (\mathcal{P}, \mathcal{E}_{\mathcal{P}})$?

$(\tilde{p})$

How do we assign a random variable?

- X random variable $\sim F_X(x) = \mathbb{P}(X \le x)$ $\forall x$ — we assign its distribution function

- $\{ X(t), t \ge 0 \}$: $\forall k \quad t_1 < t_2 < t_3 < \ldots < t_k$: $Z(X(t_1), X(t_2), \ldots, X(t_k))$

How do we assign a stochastic process??

We assign the finite dimensional distributions of the process in such a way that they've consistent (we assign it $\forall k$: $t_1 < t_2 < \ldots < t_k$)

How do we assign a random prob. measure? (Which is a particular type of stochastic process)

- P random probability measure : $P(A) \quad A \in \mathcal{B}(\mathbb{R})$

we have to assign it $\forall A \in \mathcal{B}(\mathbb{R})$ to def. a random prob. measure

particular case of stochastic process (= family of random variables)

In addition to the consistency, for the random probability measure we need to verify: $\boxed{Z(P(A_1) + P(A_2)) = Z(P(A_1 \vee A_2))}$ $\forall A_1 \cap A_2 = \emptyset$

$P(w)$ is a probability measure.

We will plot distribution functions of $P(w)$.

Finite-dimensional distribution

Dirichlet

= law of a vector $(D_1, D_2, \dots, D_{k-1}, D_k)$ s.t. $D_1 + D_2 + \dots + D_{k-1} + D_k = 1$, $0 \le D_j \le 1 \ \forall j$

$\sim Dir(\alpha_1, \dots, \alpha_k)$ $\quad \alpha_j \ge 0 \ \forall j$

A vector has Dirichlet distribution if: the distribution of this vector $(D_1, \dots, D_k)$ cannot be absolutely continuous w.r.t. the Lebesgue measure in $\mathbb{R}^k$,

but if we consider: $(D_1, \dots, D_{k-1})$, this vector has the density:

$(D_1, \dots, D_{k-1}) \sim$ density : $f(x_1, \dots, x_{k-1}) = \frac{\Gamma(\alpha_1 + \dots + \alpha_k)}{\prod_{j=1}^{k} \Gamma(\alpha_j)} x_1^{\alpha_1 - 1} x_2^{\alpha_2 - 1} \cdots x_{k-1}^{\alpha_{k-1} - 1} \cdot (1 - x_1 - \dots - x_{k-1})^{\alpha_k - 1} \cdot \mathbb{1}_{S_{k-1}}(x)$

$S_{k-1} = \{(x_1, \dots, x_{k-1}) \in \mathbb{R}^{k-1} : 0 \le x_1 + \dots + x_{k-1} \le 1, \ 0 \le x_j \le 1 \ \forall j = 1, \dots, k-1\}$.

if so, the vector $(D_1, \dots, D_{k-1}, D_k)$ is Dirichlet distributed, where $D_k = 1 - D_1 - \dots - D_{k-1}$

Random prob. measure that we call: **DIRICHLET PROCESS** (generalization of the finite dimensional Dirichlet distribution)

We want to define the distribution of a random prob. meas:

$P : (\Omega, \mathcal{F}) \to (\mathcal{P}, \mathcal{C}_{\mathcal{P}})$

$\mathcal{P}$: set of all probabilities on $\mathbb{R}$

Let α be a finite measure on $\mathbb{R}$: $0 < \alpha(\mathbb{R}) < +\infty$. ($\alpha$ not a probability) $\quad := a$

We say that P is a Dirichlet process if for any finite and measurable partition of $\mathbb{R}$, say $A_1, A_2, \dots, A_k$, we have:

$$(P(A_1), P(A_2), \dots, P(A_k)) \sim Dir(\alpha(A_1), \alpha(A_2), \dots, \alpha(A_k))$$

Notations:

$P \sim DP(a, \alpha_0)$

equivalently:

$P \sim \mathcal{D}_\alpha$

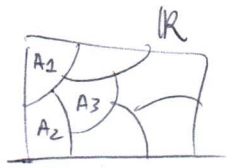
↳ Dirichlet measure with parameter α

$a := \alpha(\mathbb{R})$

$\alpha_0(A) = \frac{\alpha(A)}{\alpha(\mathbb{R})} = \frac{\alpha(A)}{a}$ This is a probability measure

Finite and measurable partition $\{A_1, \dots, A_k\}$ means that:

- disjoint 2 by 2
- and their union is $\mathbb{R}$: $\bigcup_{i=1}^{k} A_i = \mathbb{R}$



Whichever is k, the mass assigned by the random probability measure to each A_j jointly has a finite dimensional Dirichlet distribution:

$$(P(A_1), P(A_2), \dots, P(A_k)) \sim Dir(\alpha(A_1), \dots, \alpha(A_k))$$

$P(A_1) + \dots + P(A_k) = 1$

Properties: $P \sim DP(a, \alpha_0)$ (P Dirichlet Process on $\mathbb{R}$ [i.e. $P(\omega)$ is a probability measure on $\mathbb{R}$] (notice that P has values in $\mathcal{P}$) $\quad (P \sim \mathcal{D}_\alpha, \ \alpha = a \cdot \alpha_0)$

1. $P(A) \sim Beta(\alpha(A), \alpha(A^c)) \quad \forall A \in \mathcal{B}(\mathbb{R})$ [$\forall A \in \mathcal{B}(\mathbb{R})$, $P(A)$ is a random var

proof.

Partition of $\mathbb{R}$: A, A^c.

By def.: $(P(A), P(A^c)) \sim Dir(\alpha(A), \alpha(A^c))$

with $P(A) + P(A^c) = 1$

This is like having $D_1 + \dots + D_k = 1$, $(D_1, \dots, D_k) \sim Dir(\dots)$ $(k = 2)$

And so we know the density of $(D_1, \dots, D_{k-1})$, which in our case is:

$P(A)$ has density: $f(x) \propto \underbrace{x^{\alpha(A)-1}(1-x)^{\alpha(A^c)-1}}_{\text{kernel of a Beta distribution with parameters } \alpha(A),\, \alpha(A^c)} \mathbb{1}_{S_1}(x)$ where $S_1 = (0,1)$

2. $\mathbb{E}[P(A)] = \alpha_0(A) \quad \forall A \in \mathcal{B}(\mathbb{R})$

$\mathrm{Var}(P(A)) = \dfrac{\alpha_0(A)\,\alpha_0(A^c)}{a+1} \quad \forall A \in \mathcal{B}(\mathbb{R})$

proof.

In 1. we proved that $P(A)$ is Beta distributed, so the expectation of a Beta is its first parameter over the sum of the two parameters.

$$\mathbb{E}[P(A)] = \frac{\alpha(A)}{\alpha(A)+\alpha(A^c)} = \frac{\alpha(A)}{\alpha(\mathbb{R})} = \frac{\alpha(A)}{a} = \alpha_0(A)$$

Similarly:

$$\mathrm{Var}(P(A)) = \frac{\alpha(A)\cdot\alpha(A^c)}{(\alpha(A)+\alpha(A^c))^2(\alpha(A)+\alpha(A^c)+1)} = \frac{\alpha(A)\,\alpha(A^c)}{a^2(a+1)}$$

$$= \frac{\alpha(A)}{a}\cdot\frac{\alpha(A^c)}{a}\cdot\frac{1}{a+1} = \frac{\alpha_0(A)\,\alpha_0(A^c)}{a+1}$$

The probability measure α_0 is the mean of P.
Moreover, notice that a influences the variances.

3. $\mathbb{E}[P(A)\cdot P(B)] = \dfrac{\alpha(A\cap B)+\alpha(A)\alpha(B)}{a(a+1)} \quad \forall A,B \in \mathcal{B}(\mathbb{R})$ 03/12/2020

$\mathrm{Cov}(P(A),P(B)) = \dfrac{\alpha_0(A\cap B)-\alpha_0(A)\alpha_0(B)}{a+1}$

proof.

- Assume $A\cap B = \emptyset$.
  Let's consider the partition generated by $\{A,B\} \longrightarrow (A, B, (A\cup B)^c) = (A, B, A^c\cap B^c)$

  Since it's a finite measurable partition:

  $$\underbrace{(P(A), P(B), P(A^c\cap B^c))}_{P(A)+P(B)+P(A^c\cap B^c)=1} \sim \mathrm{Dir}(\alpha(A),\alpha(B),\alpha(A^c\cap B^c))$$

  and so:

  $$(P(A),P(B)) \sim f_{\cdot\cdot}(x_1,x_2) = \frac{\Gamma(a)\,x_1^{\alpha(A)-1}x_2^{\alpha(B)-1}(1-x_1-x_2)^{\alpha(A^c\cap B^c)-1}}{\Gamma(\alpha(A))\cdot\Gamma(\alpha(B))\cdot\Gamma(\alpha(A^c\cap B^c))}\,\mathbb{1}_{S_2}$$

  $$\mathbb{E}[P(A)\cdot P(B)] = \int_{S_2} x_1\cdot x_2\cdot\frac{\Gamma(a)}{\Gamma(\cdot)\Gamma(\cdot)\Gamma(\cdot)}\,x_1^{\alpha(A)-1}x_2^{\alpha(B)-1}(1-x_1-x_2)^{\alpha(A^c\cap B^c)-1}\,dx_1\,dx_2$$

  $$= \frac{\Gamma(\cdot)}{\Gamma(\cdot)\Gamma(\cdot)\Gamma(\cdot)}\int_{S_2}\underbrace{x_1^{\alpha(A)+1-1}x_2^{\alpha(B)+1-1}(1-x_1-x_2)^{\alpha(A^c\cap B^c)-1}}_{\mathrm{Dir}(\alpha(A)+1,\,\alpha(B)+1,\,\alpha(A^c\cap B^c))}\,dx_1\,dx_2$$

  $$= \frac{\Gamma(\cdot)}{\Gamma(\cdot)\Gamma(\cdot)\cancel{\Gamma(\cdot)}}\cdot\frac{\Gamma(\alpha(A)+1)\,\Gamma(\alpha(B)+1)\,\cancel{\Gamma(\alpha(A^c\cap B^c))}}{\Gamma(a+2)}$$

  $$= \frac{\alpha(A)\,\alpha(B)}{a(a+1)}$$

$$\mathrm{Cov}(P(A),P(B)) = \mathbb{E}[P(A)P(B)] - \mathbb{E}[P(A)]\,\mathbb{E}[P(B)]$$
$$= \frac{a\,\alpha_0(A)\,a\,\alpha_0(B)}{a(a+1)} - \alpha_0(A)\alpha_0(B)$$
$$= \alpha_0(A)\alpha_0(B)\left[\frac{a}{a+1} - 1\right] = -\frac{\alpha_0(A)\alpha_0(B)}{a+1}$$

- If $A \cap B \neq \emptyset$.

$A = (A\cap B)\cup(A\setminus B) = (A\cap B)\cup(A\cap B^c)$

$\mathrm{Cov}(P(A),P(B)) = \mathrm{Cov}(P(A\cap B) + P(A\setminus B),\ P(A\cap B) + P(B\setminus A))$
$= \mathrm{Var}(P(A\cap B)) + \mathrm{Cov}(P(A\setminus B), P(A\cap B)) + \dots$

disjoint (we can apply the previous formula and we get to end).

4. Theorem

in case of iid sampling from the Dirichlet Process the Dirichlet measure is conjugate

$$X_1, X_2, \dots, X_n \mid P \overset{iid}{\sim} P$$
$$P \sim \mathcal{D}_\alpha \quad (P \text{ is a Dirichlet Process})$$
$$\alpha \text{ finite measure on } \mathbb{R} \neq 0$$
$$\Longrightarrow \quad P \mid X_1,\dots,X_n \sim \mathcal{D}_{\alpha + \sum_{i=1}^n \delta_{X_i}}$$

Remark: $X_1, X_2, \dots, X_n$ categorical random variables $\in \{1,2,\dots,k\}$

$\Longrightarrow (p_1,p_2,\dots,p_k)$ random object s.t. $p_i = \mathbb{P}(X_1 = 'i')$

We're assuming:

$$X_1, X_2, \dots, X_n \mid (p_1,p_2,\dots,p_n) \overset{iid}{\sim} \text{categorical}(1,\dots,k,p_1,\dots,p_k).$$

Suppose: $(p_1,p_2,\dots,p_k) \sim \mathrm{Dir}(\underbrace{\alpha_1,\alpha_2,\dots,\alpha_k}_{>0})$

$$\Longrightarrow (p_1,p_2,\dots,p_k) \mid X_1,\dots,X_n \sim \mathrm{Dir}(\alpha_1+n_1, \alpha_2+n_2, \dots, \alpha_k+n_k)$$

(n_k: How many of $X_1,\dots,X_n$ were of category 'k')

(Always refering to the theorem)
Let's check what happens a priori and a posteriori. (mean features)

Situation: $X_1,\dots,X_n \mid P \sim P$, $P \sim \mathcal{D}_\alpha$, $a := \alpha(\mathbb{R}) > 0$ $\Longrightarrow P \mid \underline{X} \sim \mathcal{D}_{\alpha + \sum_{i=1}^n \delta_{X_i}}$

A priori: $\mathbb{E}[P(A)] = \alpha_0(A)$

A posteriori:
$$\mathbb{E}[P(A) \mid X_1,\dots,X_n] = \frac{(\alpha + \sum_{i=1}^n \delta_{X_i})(A)}{(\alpha + \sum_{i=1}^n \delta_{X_i})(\mathbb{R})} = \frac{(\cdot)(A)}{\alpha(\mathbb{R}) + \sum_{i=1}^n \delta_{X_i}(\mathbb{R})} = \frac{(\cdot)(A)}{\alpha(\mathbb{R}) + n}$$

($\sum_{i=1}^n \delta_{X_i}(\mathbb{R})$: this counts 1 every time that $X_i \in \mathbb{R}$ (always!))

$$= \frac{(\cdot)(A)}{a+n} = \frac{1}{a+n}\left[\alpha(A) + \sum_{i=1}^n \delta_{X_i}(A)\right]$$
$$= \frac{a}{a+n}\,\alpha_0(A) + \frac{n}{a+n}\,\frac{\sum_{i=1}^n \delta_{X_i}(A)}{n}$$

a posteriori the expectation of P is a convex linear comb. between the prior mean and the empirical distribution.

($\frac{\sum_{i=1}^n \delta_{X_i}(A)}{n}$: empirical distribution associated to the data $\{X_1,\dots,X_n\}$)

If n is large: $\mathbb{E}[P(A) \mid X_1,\dots,X_n] \approx \frac{\sum_{i=1}^n \delta_{X_i}(A)}{n}$

Total mass parameter:
A priori: $a = \alpha(\mathbb{R})$
A posteriori: $(\alpha + \sum_{i=1}^{n} \delta_{X_i})(\mathbb{R}) = a+n$

Note: $\frac{\sum_{i=1}^{n} \delta_{X_i}}{n}$ = distribution that assign mass = $1/n$ to every point (every observation) — empirical distribution of $\{X_1, \dots, X_n\}$

(BTW: this is an example of random prob. measure)

We will build directly trajectories of Dirichlet Process

5. Sethuraman construction of the Dirichlet Process

(we're gonna call it:)

STICK-BREAKING CONSTRUCTION OF DIRICHLET PROCESS

We will show that we'll represent the Dirichlet Process as a discrete random probability measure. The corresponding distribution function, for any ω, are distribution functions of a discrete probability measure for which the support is an infinite sequence of random variables. This is also how we'll represent trajectories of a Dirichlet Process. (with R)

Theorem

Let α be a finite measure on $\mathbb{R}$ not null.
We denote by a the total mass: $a := \alpha(\mathbb{R})$.
We consider two families of random variables: $\{\vartheta_i\}_i$, $\{Y_i\}_i$.
Those two families are independent.
Moreover:

$$\vartheta_1, \vartheta_2, \dots \overset{iid}{\sim} \alpha_0 \qquad \alpha_0 = \frac{\alpha(\cdot)}{\alpha(\mathbb{R})}$$
$$Y_1, Y_2, \dots \overset{iid}{\sim} \text{Beta}(1, a)$$

We define:

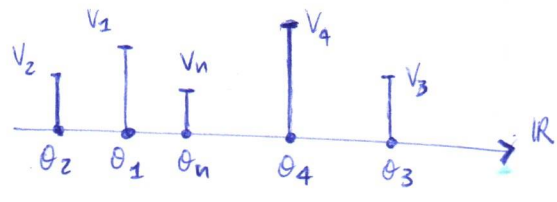
$$V_1 = Y_1$$
$$V_2 = Y_2(1-Y_1)$$
$$V_3 = Y_3(1-Y_1)(1-Y_2)$$
$$\vdots$$
$$V_i = Y_i \prod_{j=1}^{i-1}(1-Y_j) \qquad \forall i$$

This construction of the weights is called STICK-BREAKING *

We define: P (random prob. measure) as: $P = \sum_{i=1}^{+\infty} V_i \delta_{\vartheta_i}$

$P(A) = \sum_{i:\vartheta_i \in A} V_i$, and so:

P assign to every ϑ_i a weight (V_i)

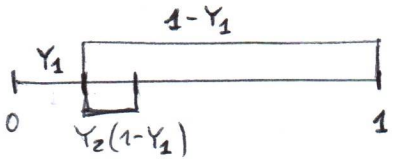


Th. $\Longrightarrow P \sim \mathcal{D}_\alpha$ (= P so defined has Dirichlet distribution with par. α)

so that if we need to simulate a random prob. measure that is a Dirichlet Process we have to simulate an infinite number of iid random variables from α_0, simulate the Y's and then construct the corresponding weights and then consider the corresponding discrete random probability measure.

* Why is it called stick-breaking?

Consider a stick of unitary length:



from Beta(1,a)

from the beginning we drop: Y_1.

$V_1 = Y_1$

Now the stick has a length of $1-Y_1$.
We now have Y_2 (from Beta(1,a)).
The thing we take is:

$V_2 = Y_2(1-Y_1)$

because now the maximum length is not anymore 1 but $(1-Y_1)$

Now what is left has length:

$1 - Y_1 - Y_2(1-Y_1) = (1-Y_1)(1-Y_2)$ [...]

So: $P := \sum_{i=1}^{+\infty} V_i \delta_{\theta_i}$

We want to prove: $\sum_{i=1}^{+\infty} V_i = 1$ a.s. (almost surely)

proof.

We're going to prove: $1 - \sum_{i=1}^{k} V_i \xrightarrow{k \to \infty} 0$.

$1 - \sum_{i=1}^{k} V_i$ = what is left from the stick (length) after chopping it k times

$= 1 - Y_1 - Y_2(1-Y_1) - Y_3(1-Y_1)(1-Y_2) - \dots - Y_k(1-Y_1)\cdots(1-Y_{k-1})$

$= (1-Y_1)\left[1 - Y_2 - Y_3(1-Y_2) - \dots - Y_k(1-Y_2)\cdots(1-Y_{k-1})\right]$

$= (1-Y_1)(1-Y_2)\left[1 - Y_3 - \dots - Y_k(1-Y_3)\cdots(1-Y_{k-1})\right]$

$= [\dots]$

$= \prod_{i=1}^{k}(1-Y_i)$

Homework: get here from $1-\sum_{i=1}^{k} V_i$ by induction

$\mathbb{E}\left[1 - \sum_{i=1}^{k} V_i\right] = \mathbb{E}\left[\prod_{i=1}^{k}(1-Y_i)\right]$

$= \prod_{i=1}^{k} \mathbb{E}[1-Y_i]$ (Y_i are i.i.d.)

$= \prod_{i=1}^{k} \frac{a}{a+1}$ ($Y_i \sim$ Beta(1,a))

$= \left(\frac{a}{a+1}\right)^k \longrightarrow 0$ as $k \to \infty$

Moreover remember that $\left\{1 - \sum_{i=1}^{k} V_i\right\}_{k=1,2,\dots}$ is bounded (from above and below): $0 \le \dots \le 1$.

And so, for bounded random variables convergence in mean implies convergence almost surely.

$\Longrightarrow 1 - \sum_{i=1}^{k} V_i \xrightarrow{k \to \infty} 0$ a.s.

$\Longrightarrow \sum_{i=1}^{k} V_i = 1$ a.s. ■

Remark: $P(\omega) = \sum_{i=1}^{\infty} V_i(\omega)\, \delta_{\theta_i(\omega)}$ probability measure on $\mathbb{R}$ discrete.

The Dirichlet Process' values are discrete probability measures.
However the support of the Dirichlet measure prior is full.

Remark: D_α is a probability measure on $\mathcal{P}$ (= the space of all probability measures on $\mathbb{R}$).

The support of any probability Q on $\mathcal{P}$ is the intersection of all closed sets C such that: $Q(C) = 1$. Or (equivalently) the support of Q can be described as:

$= \{p \in \mathcal{P} : Q(U(p,\varepsilon)) > 0 \quad \forall \varepsilon > 0\}$

$U(p,\varepsilon)$: neighborhood of p of ray ε

We can prove that:

$$supp(\mathcal{D}_\alpha) = \{p : \mathcal{P} : supp(p) \subset supp(\alpha_0)\}.$$

If we fix α_0 such that $supp(\alpha_0) = \mathbb{R} \longrightarrow supp(\mathcal{D}_\alpha) = \{p \in \mathcal{P} : supp(p) \subset \mathbb{R}\} = \mathcal{P}$

all the prob. measures on $\mathbb{R}$ have support contained in $\mathbb{R}$

$\longrightarrow supp(\mathcal{D}_\alpha) = \mathcal{P}$

all probability measures in $\mathbb{R}$

$\Longrightarrow$ We're dealing with a random prob. measure for which the corresponding distribution $\mathcal{D}_\alpha$ selects only trajectories which are discrete with an ∞ number of support points, i.e. with full support.

Convergence Theorem

Let $\{\alpha_m\}_m$ be a sequence of finite measures on $\mathbb{R}$.

$\alpha_m = a_m \cdot (\alpha_0)_m$, where $a_m = \alpha_m(\mathbb{R})$, $\forall m$.

We assume α_{m_0} converging (weakly) to α_0: $\alpha_{m_0} \longrightarrow \alpha_0$ as $m \to \infty$,

(α_0: probability measure on $\mathbb{R}$)

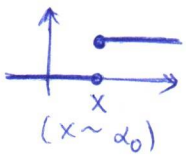
which means:

$$F_{\alpha_{0_m}}(x) \longrightarrow F_{\alpha_0}(x) \quad \forall x \text{ continuity point of } F_{\alpha_0}$$

We define $\forall m$: $p_m \sim \mathcal{D}_{\alpha_m}$.

Then:

1. If $a_m \xrightarrow{m\to\infty} 0 \Longrightarrow p_m \xrightarrow{(d)} \delta_X$, $X \sim \alpha_0$ (in distribution)

(δ_X: 

)

(= approximately the trajectories of p_m will be very similar to the trajectories of a distribution function with only 1 support point in X (but the support point is random))

2. If $a_m \xrightarrow{m\to\infty} \infty \Longrightarrow p_m \xrightarrow{d} \alpha_0$ (constant (there is no randomness))

3. If $a_m \xrightarrow{m\to\infty} a$, $0 < a < \infty \longrightarrow p_m \xrightarrow{d} p$, $p \sim \mathcal{D}_{a \cdot \alpha_0}$

(First part of the R script)

How to simulate the Dirichlet Process P? (A draw)

$P \sim \mathcal{D}_\alpha$

$P = \sum_{i=1}^{\infty} V_i \delta_{\vartheta_i}$

Let's define: $\boxed{P_M = \sum_{i=1}^{M} \tilde{V}_i \delta_{\vartheta_i}}$, $\tilde{V}_i = \frac{V_i}{\sum_{i=1}^{M} V_i} \longrightarrow \sum_{i=1}^{M} \tilde{V}_i = 1$

instead of plotting the true P we'll be able to plot in R an approximation of P

We simulate $\vartheta_1, \dots, \vartheta_M \overset{iid}{\sim} \alpha_0 = N(0,1)$.

We simulate $Y_1, \dots, Y_M \overset{iid}{\sim}$ Beta $(1, a)$ for different values of a.

We compute $V_1 = Y_1$, $V_2 = Y_2(1-Y_1)$, $\dots$, $V_M = Y_M(1-Y_1)\cdots(1-Y_{M-1})$.

If needed, we're going to consider:

$$\tilde{V}_1 = \frac{V_1}{\sum_{i=1}^{M} V_i}, \dots, \tilde{V}_M = \frac{V_M}{\sum_{i=1}^{M} V_i}$$

(Second part of the R script)

4/12/2020

JOINT MARGINAL DISTRIBUTION OF A SAMPLE FROM A DP

$X_1, \dots, X_n \mid P \overset{iid}{\sim} P$

$P \sim \mathcal{D}_\alpha$

We want to describe $\mathcal{L}(X_1, \dots, X_n)$. — The marginal law of $X_1, \dots, X_n$ integrated out w.r.t. P

If we want to simulate $\mathcal{L}(X_1, \dots, X_n)$:

- fix ω. Simulate $P(\omega)$ (through $\sum_{i=1}^{M} \tilde{V}_i \delta_{\theta_i}$)
- Simulate $X_1(\omega), \dots, X_n(\omega) \overset{iid}{\sim} P(\omega)$ (*)
- Change ω and repeat (to simulate another trajectory)

In the end we'll have (e.g.) 1.000 times MC sample from $\mathcal{L}(X_1, \dots, X_n)$.

(*) How to sample $X_1, \dots, X_n \overset{iid}{\sim} \sum_{i=1}^{M} \tilde{V}_i(\omega) \delta_{\theta_i(\omega)}$?

We sample $\{1, 2, \dots, M\}$ using the corresponding weights $\{\tilde{V}_1, \tilde{V}_2, \dots, \tilde{V}_M\}$

(for any X_i we sample from $\{1, 2, \dots, M\}$ with the probabilities : $\{\tilde{V}_1, \tilde{V}_2, \dots, \tilde{V}_M\}$)

Then we assume X_i to be equal to the produced θ_i.

e.g. X_1 : $i = 1 \Rightarrow$ we simulate from $\{1, 2, \dots, M\}$ with prob. $\{\tilde{V}_1, \tilde{V}_2, \dots, \tilde{V}_M\}$
obtaining, for instance, : 2
$\Rightarrow X_1 = \theta_2$

X_2 : $i = 2 \Rightarrow$ we sample again and we got M :
$\Rightarrow X_2 = \theta_M$

Coming back to: how to describe the law of the marginal $(X_1, \dots, X_n)$? — integrated out w.r.t. P

We need to describe $\mathcal{L}(X_1, \dots, X_n)$:

$$\mathcal{L}(X_1, \dots, X_n) = \mathcal{L}(X_1) \cdot \mathcal{L}(X_2 \mid X_1) \cdot \mathcal{L}(X_3 \mid X_2, X_1) \cdot \dots \cdot \mathcal{L}(X_n \mid X_{n-1}, \dots, X_1)$$

$\mathcal{L}(X_1) = \mathcal{L}(X_i) = ?$

$$\mathbb{P}(X_1 \in A) = \underbrace{\int_P \mathbb{P}(X_1 \in A \mid P)\, \mathcal{D}_\alpha(dP)}_{\text{Dirichlet Process}}$$

$$= \int \mathcal{L}(X_1, dP) = \int \mathcal{L}(X_1 \mid P)\, \mathcal{L}(dP)$$

we know what it $X_1 \mid P \Rightarrow X_1 \mid P \sim P$

$\longrightarrow \mathbb{P}(X_1 \in A \mid P) = P(A)$

$$= \int_P P(A)\, \mathcal{D}_\alpha(dP) = \mathbb{E}[P(A)] = \alpha_0(A)$$

$\Longrightarrow \mathcal{L}(X_i) = \alpha_0$

$\mathcal{L}(X_2 \mid X_1)$?

$$\mathcal{L}(X_2 \mid X_1) = \int_P \mathcal{L}(X_2, dP \mid X_1) = \int_P \mathcal{L}(X_2 \mid P, \cancel{X_1})\, \mathcal{L}(dP \mid X_1)$$

$\perp\!\!\!\perp X_1$ since conditioned to P, X_2 depend only on P ($X_2 \mid P \sim P$ *)

$$= \int_P \underbrace{P}_{*}\, \underbrace{\mathcal{L}(dP \mid X_1)}_{\text{posterior distr. of } P \text{ given that we have only one datum}} = \underbrace{\int_P P \cdot \mathcal{D}_{\alpha + \delta_{X_1}}(dP)}_{\text{posterior expectation of } P \text{ (after one single datum)}}$$

$$= \mathbb{E}[P \mid X_1] = \frac{\alpha + \delta_{X_1}}{\alpha + 1}$$

Dirichlet measure where the parameter is $\alpha + \delta_{X_1}$ (over the total mass + 1)

$$\Rightarrow \mathcal{L}(X_2|X_1) = \frac{a\cdot\alpha_0 + \delta_{X_1}}{a+1} = \frac{a}{a+1}\alpha_0 + \frac{1}{a+1}\delta_{X_1}$$

Written differently:

$$X_2|X_1 = \begin{cases} \sim \alpha_0 & \text{with prob. } \frac{a}{a+1} \\ = X_1 & \text{with prob. } \frac{1}{a+1} \end{cases}$$

"$\sim \alpha_0$" = sample from α_0
"$= X_1$" = equal to X_1

So we can see that:

$\mathbb{P}(X_2 = X_1) > 0$, in particular, if α_0 has no atoms ($\alpha_0\{x\} = 0 \quad \forall x$) then $\mathbb{P}(X_2 = X_1) = \frac{a}{a+1} > 0$

$\mathcal{L}(X_i|X_{i-1}, \dots, X_1)$? $\quad i = 3, 4, \dots, n$

$$\begin{aligned}
\mathcal{L}(X_i|X_{i-1}, \dots, X_1) &= \int_P \mathcal{L}(X_i, dP|X_{i-1}, \dots, X_1) \\
&= \int_P \mathcal{L}(X_i|P, \cancel{X_{i-1}, \dots, X_1}^{\,\perp}) \, \mathcal{L}(dP|X_{i-1}, \dots, X_1) \\
&= \int_P \underbrace{\mathcal{L}(X_i|P)}_{P} \underbrace{\mathcal{L}(dP|X_{i-1}, \dots, X_1)}_{\mathcal{D}_{\alpha + \sum_{j=1}^{i-1}\delta_{X_j}}} \\
&= \int_P P \cdot \mathcal{D}_{\alpha+\sum_{j=1}^{i-1}\delta_{X_j}}(dP) \\
&= \mathbb{E}\left[P \,|\, X_1, \dots, X_{i-1}\right] = \frac{\alpha + \sum_{j=1}^{i-1}\delta_{X_j}}{a+i-1} \\
&= \frac{a}{a+i-1}\alpha_0 + \frac{1}{a+i-1}\sum_{j=1}^{i-1}\delta_{X_j}
\end{aligned}$$

Written differently:

$$X_i|X_{i-1}, \dots, X_1 = \begin{cases} \sim \alpha_0 & \text{with prob. } \frac{a}{a+i-1} \\ = X_1 & \text{with prob. } 1/(a+i-1) \\ = X_2 & \text{with prob } 1/(a+i-1) \\ \vdots & \\ = X_{i-1} & \text{with prob. } 1/(a+i-1) \end{cases}$$

$$\Rightarrow \boxed{\mathcal{L}(X_1, X_2, \dots, X_n) = \alpha_0 \cdot \prod_{i=2}^{n}\left(\frac{\alpha + \sum_{j=1}^{i-1}\delta_{X_j}}{a+i-1}\right)}$$

GENERALIZED **POLYA URN** (or CHINESE RESTAURANT PROCESS)

We have seen that if $(X_1, X_2, \dots, X_n)$ is a sample from a DP then we may have ties in the sample. Actually this is convenient because it allows to make clusters in the sample:

$i \sim j$ (i and j are in the same cluster) iff $X_i = X_j$

$\Rightarrow$ the set of labels $\{1, 2, \dots, n\}$ can be partitioned according to this rule.

e.g.

$\{1, n\}, \{3, 4\}, \dots, \{n-1, 2, 5\}$
$X_1 = X_n \quad X_3 = X_4 \quad X_{n-1} = X_2 = X_5$

If we sample another sample from the joint distribution we get other values and to different clusters
$\rightarrow$ these are **RANDOM PARTITIONS**

we can describe the distribution of this random object

$\rho_n = \{A_1, A_2, \dots, A_k\}$ = partition of $\{1, 2, \dots, n\}$

ρ_n is a random object :

$$\mathbb{P}(\rho_n = \{A_1, A_2, \dots, A_k\}) = p(n_1, n_2, \dots, n_k)$$

(function only of the sizes $n_1, n_2, \dots, n_k$, where :
$n_j = \# A_j$
$n_1 + n_2 + \dots + n_k = n$
$0 \le n_j \le n$)

(k) is random as well

$$= \frac{\Gamma(a)}{\Gamma(a+n)} a^k \prod_{j=1}^{k} \Gamma(n_j)$$

$k = k_n$
= number of unique values in the sample $(X_1, X_2, \dots, X_n)$
(K_n = random variable, k value)

$$\mathbb{P}(K_n = k \mid a, n) = |S_1(n,k)| \, a^k \frac{\Gamma(a)}{\Gamma(a+n)} \qquad k = 1, 2, \dots, n$$

$|S_1(n,k)|$: particular function from combinatorial

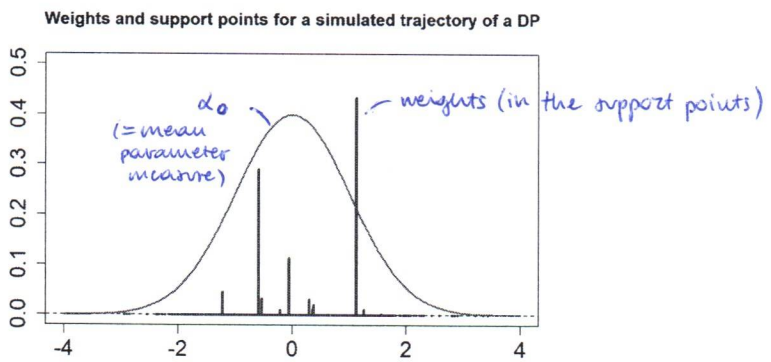
Prior mean:

$$\mathbb{E}[K_n] = a \sum_{i=0}^{n-1} \frac{1}{a+i} \overset{n \text{ large}}{\approx} a \log\left(1 + \frac{n}{a}\right) \approx \log(n)$$

```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### SETHURAMAN construction (Sethuraman, 1994)
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
# Simulation of trajectories of a Dirichlet process with parameter (a, alpha_0);
# truncation of the infinite series and renormalization of the weightsin Sethuraman's
# representation.  The mean trajectory alpha_0() is the standard Gaussian

a   <- 1               # TOTAL MASS parameter
M   <- 1000            # truncation level   M=1000 o M=500
Y   <- vector(length=M) # Y is the vector of the beta proportions in the stick-breaking
tau <- vector(length=M) # support points of the DP (theta_i at the blackboard)
V   <- vector(length=M) # V is the vector of the weights of the support points tau_i

### SIMULATION
set.seed(13)          # fisso il seme aleatorio
Y     <- rbeta(M,1,a) # simulated values of the M rvs which are beta(1,a)-distributed
tau   <- rnorm(M,0,1) # simulated tau_i, iid from alpha_0= N(0,1)
cprod <- cumprod(1-Y)
cprod <- c(1,cprod[1:M-1])
V     <- Y*cprod    ## V_i = Y_i \prod_{j=1}^{i-1}(1-Y_j)
                    ## "spezzo i bastoncini" utilizzando le M v.a. Y_i prima simulate
print(sum(V))  ## Compute the sum of all weights so far
```

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

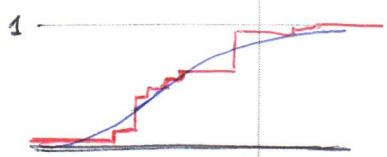
```
V <- V/sum(V)  ## Rinormalization of the weights (needed because truncation of the infinite series)

##### VISUALIZATION of support points (tau_i) and weights (V_I), and alpha_0
# postscript("weights_a1.eps")
x11()
curve(dnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,ylim=c(0,0.5),xlab=" ",ylab="",cex.axis=1.5) # mean parameter alpha_0 (i
n magenta)
abline(h=0,lty=2)
lines(tau,V,"h",lwd=3,col="red")
title("Weights and support points for a simulated trajectory of a DP")
```




```
sort(V) # When a is SMALL, there is one SINGLE LARGE weight (about 1);
```

```
       # all the other weights are almost 0, so that we do NOT see them in the plot.
       # When a is LARGE, all the weights V_i are small, and their values are very similar

##### VISUALIZATION of the trajectory of the corresponding distribution function
#####  it is easier to plot a df than a probability measure!
X11()
oth <- order(tau)

#postscript("trajectories_a1.eps")
curve(pnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,xlab=" ",ylab="",cex.axis=1.5)
# in magenta la f.r. di alpha_0, la misura media
lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col="red",lwd=3)
title("Simulated trajectory of the DP distribution function")

##### TWO more trajectories
set.seed(99)
Y     <- rbeta(M,1,a)
tau   <- rnorm(M,0,1)
cprod <- cumprod(1-Y)
cprod <- c(1,cprod[1:M-1])
V     <- Y*cprod
print(sum(V))
```

Simulated trajectory of the DP distribution function



The red one is one simulated value from the Dirichlet Process P. Let's plot other simulated values.

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

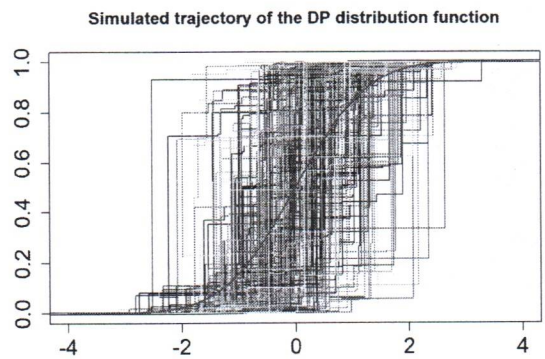
```
V   <- V/sum(V)
oth <- order(tau)
lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col="green",lwd=3)

set.seed(76)
Y     <- rbeta(M,1,a)
tau   <- rnorm(M,0,1)
cprod <- cumprod(1-Y)
cprod <- c(1,cprod[1:M-1])
V     <- Y*cprod
print(sum(V))
```

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

```
V   <- V/sum(V)
oth <- order(tau)
lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col="blue",lwd=3)

### SAMPLING MANY trajectories of the DP df: we add them to the same plot
gra = gray(1:100/100)
gra = rep(gra,10)

for(j in 1:200){
  set.seed(j)
  Y     <- rbeta(M,1,a)
  tau   <- rnorm(M,0,1)
  cprod <- cumprod(1-Y)
  cprod <- c(1,cprod[1:M-1])
  V     <- Y*cprod
  print(sum(V))
  V   <- V/sum(V)
  oth <- order(tau)
  lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col=gra[j])
}
```

```
## [1] 1
## [1] 1
## ..
```

```
curve(pnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,add=T)
```



Obviously the support points are different from simulation to simulation, as well as the step increases.

```
x11()

### What if we change the value of a?? a=0.1, a=100
### Pay attention to sum(V)!

######### Comparison of simulated trajectories among 3 different values of a
a     <- 0.5
M     <- 750              # livello di troncamento   M=500
Y     <- vector(length=M) # Il vettore V conterrà le v.a. beta della costruzione stick breaking
tau   <- vector(length=M) # punti di supporto della mpa processo di Dirichlet - PRIMA li abbiamo chiamati theta
V     <- vector(length=M)
# Simulo
Y     <- rbeta(M,1,a) #simulo le beta
tau   <- rnorm(M,0,1) # simulo i tau in modo iid da alpha_0= N(0,1)
cprod <- cumprod(1-Y)
cprod <- c(1,cprod[1:M-1])
V     <- Y*cprod ## "spezzo i bastoncini" utilizzando le Y prima sumulate
print(sum(V))    ## Valuto la somma dei V
```

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

```
V     <- V/sum(V)    ## Rinormalizzo i V

# postscript("comparison.eps")
par(mfrow=c(1,3))
oth <- order(tau)
curve(pnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,xlab="",ylab="",cex.axis=1.5)
lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col="red",lwd=3)
title("Total mass a=0.5")
gra = gray(1:100/100)
gra = rep(gra,10)

for(j in 1:50){   #for(j in 1:100){
  set.seed(j)
  Y     <- rbeta(M,1,a)
  tau   <- rnorm(M,0,1)
  cprod <- cumprod(1-Y)
  cprod <- c(1,cprod[1:M-1])
  V     <- Y*cprod
  print(sum(V))
  V     <- V/sum(V)
  oth   <- order(tau)
  lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col=gra[j])
}
```

```
## [1] 1
## [1] 1
## ..
```

```
curve(pnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,add=T)

##
a     <- 5
M     <- 750              # truncation level   M=500
Y     <- vector(length=M) # beta proportions in the stick breaking construction
tau   <- vector(length=M) # support points
V     <- vector(length=M)
# Simulation
Y     <- rbeta(M,1,a) #simulation of the beta rvs
tau   <- rnorm(M,0,1) # tau_i iid from alpha_0= N(0,1)
cprod <- cumprod(1-Y)
cprod <- c(1,cprod[1:M-1])
V     <- Y*cprod ## "spezzo i bastoncini" utilizzando le Y prima sumulate
print(sum(V))    ##summation of the weights V
```

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

```
V     <- V/sum(V)  ## Rinormalization of the weights

oth <- order(tau)
curve(pnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,xlab="",ylab="",cex.axis=1.5)
lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col="red",lwd=3)
title("Total mass a=5")
gra = gray(1:100/100)
gra = rep(gra,10)

for(j in 1:50){   #for(j in 1:100){
  set.seed(j)
  Y     <- rbeta(M,1,a)
  tau   <- rnorm(M,0,1)
  cprod <- cumprod(1-Y)
  cprod <- c(1,cprod[1:M-1])
  V     <- Y*cprod
  print(sum(V))
  V     <- V/sum(V)
  oth   <- order(tau)
  lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col=gra[j])
}
```

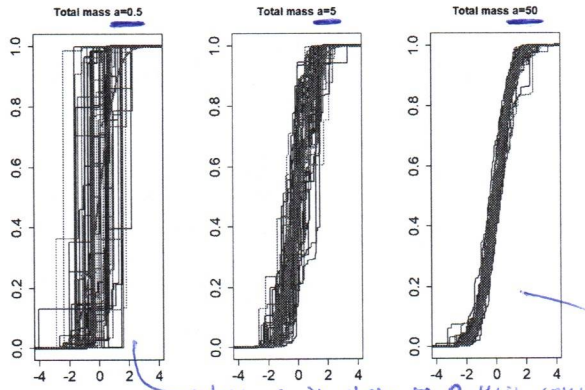
```
curve(pnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,add=T)

##
a     <- 50
M     <- 750 # Livello di troncamento   M=500
Y     <- vector(length=M) # Il vettore V conterrà le v.a. beta della costruzione stick breaking
tau   <- vector(length=M) # punti di supporto della mpa processo di Dirichlet - PRIMA li abbiamo chiamati theta
V     <- vector(length=M)
# Simulo
Y     <- rbeta(M,1,a) #simulo le beta
tau   <- rnorm(M,0,1) # simulo i tau in modo iid da alpha_0= N(0,1)
cprod <- cumprod(1-Y)
cprod <- c(1,cprod[1:M-1])
V     <- Y*cprod ## "spezzo i bastoncini" utilizzando le Y prima sumulate
print(sum(V))  ##Valuto la somma dei V
```

```
## [1] 0.9999997
```

```
V     <- V/sum(V)  ## Rinormalizzo i V

oth <- order(tau)
curve(pnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,xlab="",ylab="",cex.axis=1.5)
lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col="red",lwd=3)
title("Total mass a=50")
gra = gray(1:100/100)
gra = rep(gra,10)

for(j in 1:50){    #for(j in 1:100){
  set.seed(j)
  Y     <- rbeta(M,1,a)
  tau   <- rnorm(M,0,1)
  cprod <- cumprod(1-Y)
  cprod <- c(1,cprod[1:M-1])
  V     <- Y*cprod
  print(sum(V))
  V     <- V/sum(V)
  oth   <- order(tau)
  lines(c(min(tau)-100,tau[oth],max(tau)+100),c(0,cumsum(V[oth]),1),type="s",col=gra[j])
}
```

```
## [1] 0.9999997
## [1] 0.9999994
## [1] 0.9999998
## [1] ..
```

```
curve(pnorm(x,0,1),from=-4,to=4,col="magenta",lwd=2,add=T,xlab="",ylab="",cex.axis=1.5)
```



a is related to the variance. Even if we simulate always the same number of points, with small a we will have a smaller support (in terms of quantity of points) and wider steps. Instead, with large a we will have a lot of different points, and so small steps.

with a very large we can see a good approximation of d_0 (as we saw in the theory)

when a is close to 0 this converges to a degenerate distribution centered in one point

```
### DIRECT sampling from a Dirichlet process
a     <- 1
M     <- 100 # Livello di troncamento   M=30
Y     <- vector(length=M) ##Il vettore V conterrà le v.a. beta della costruzione stick breaking
tau   <- vector(length=M) ## i punti di supporto della mpa processo di Dirichlet - PRIMA li abbiamo chiamati theta
V     <- vector(length=M)
# Simulo
Y     <- rbeta(M,1,a) #simulo le beta
tau   <- rnorm(M,0,1) # simulo i tau in modo iid da alpha_0= N(0,1)
cprod <- cumprod(1-Y)
cprod <- c(1,cprod[1:M-1])
V     <- Y*cprod ## "spezzo i bastoncini" utilizzando le Y prima sumulate
print(sum(V))  ##Valuto la somma dei V
```

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

```
V     <- V/sum(V)

round(V,4)
```

```
##   [1] 0.3502 0.3097 0.1826 0.1263 0.0159 0.0016 0.0131 0.0001 0.0004 0.0000
##  [11] 0.0000 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
##  [21] ..
```

```
# Once that I have simulated the trajectory P(omega), I simulate
# an iid sample of size n frpm P(omega)

n     <- 10 ## sample size
theta <- vector(length=n)
index <- vector(length=n)
for(j in 1:n){
  index[j] <- sample(1:M,size=1,prob=V)
  theta[j] <- tau[index[j]]
}
index
```

```
##  [1] 4 4 1 1 1 4 4 2 1 4
```

```
theta
```

```
##  [1] -1.1701693 -1.1701693 -1.1906103 -1.1906103 -1.1906103 -1.1701693
##  [7] -1.1701693 -0.2958249 -1.1906103 -1.1701693
```

```
unique(theta)
```

```
## [1] -1.1701693 -1.1906103 -0.2958249
```

```
################################################################################
### We could simulate a sample from a Dirichlet process X_1,...,X_n by
### (i) first simulating a (truncated) stick-breaking trajectory of a Dirichlet process P
### (ii) then simulating X_1,...,X_n .
### However, we could use a DIRECT sampling scheme
### GENERALIZED POLYA URN
################################################################################
a         <- 100  #0.1, 1,  100
n         <- 100              # sample size ##n=2, 10, 100, 697
theta     <- vector(length=n) # simulated values: I rename it theta instead than X
theta[1] <- rnorm(1)          # first simulated value from alpha_0 = N(0,1)

for(j in 2:n){ ## from the second simulated value, we get
  w0 <- a/(j-1+a)           ## a NEW observation from alpha_0 with probability w0
  w1 <- rep(1/(j-1+a),j-1) ## or an OLD observation, each of the old obs with prob w1

  index <- sample(0:(j-1),size=1,prob=c(w0,w1))
  # Sample the index from a discrete distribution with weights
  #   contained in the vector prob, i.e. sample from the categorical distr.
  if(index==0){
    theta[j] <- rnorm(1) # NEW observation
  }
  else{
    theta[j]=theta[index] # OLD observation
  }
}

theta
```

$$X_i | X_{i-1}, \dots, X_1 = \begin{cases} \sim d\sigma & \text{with prob } \dots \\ = X_1 & \\ \vdots & \\ = X_{i-1} & \end{cases}$$

```
##   [1] -0.217232687 -1.915710698  0.004102229  1.666478236 -0.980752344
##   [6] -1.237849900  0.778024737  0.307461609 -2.242566186  0.012960828
##  [11] ..
```

```
unique(theta)
```

```
##  [1] -0.217232687 -1.915710698  0.004102229  1.666478236 -0.980752344
##  [6] -1.237849900  0.778024737  0.307461609 -2.242566186  0.012960828
## [11] ..
```

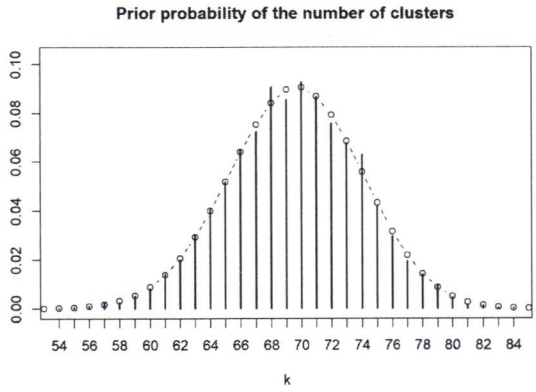
```
# K_n: number of UNIQUE values in the sample theta
k <- length(unique(theta))
k
```

```
## [1] 72
```

```
## k is a random variable with distribution specified by the Ewens formula

# In order to plot this discrete density, we could use a Monte Carlo method

k <- vector(length=5000)
set.seed(19)
for(i in 1:5000){ ##Sampling from the generalized Polya urn repeatedly
  theta[1] <- rnorm(1)
    for(j in 2:n){
        w0    <- a/(j-1+a)
        w1    <- rep(1/(j-1+a),j-1)
        index <- sample(0:(j-1),size=1,prob=c(w0,w1))
        if(index==0){
            theta[j] <- rnorm(1)
        }
        else{
            theta[j]=theta[index]
        }
    }
    k[i] <- length(unique(theta))
}

k
```

```
##    [1] 68 70 64 71 74 69 61 74 69 70 71 65 75 62 72 64 67 69 59 67 76 70 73 60
##   [25] 70 71 79 70 69 61 76 65 67 77 67 66 69 70 72 72 76 65 71 70 64 75 57 68
##   [49] ..
```

```
ymax = max(table(k)/5000)+0.01
x11()
plot(table(k)/5000,ylim=c(0,ymax),ylab="")
title("Prior probability of the number of clusters")
#### NON chiudere la finestra grafica

# Compare thios MC distribution to the "exact" one - Antoniak 1974 (Ewens formula)
library(gsl) #carico la libreria scientifica gsl per la  lngamma
library(untb) #tale libreria contiene la matrice logS1
#LogS1[] e' una matrice che contiene il logaritmo dei numeri di Stirling di primo tipo
# di primo tipo sulla matrice [1:100]*[1:100]
## dato che n=100 la possiamo  utilizzare

pk     <- vector(length=n) ## EXACT (analytic) probability density of K_n  in the log scale
cost.a <- lngamma(a)-lngamma(a+n) ## fattori che non dipendono da l=1...n
for(l in 1:n){
  pk[l] <- logS1[n,l]+l*log(a)+cost.a #Logaritmo di pk
}

pk = exp(pk) #riporto le probabilita' in scala originale
points(1:n,pk,type="b",col="red",lty=2) # disegno i valori delle masse della distribuzione esatta di Kn
```




```
#### EXPECTATION of  K_n
meanKn = a *sum(1/(a+seq(0,n-1))) # computed via its analytic expression
meanKn
```

```
## [1] 69.56534
```

```
sum(seq(1:n)*pk) # Monte Carlo approximation
```

```
## [1] 69.56534
```

```
###  How does this distribution change varying n or/and a?
###  When a is SMALL (e.g. a=0.1), the sample from the DP yields
###    a few unique values, i.e. the distribution of K_n is
###    concentrated on small values, close to 1
###  When a is LARGE (e.g. a=100), the sample from the DP has may small jumps
###     so that we get larger values for K_n

### Per quanto riguarda la partizione aleatoria indotta da un
### campione dal processo di Dirichlet, in generale, si otterrà un
### numero PICCOLO di cluster GRANDI e un numero GRANDE di cluster di PICCOLA numerosità.
```

DIRICHLET PROCESS MIXTURES (DPM)

9/12/20

Suppose we're interest in density estimation of:

$$X_1, X_2, \dots, X_n \mid P \overset{iid}{\sim} P \quad \text{— population distribution}$$
$$P \sim \pi$$

We want an estimate for P.
A density estimate of the population distribution is for instance:

$$\mathbb{E}[P \mid X_1, \dots, X_n] = \int_{\mathcal{P}} P \cdot \mathcal{L}(dP \mid X_1, X_2, \dots, X_n)$$
$$\perp\!\!\!\perp \mathcal{L}(X_{n+1} \mid X_1, \dots, X_n)$$

proof.

$$\mathcal{L}(X_{n+1} \mid X_1, \dots, X_n) = \frac{\mathcal{L}(X_1, \dots, X_n, X_{n+1})}{\mathcal{L}(X_1, \dots, X_n)} = \frac{\int_{\mathcal{P}} \mathbb{P}(X_{n+1} \in \cdot) \prod_{i=1}^n \mathbb{P}(X_i \in \cdot)\, \mathcal{L}(dP)}{\int_{\mathcal{P}} \prod_{i=1}^n \mathbb{P}(X_i \in \cdot)\, \mathcal{L}(dP)}$$

posterior distribution of P given data

$$= \int_{\mathcal{P}} P\, \mathcal{L}(dP \mid X_1, \dots, X_n)$$

■

(we already saw that:)

If $P \sim \mathcal{P}_\alpha \Longrightarrow (X_1, \dots, X_n)$ will have ties.
(Dirichlet Process)

Remark: The Dirichlet Process is <u>not</u> suitable to represent the conditional distribution of continuous data.

We're going to assume:

$$X_i \mid P \overset{iid}{\sim} \underbrace{f(x)(\omega)}_{\text{random density w.r.t. the Lebesgue measure on } \mathbb{R}} = \int_{\Theta} k(x;\theta)\, P(d\theta)(\omega)$$

where $k(x;\theta)$ is a parametric kernel: it's a density as a function of x and it's a measurable function as function of the parameter θ:

$$x \longmapsto k(x;\theta) \quad \text{density on } \mathbb{R} \quad \forall \theta$$
$$\theta \longmapsto k(x;\theta) \quad \text{measurable} \quad \forall x$$

Moreover: $P \sim \mathcal{D}_\alpha$.

$\Longrightarrow$ We're assuming that (X_i's are absolutely continuous w.r.t. the Lebesgue measure on $\mathbb{R}$ ← the density of) Moreover we assume a parametric form for these densities (with which these X_i's are distributed) which are a mixture of a PARAMETRIC KERNEL mixtured w.r.t. the parameter θ, where the mixing measure is a DIRICHLET PROCESS.

For instance:

$$P = \sum_{j=1}^{\infty} p_j \delta_{\tilde{\theta}_j}, \quad \{\tilde{\theta}_j\} \perp\!\!\!\perp \{V_j\} \qquad \theta_j \overset{iid}{\sim} \alpha_0, \quad V_j \overset{iid}{\sim} \text{Beta}(1, a)$$

and $\{p_j\}$ are def. with the stick-breaking construction

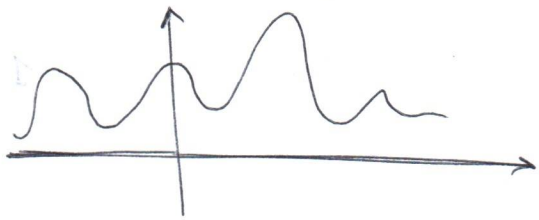
The simplest case is $k(x;\theta)$ density of $N(\,;\theta, 1)$ (mean, variance).

If P has the stick-breaking representation

$$\Longrightarrow f(x)(\omega) = \int_{\Theta} k(x;\theta) \sum_{j=1}^{\infty} p_j \delta_{\tilde{\theta}_j}(d\theta) = \sum_{j=1}^{\infty} p_j\, k(x;\tilde{\theta}_j)$$

for instance: (gaussian) $\sum_{j=1}^{\infty} p_j \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\tilde{\theta}_j)^2}{2}}$ ($\Theta = \mathbb{R}$ = where θ lives)

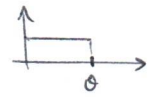
A plot of the example will be:



parametric space = where the parameters live

Move generally: $k(x;\underline{\theta}) \Rightarrow N(\cdot;(\mu,\sigma^2))$, $\Theta = \mathbb{R}\times(0,+\infty)$ (μ,σ^2)

More generally $k(x;\theta)$ can be any location scale density. It can be for instance $= \frac{1}{\theta}\mathbb{1}_{(0,\theta)}(x)$, which is the uniform over $(0,\theta)$:



let's go back to the model:

$$\begin{cases} X_1,\dots,X_n \mid P \overset{iid}{\sim} f(x)(\omega) = \int_\Theta k(x;\theta)\,P(d\theta) & \Theta \subset \mathbb{R}^k \\ P \sim \mathcal{D}_\alpha\,, \quad \alpha = a\cdot\alpha_0 \end{cases}$$

MODEL 1

:= DIRICHLET PROCESS MIXTURE :
we're assuming that data have a density and the density is random.
The parameter now it the density. (∞ dimensional)

We want to compute the Bayesian estimate of the density: we can, for instance, compute the posterior expectation:

$$\mathbb{E}\left[f(x)(\omega)\mid X_1,\dots,X_n\right] = \mathbb{E}\left[\int_\Theta k(x;\theta)P(d\theta)\mid X_1,\dots,X_n\right] = \int_\Theta k(x;\theta)\,\mathbb{E}\left[P\mid X_1,\dots,X_n\right](d\theta)$$

by the way, this coincides with the predictive distribution for a new observation given $X_1,\dots,X_n$:
$= f_{X_{n+1}\mid X_1,\dots,X_n}(x)$

We can prove that MODEL 1 is equivalent to:

$$\begin{cases} \{X_1,\dots,X_n\} \text{ and } P \text{ are independent conditionally to } \theta_1,\dots,\theta_n \\ X_i\mid\theta_i \overset{iid}{\sim} k(\cdot;\theta_i) \quad i=1,\dots,n \\ \theta_1,\dots,\theta_n \mid P \overset{iid}{\sim} P \\ P \sim \mathcal{D}_\alpha \end{cases}$$

latent parameters (latent variables)

MODEL 2

proof: (model 2 implies model 1)

Assuming model 2, let's prove that integrating out the latent parameters we have the distribution for the data that we have in model 1.

$$\mathcal{L}(X_1,\dots,X_n\mid P) = \int_{\Theta^n}\mathcal{L}(X_1,\dots,X_n,\theta_1,\dots,\theta_n\mid P)\,d\theta_1\dots d\theta_n$$

$$= \int_{\Theta^n}\mathcal{L}(X_1,\dots,X_n\mid\theta_1,\dots,\theta_n,\cancel{P})\,\mathcal{L}(d\theta_1,\dots,d\theta_n\mid P)$$

since $\mathcal{L}$ holds: $X_i\mid\theta_i \sim k(\cdot;\theta_i)$ (X_i are cond $\perp\!\!\!\perp$)

$$= \int_{\Theta^n} k(x_1;\theta_1)\cdots k(x_n;\theta_n)\,\mathcal{L}(d\theta_1,\dots,d\theta_n\mid P)$$

$$\mathcal{L}(X_1,\dots,X_n|P) = \int_{\mathbb{H}^n} k(x_1;\theta_1)\cdots k_n(x_n;\theta_n)\,P(d\theta_1)\cdots P(d\theta_n)$$
$$\Updownarrow$$
$$\stackrel{!}{=} \int_{\mathbb{H}^n} k(x_1;\theta_2)\,P(d\theta_1)\cdots k(x_n;\theta_n)\,P(d\theta_n)$$
$$\Longrightarrow X_1,\dots,X_n|P \overset{iid}{\sim} f(\cdot)(\omega) = \int_{\mathbb{H}} k(\cdot;\theta)\,P(d\theta). \quad \blacksquare$$

The most important ingredient is P $(P \sim \mathcal{D}_\alpha)$
The randomness of $f(\cdot)$ comes from the randomness of P.
What is the posterior of P in this case?

$$P|X_1,\dots,X_n \sim \int \mathcal{D}_{\alpha+\sum_{i=1}^n \delta_{\theta_i}}\, H(d\theta_1,\dots,d\theta_n|X_1,\dots,X_n)$$

we have to get rid of the latent parameters

$$H(d\theta_1,\dots,d\theta_n|X_1=x_1,\dots,X_n=x_n) \propto \prod_{i=1}^n k(x_i;\theta_i)\,\alpha_0(d\theta_1)\prod_{i=2}^n a\cdot\alpha_0(d\theta_i)+\sum_{j=1}^{i-1}\delta_{\theta_j}(d\theta_i)$$

this is computationally heavy!
However we can use an MCMC algorithm (Gibbs sampler) which simulates from this posterior distr.

How to compute : $\mathbb{E}[f(x)|X_1,\dots,X_n]$?
It's possible to prove that:

$$\mathbb{E}[f(x)|X_1,\dots,X_n] = \int_{\mathbb{H}^n} f_{X_{n+1}|\theta_1,\dots,\theta_n}(x|\theta_1,\dots,\theta_n)\,H(d\theta_1,\dots,d\theta_n|X_1,\dots,X_n)$$

where:

$$f_{X_{n+1}|\theta_1,\dots,\theta_n}(x|\theta_1,\dots,\theta_n) = \frac{a}{a+n}\,f_0(x) + \frac{a}{a+n}\sum_{j=1}^n k(x;\theta_j)$$

marginal distribution that we obtain integrating out $k(\cdot;\cdot)$ w.r.t. θ using the measure α_0 :

$$f_0(x) = \int_{\mathbb{H}} k(x;\theta)\,\alpha_0(d\theta)$$

$$\Longrightarrow \underbrace{\mathbb{E}[f(x)|X_1,\dots,X_n]}_{\text{density estimate}} \cong \frac{1}{M}\sum_{j=1}^M f_{X_{n+1}|\theta_1,\dots,\theta_n}(x|\theta_1^{(j)},\dots,\theta_n^{(j)})$$

given an MCMC sample from $H(..)$: $(\theta_1^{(j)},\dots,\theta_n^{(j)})$

Clustering via BNP models

Alessandra Guglielmi

December 9, 2020

BNP mixture models for clustering

MODEL

$$X_i|\theta_i \overset{\text{ind.}}{\sim} f(x_i|\theta_i)\ i = 1, \ldots, n$$
$$\theta_1, \ldots, \theta_n|P \overset{\text{i.i.d.}}{\sim} P$$
$$P \sim \Pi(\cdot; \rho, P_0)$$

where P is a discrete r.p.m., for instance $P \sim DP(\alpha, P_0)$, so that the sample $(\theta_1, \theta_2, \ldots, \theta_n)$ from P will have ties

The likelihood is fixed according the type of data we have

The prior: the prior (induced by the model) on ρ, the random partition of the data label set $\{1, 2, \ldots, n\}$, $\rho = \{A_1, A_2, \ldots, A_K\}$, where each A_l contains all the data indexes put in the same group by the relationship

also k is random

$$X_i \sim X_j \Leftrightarrow \theta_i = \theta_j$$

BNP model-based clustering

POSTERIOR of ρ, given data $X_1, X_2, \ldots, X_n$

CLUSTER ESTIMATE: as clustering of data we choose a summary of the posterior distribuzion of ρ; for instance the the value of the random partition minimizing a posteriori the loss function L:

$$\hat{\rho} \in \arg\min_y \mathbb{E}[L(\rho, y)|data]$$

where $L(\rho, y)$ represents the cost of estimating the "true" ρ by y

Binder's loss function L: quella che assegna un costo b quando due elementi sono erroneamente classificati nello stesso cluster e un costo a quando due elementi sono erroneamente assegnati a cluster distinti.

$$K = \frac{b}{(a+b)} \in [0, 1]$$

Model-based clustering: posterior estimate of $\boldsymbol{c}$

$\boldsymbol{c} = (c_1, \ldots, c_n)$= vector of allocation variables
c_i denotes the cluster the i observation is associated,
$c_i \in \{1, \ldots, k\}$ when $K = k$

Ex: $n = 3$, and $\theta_1 = \theta_2 \neq \theta_3$, then $\boldsymbol{c}$ may be either $(\mathbf{1}, \mathbf{1}, \mathbf{2})$ or $(2, 2, 1)$

posterior of the random partition identified by

$$P(c_i = k|data) \simeq \frac{1}{M} \sum_{m=1}^{M} \mathbf{1}_{c_i^{(m)}=k}\ k = 1, \ldots, K,\ i = 1, \ldots, n$$

$$\pi_{ij} = P(c_i = c_j|data) \simeq \frac{1}{M} \sum_{m=1}^{M} \mathbf{1}_{\{c_i^{(m)}=c_j^{(m)}\}},\ i, j = 1, \ldots, n,\ i < j$$

Model-based clustering: posterior estimate of ρ

Dahl (2006): minimize (over the simulated draws) the sum of the squares

$$\sum_{i<j} \left(\mathbf{1}_{c_i^{(m)}=c_j^{(m)}} - \pi_{ij} \right)^2$$

Equivalent to min Binder's loss a posteriori: see

- Dahl, D. B. (2006). Model-based clustering for expression data via a Dirichlet process mixture model. In "Bayesian inference for gene expression and proteomics", 201-218.
- Fritsch, Ickstadt (2009). Improved criteria for clustering based on the posterior similarity matrix. *Bayesian analysis*, 4, 367-391.

R packages: `mcclust`, **sdols**

BNP mixture models for clustering

Probabilità che ogni dato sia stato assegnato al cluster esatto

$$\mathbb{P}(Y_{new}(X_i) = 1 | X_{new} = X_i, data),$$

con $Y_{new}(X_i) = 1$ se X_{new} è nello stesso cluster di X_i (or 0)

È la probabilità che una nuova osservazione sia assegnata allo stesso cluster di X_i, condizionatamente a $X_{new} = X_i$; questo numero ha senso solo se confrontato a tutti gli altri valori $j = 1, \ldots, n, j \neq i$

Valori grandi indicano che X_i è "nested" nel suo cluster
valori piccoli indicano che X_i è un "frontier" point nel cluster cui è stato assegnato
Tali probabilità hanno un significato di misclassification tool

Argiento, Cremaschi, Guglielmi (2014). A density-based algorithm for cluster analysis using species sampling Gaussian mixture models. *Journal of Computational and Graphical Statistics*, 23, 1126-1142.

GLMM: random effects are a sample from DP

Assume a GLMM model, and assume that the group specific random-effects parameters are a sample from a DP

Ex: LMM; for $i = 1, \ldots, n_j$, $j = 1, \ldots, J$:

$$\begin{aligned} Y_{ij} &= \boldsymbol{x}_{ij}^T \beta_j + \epsilon_{ij}, \quad \epsilon_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2) \\ \beta_j | \boldsymbol{\theta}, \Sigma &\overset{\text{i.i.d.}}{\sim} P \quad j = 1, \ldots, J \\ P &\sim DP(\alpha, P_0) \\ \sigma^2 &\sim \pi(\sigma^2) \end{aligned}$$

There is one parameter for each group, but the prior and the posterior for $(\beta_1, \ldots, \beta_J)$ contains ties; clustering of the groups is achieved through clustering of $(\beta_1, \ldots, \beta_J)$ a posteriori

GLMM: random effects are a sample from DP

JAGS example: regression model for infarction patients who went under angioplasty

Y_i: =1 if the patient was discharged alive, =0 otherwise
$\boldsymbol{x}_i$: covariate vector; age, OB (onset-balloon) time in the log scale, infarction severity (killip=1 serious), hospital the patient was admitted to and went under surgery

GOAL: hospitals clustering based on covariates and survival

Guglielmi, Ieva, Paganoni, Ruggeri, Soriano, J. (2013). Semiparametric Bayesian modeling for the classification of patients with high observed survival probabilities. *Journal Royal Stat Soc, C.*

GLMM with random effects iid from DP

For $i = 1, \ldots, n$:

$$\begin{aligned} Y_i | p_i &\sim \text{Be}(p_i) \quad logit(p_i) = \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i2} + b_{j[i]} \\ (\beta_1, \beta_2, \beta_3) = \boldsymbol{\beta} &\perp \{b_j, j = 1, \ldots, J\}, \\ \boldsymbol{\beta} &\overset{\text{i.i.d.}}{\sim} \mathcal{N}_3(\mathbf{0}, 100\, I_3), \quad b_1, \ldots, b_J | P \overset{iid}{\sim} P, \; P \sim DP(\alpha, P_0). \end{aligned}$$

Through the marginal prior of $(b_1, \ldots, b_J)$, we induce a prior for ρ, the partition of the hospital labels $\{1, 2, ..., J\}$

ρ is the partition identified by unique values in $(b_1, \ldots, b_J)$, which is a (prior) sample from a Dirichlet process

Multivariate GLMM with random effects iid from the Pitman-Yor process

Guglielmi, Ieva, Paganoni, Quintana (2016). A semiparametric Bayesian joint model for multiple mixed-type outcomes: an application to acute myocardial infarction. *Advances in Data Analysis and Classification*, 1-25.

We have an example of grouped data. We have patients in different hospitals and the idea it to cluster the different hospitals. Are there group of hospitals which behave similarly?

j = hospital, i = patient

Y_{ij} = answer of the i-th patient of the j-th hosp.

$\vec{x}_{ij}$ = characteristics of the i-th patient of the j-th hosp.

Actually we'll consider:

ties in β's will give the clustering structure of the groups

(The clustering of all the different groups (of all the different hospitals) will be achieved through the clustering a posteriori of the params $(\beta_1, \ldots, \beta_J)$)

Model that we assume

$= \begin{cases} 1 & \text{if patient } i \text{ is admitted to } j(i) \\ 0 & \text{otherwise} \end{cases}$

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Modelli lineari generalizzati con prior nonparametrica sugli effetti casuali
### GLMM con JAGS   su 240 pz e 17 ospedali
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
# DATI CENTRATI
# Importo i dati
input = read.table("dataset.txt",header=T)
names(input)
```

```
## [1] "vivo"   "eta"    "logOB"  "killip" "centro"
```

```
summary(input)
```

```
##       vivo           eta               logOB             killip
##  Min.   :0.00   Min.   :-31.06667   Min.   :-1.4655   Min.   :0.0000
##  1st Qu.:1.00   1st Qu.: -8.06667   1st Qu.:-0.7034   1st Qu.:0.0000
##  Median :1.00   Median : -0.06667   Median :-0.1727   Median :0.0000
##  Mean   :0.95   Mean   :  0.00000   Mean   : 0.0000   Mean   :0.0625
##  3rd Qu.:1.00   3rd Qu.:  8.93333   3rd Qu.: 0.6013   3rd Qu.:0.0000
##  Max.   :1.00   Max.   : 28.93333   Max.   : 3.6908   Max.   :1.0000
##      centro
##  Min.   : 1.000
##  1st Qu.: 5.000
##  Median : 9.000
##  Mean   : 8.367
##  3rd Qu.:12.000
##  Max.   :17.000
```

```
n.data = dim(input)[1] # numero totale pazienti
J      = 17            # numero ospedali
M      = 40            # troncamento serie Sethuraman
## eta and logOB have been standardized

library(rjags)      # interfaccia R con JAGS
```

```
library(plotrix)    # per fare plot CIs
set.seed(1)         # fisso il seed per poter riprodurre

# genero una lista con i dati da passare a JAGS
data <- list(Y = input$vivo, AGE = input$eta, KILLIP = input$killip, LOGOB = input$logOB,
             CENTRO = input$centro, n = n.data, J = J, M = M)

# definisco una lista con lo stato iniziale della catena
# fornisco anche il seed e il tipo di generatore di numeri casuali
# theta sono i location points della mistura (serie di Sethuraman troncata)
r     = rep(0.5,M)
theta = rep(0,M)
S     = rep(1,J)
inits = list(beta = rep(0,3), a = 1, lambda.bb = 1, r = r, theta = theta, S = S, Snew = 1,
             .RNG.seed = 2, .RNG.name = 'base::Wichmann-Hill')
```

```
#########################
### CREATE JAGS MODEL ###
#########################
# Inside: 'jags_model.bug'--------------------------------------------------------------------
# # MODEL
# model{
#   # Precision Parameter
#   # alpha ~ <- 1;
#         a ~ dexp(1) ;
#   alpha <- a + 0.5 ;   # Exp con supporto (0.5, +infinity) (per problemi numerici)
#
#   # Constructive DP
#   pp[1] <- r[1];
#   for (j in 2:M) {
#       pp[j] <- r[j] * (1 - r[j - 1]) * pp[j -1 ] / r[j - 1]; # pesi della serie
#   }
#   p.sum <- sum(pp[]);
#
#   for (j in 1:M){
#       theta[j] ~ dnorm(mu.bb,tau.bb);   # locations della serie
#             r[j] ~ dbeta(1, alpha);
#   # scaling to ensure sum to 1
#       pi[j] <- pp[j] / p.sum ;
#   }
#
#   mu.bb  <- 0 ;                  # media della baseline P_0 #mu.bb ~ dnorm(0, 0.01);
#   tau.bb <- pow(Lambda.bb,-2);   # precisione  della baseline P_0
#   Lambda.bb ~ dunif(0,50);       # dev std
#
#   for(i in 1:J){
#       S[i] ~ dcat(pi[]);     # campiono dalla discreta
#       bb[i] <- theta[S[i]];  # locations associate al peso campionato
#
#       for (j in 1 : M) {
#               SC[i, j] <- equals(j, S[i]);  # SC mi serve per costruire la partizione e il numero di gruppi
#           }
#   }
#
# # verosimiglianza
#   for(i in 1:n){
#       logit(p[i]) <- beta[1]*AGE[i] + beta[2]*LOGOB[i]+ beta[3]*KILLIP[i] + bb[CENTRO[i]];
#       Y[i] ~ dbern(p[i]);
#       }
# # Prior sugli effetti fissi
#   for(i in 1:3){
#       mu[i] <- 0;
#       beta[i] ~ dnorm(mu[i], 0.001);
#   }
#
#   # New random CENTRO
#
#   Snew ~ dcat(pi[]);
#   newcentro <- theta[Snew];
#
#   # total clusters
#   K <- sum(cl[])
#   for (j in 1 : M) {
#       sumSC[j] <- sum(SC[ , j])
#       cl[j]    <- step(sumSC[j] -1)
#   }
# }
# -------------------------------------------------------------------------------------------
```

$b_1, \dots, b_J \mid P_M \overset{iid}{\sim} P_M = \sum_{j=1}^{M} \tilde{v}_j \delta_{\theta_j}$

random intercept that we're adding to $\text{logit}(p_i)$

we need to sample from this:

1. sample $\forall i$: $S_i \sim \text{categ.}\{1, 2, \dots, M\}$ with prob. masses $\{\tilde{v}_1, \dots, \tilde{v}_M\}$
2. PUT $b_i = \theta(S_i)$

```
modelGLMM_DP=jags.model("jags_model.bug", data=data, inits=inits, n.adapt=1000, n.chains=1)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 240
##    Unobserved stochastic nodes: 103
##    Total graph size: 2991
##
## Initializing model
```

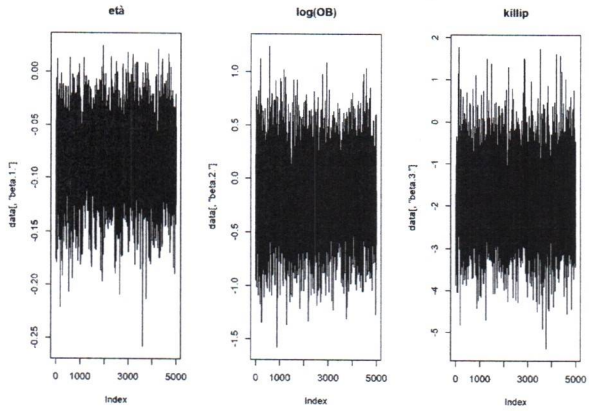
```
# Faccio un update del modello per il numero di iterazioni specificato SENZA salvare nulla
update(modelGLMM_DP,19000)

variable.names = c("bb", "beta",  "tau.bb", "newcentro", "alpha","K")
# Monitoro i parametri:
n.iter = 50000
thin   = 10

outGLMM_DP = coda.samples(model=modelGLMM_DP,variable.names=variable.names,n.iter=n.iter,thin=thin)
# salvo l'intera catena dei parametri monitorati (si tratta di una lista mcmc)

save(outGLMM_DP,file='GLMMDP_output.Rdata')
```

```
##############
### OUTPUT ###
##############

# Importo i dati
input = read.table("dataset.txt",header=T)
names(input)
```

```
## [1] "vivo"   "eta"    "logOB"  "killip" "centro"
```

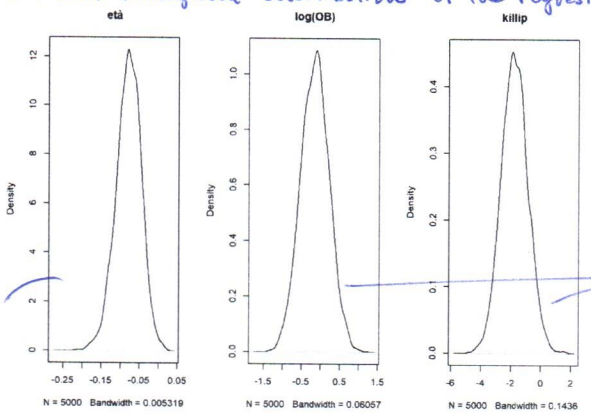
```
n.data = dim(input)[1] # numero totale pazienti
J      = 17            # numero ospedali
M      = 40            # troncamento serie Sethuraman

#install.packages("plotrix")
library(coda)        # pacchetto per analizzare catene
library(plotrix)     # per fare plot CIs

load('GLMMDP_output.Rdata') # carico l'output ottenuto con coda.samples

# summary(output)
data = as.matrix(outGLMM_DP) # trasformo il dataframe in matrice
data = data.frame(data)
attach(data)
n.chain = dim(data)[1]   # lunghezza catena (final sample size)
```

```
##################
### TRACEPLOTS ###
##################
x11()
par(mfrow=c(1,3))
plot(data[,'beta.1.'],main='età',type='l')
plot(data[,'beta.2.'],main='log(OB)',type='l')
plot(data[,'beta.3.'],main='killip',type='l')
```




```
x11()
par(mfrow=c(1,3))
plot(density(data[,'beta.1.']),main='età')
plot(density(data[,'beta.2.']),main='log(OB)')
plot(density(data[,'beta.3.']),main='killip')
```

Posterior marginal distributions of the regression parameters:



significative ($\neq 0$)

a bit of uncertainty for these variables since the support has the 0

```
# età e killip sono significative; un po' meno significativo è log(OB)
mean(data[,'beta.1.']<0)
```

```
## [1] 0.9942
```

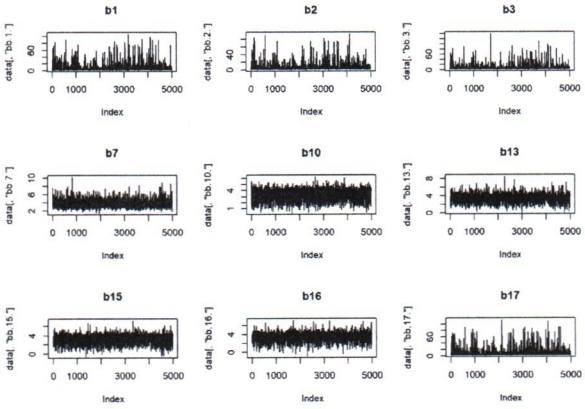
```
mean(data[,'beta.3.']<0)
```

```
## [1] 0.9744
```

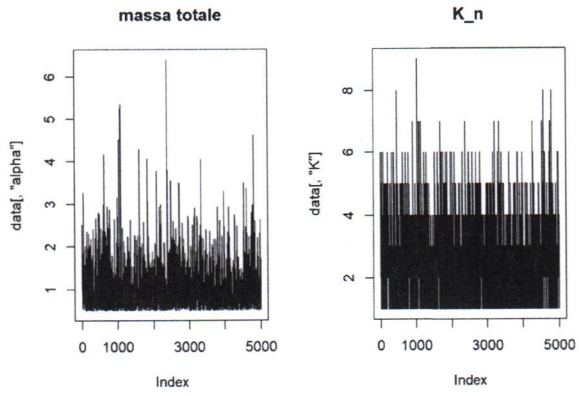
```
mean(data[,'beta.2.']<0)
```

```
## [1] 0.668
```

```
x11()
par(mfrow=c(3,3)) # qualche traceplot degli effetti casuali
plot(data[,'bb.1.'],main='b1',type='l')
plot(data[,'bb.2.'],main='b2',type='l')
plot(data[,'bb.3.'],main='b3',type='l')
plot(data[,'bb.7.'],main='b7',type='l')
plot(data[,'bb.10.'],main='b10',type='l')
plot(data[,'bb.13.'],main='b13',type='l')
plot(data[,'bb.15.'],main='b15',type='l')
plot(data[,'bb.16.'],main='b16',type='l')
plot(data[,'bb.17.'],main='b17',type='l')
```

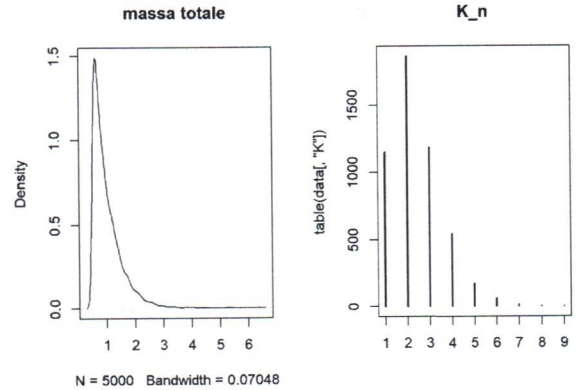



```
x11()
par(mfrow=c(1,2))
plot(data[,'alpha'],main='massa totale',type='l') # se ho assegnato una prior per alpha
plot(data[,'K'],main='K_n',type='l')
```




```
x11()
par(mfrow=c(1,2))
plot(density(data[,'alpha']),main='massa totale')
plot(table(data[,'K']),main='K_n')
```

Posterior distribution of the number of clusters:




```
mean(alpha);var(alpha)
```

```
## [1] 1.024756
```

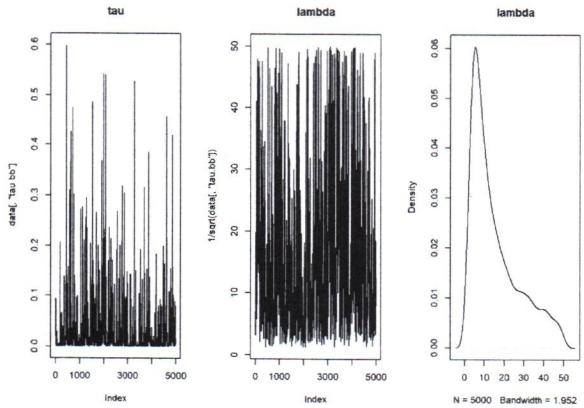
```
## [1] 0.3070238
```

```
mean(data[,'K']);var(data[,'K'])
```

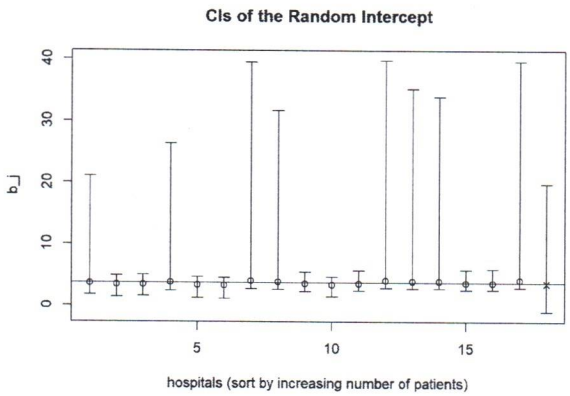
```
## [1] 2.3966
```

```
## [1] 1.35558
```

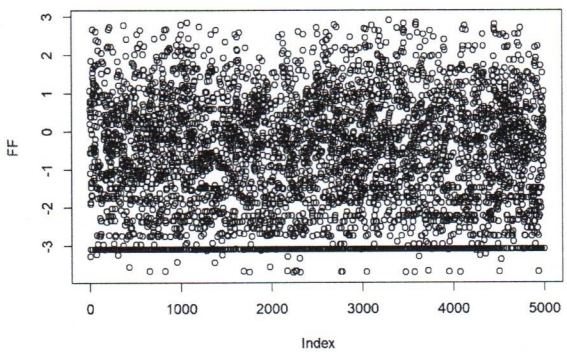
```
x11()
par(mfrow=c(1,3))
plot(data[,'tau.bb'],main='tau',type='l')
plot(1/sqrt(data[,'tau.bb']),main='lambda',type='l')
plot(density(1/sqrt(data[,'tau.bb'])),main='lambda')
```




```
#######################################
##  Plot IC per intercetta casuale  ##
#######################################
## ordered by number of patients
# creo un vettore contenente il numero di pazienti per ogni ospedale
patients=rep(0,J)
for(i in 1:J){
    patients[i] = length(which(input$centro==i))
}
sort_patients = sort(patients,index.return=T)   # ordino

# ricalcolo i quantili per ogni ospedale in ordine di numero di pazienti
Q = matrix(nrow=J+1, ncol=3)
for (j in 1:J){
    Q[j,] = quantile(data[, 2 + sort_patients$ix[j]], probs=c(0.025,0.5,0.975))
}
Q[J+1,] = quantile(data$newcentro,probs=c(0.025,0.5,0.975))
colnames(Q) <- c("2.5","median","97.5")

# symbol for point estimate: all points are round, the last point is x (new random hospital)
pch=c(rep(21,J),4)
x11()
plotCI(x=seq(1,J+1),y=Q[,2],uiw=(Q[,3]-Q[,2]) ,liw=(Q[,2]-Q[,1]),pch=pch,
       scol=1 , xlab="hospitals (sort by increasing number of patients)",
       ylab="b_j", main="CIs of the Random Intercept")
abline(h=mean(Q[,2]))
```



```
################################
# Cluster estimate di Lau&Green #
################################
#Load('M1.Rdata') # load coda output
#data=as.matrix(M1)
#data=data.frame(data)  # convert into a dataframe
J         = 17
label.mat = as.matrix(data[,3:19]) # extract cluster labels
m         = J
G         = n.chain
pihat     = matrix(0,ncol=m,nrow=m)
for(i in 1:G){
     ss    <- label.mat[i,]
     cij   <- outer(ss,ss,'==')
     pihat <- pihat+cij
}
pihat <- pihat/G

##### Binder loss function
FF <- vector("numeric")
K  <- 0.7 #prova con K=0.5, >=0.7
for(i in 1:G){
     ss    <- label.mat[i,]
     cij   <- outer(ss,ss,'==')
     pluto <- (pihat-K)*as.matrix(cij)
     pluto <-  pluto[upper.tri(pluto)]
     FF[i] <- sum(pluto)
}
plot(FF)
```




```
ind.bind <- which.max(FF)[1]
label.mat[ind.bind,]#Leti(ind.bind,L,G,m)
```

```
##    bb.1.    bb.2.    bb.3.    bb.4.    bb.5.    bb.6.    bb.7.    bb.8.
## 7.447876 9.137530 7.447876 9.137530 2.185216 2.185216 2.185216 2.185216
##    bb.9.   bb.10.   bb.11.   bb.12.   bb.13.   bb.14.   bb.15.   bb.16.
## 9.137530 2.185216 2.185216 4.270183 2.185216 2.185216 2.185216 2.185216
##   bb.17.
## 7.519569
```

```
#plot(FF)
ll.bind <- label.mat[ind.bind,] #Leti(ind.bind,L,G,m)
unici <- unique(ll.bind)
unici
```

```
## [1] 7.447876 9.137530 2.185216 4.270183 7.519569
```

```
l.uni <- length(unici)# numero di gruppi stimato
l.uni
```

```
## [1] 5
```

```
ncl = l.uni
for(i in 1:ncl){
  print(as.numeric(which(ll.bind==unici[i])))
}
```

```
## [1] 1 3
## [1] 2 4 9
##  [1]  5  6  7  8 10 11 13 14 15 16
## [1] 12
## [1] 17
```

```
#############################################################################################
### Disegno le intercette casuali al primo livello con colori diversi a seconda del gruppo #
#############################################################################################
Sest = rep(0,J)
for(i in 1:ncl){
  Sest[as.numeric(which(ll.bind==unici[i]))] = i
}
Sest
```

```
##  [1] 1 2 1 2 3 3 3 3 2 3 3 4 3 3 3 3 5
```

```
table(Sest)
```

```
## Sest
##  1  2  3  4  5
##  2  3 10  1  1
```

```
nclusLG = length(unique(Sest))
nclusLG
```

```
## [1] 5
```

```
bins   = as.numeric(names(table(Sest)))
freqs  = as.vector(table(Sest))
J      = 17
label  = rep(0,J)
mylist = list()

for(i in 1:nclusLG){
    in_gr        = which(Sest==bins[i])
    label[in_gr] = i
    mylist[[i]]  = in_gr
}
label
```

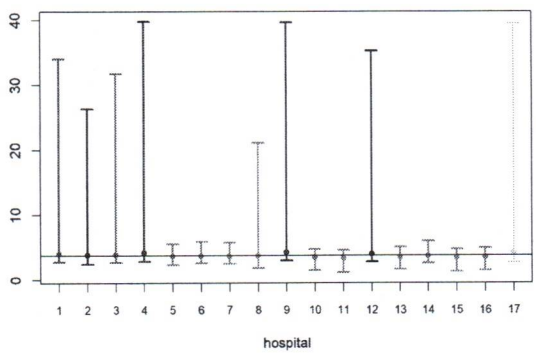
```
##  [1] 1 2 1 2 3 3 3 3 2 3 3 4 3 3 3 3 5
```

```
mylist
```

```
## [[1]]
## [1] 1 3
##
## [[2]]
## [1] 2 4 9
##
## [[3]]
##  [1]  5  6  7  8 10 11 13 14 15 16
##
## [[4]]
## [1] 12
##
## [[5]]
## [1] 17
```

Final clusters

```
### Faccio la Fig 11 come la Fig4 ma disegnando solo i 33 parametri
### Ogni colore è un gruppo
gr1 <- mylist[[1]]
gr2 <- mylist[[2]]
gr3 <- mylist[[3]]
gr4 <- mylist[[4]]
gr5 <- mylist[[5]]

colore = rep('ciao',J)
colore[gr1]='magenta'
colore[gr2]='red'
colore[gr3]='green'
colore[gr4]='blue'
colore[gr5]='yellow'

pch=rep(20,J)
# Nella matrice 'data' dalla colonna 3 in poi ci sono i b,
# gli effetti casuali (hospital-specific) al primo livello
# Per verificare il contenuto delle colonne di 'data': names(data)
# ricalcolo i quantili per ogni ospedale in ordine di numero di pazienti
Q = matrix(nrow=J, ncol=3) #matrix(nrow=J, ncol=3)
for (j in 1:J){
    Q[j,] = quantile(data[, 2 + j ],probs=c(0.025,0.5,0.975))
}

x11()
plotCI(x=seq(1,J),y=Q[,2],uiw=(Q[,3]-Q[,2]) ,liw=(Q[,2]-Q[,1]),pch=pch,xaxt='n',
       col=colore,scol=colore,xlab="hospital",ylab=' ', main=" ", lwd=2)
axis(side=1,at=seq(1,J),labels=seq(1,J),cex.axis=0.8)

# media di tutte le mediane
media <- mean(Q[,2])
abline(h= media)
```



– Here the colors correspond to the clusters