Freni-Sterrantino et al 2017 - BYM2 connected, disconnected for Scotland Lip Cancer Dataset

The BYM2 model for areal data adds to components to a GLM: an ICAR component which accounts for the spatial structure of the data, and a random effects component. See the Stan case study Spatial Models in Stan: Intrinsic Auto-Regressive Models for Areal Data for details on the ICAR, BYM, and BYM2 models. This implementation assumes that the spatial structure is a single, fully connected component, i.e., a graph where any node in the graph can be reached from any other node.

In A note on intrinsic Conditional Autoregressive models for disconnected graphs, Freni-Sterrantino et.al. show how to implement this model for disconnected graphs. In this notebook, we present that Stan implementation of this proposal.

Areal data: the counties in Scotland, circa 1980

The canonical dataset used to test and compare different parameterizations of ICAR models is a study on the incidence of lip cancer in Scotland in the 1970s and 1980s. The data, including the names and coordinates for the counties of Scotland are available from R package SpatialEpi, dataset `scotland` .

3 of these counties are islands: the Outer Hebrides (western.isles), Shetland, and Orkney. In the canonical datasets, these islands are conntected to the mainland, so that the adjacency graph consists of a single, fully connected component. However, different maps are possible: a map with 4 components, the mainland and the 3 islands; or a map with 3 components: the mainland, a component consisting of Shetland and Orkney, and a singleton consisting of the Hebrides. The following plots demonstrate the differences:

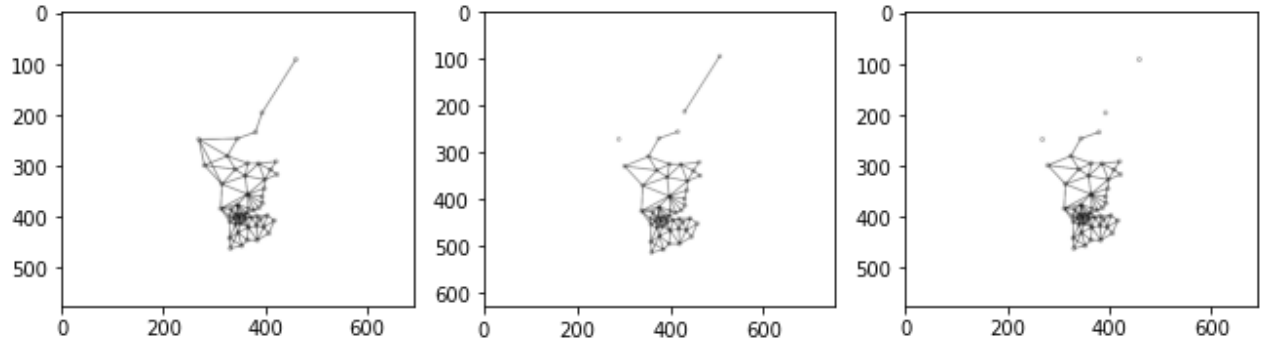
In [1]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from matplotlib import rcParams

%matplotlib inline

# figure size in inches optional
rcParams['figure.figsize'] = 11 ,8

# read images
img_A = mpimg.imread('scot_connected.png')
img_B = mpimg.imread('scot_3_comp.png')
img_C = mpimg.imread('scot_islands.png')


# display images
fig, ax = plt.subplots(1,3)
ax[0].imshow(img_A);
ax[1].imshow(img_B);
ax[2].imshow(img_C);
```



Areal data munging: from spatial polygon to 2D array of edges

Inputs to the Stan model must match the set of variables declared in the `data` block.

The Stan implementation of the ICAR model computes with a 2D array of size 2 $\times$ J where J is the number of edges in the graph. Each column entry in this array represents one undirected edge in the graph, where for each edge i, entries [i,1] and [i,2] index the nodes connected by that edge. Treating these are parallel arrays and using Stan's vectorized operations provides a transparent implementation of the pairwise difference formula used to compute the ICAR component.

The `scotland` data is a set of spatial polygons, i.e., a description of the shape of each county in terms of its lat,lon coordinates. The R package spdep extracts the adjacency relations as a `nb` object. We have written a set of helper functions which take the `nb` objects for each graph into the set of data structures needed by the Stan models, these are in file `bym2_helpers.R` .
The three versions of the Scotland spatial structure are in files `scotland_nbs.data.R` , `scotland_3_comp_nbs.data.R` , and `scotland_islands_nbs.data.R` . The file `munge_scotland.R` munges the data, and it has been saved as JSON data files.

Fit connected graph on Scotland Lip cancer dataset with BYM2 model implemented in Stan.

In [2]:
```
from cmdstanpy import cmdstan_path, CmdStanModel, install_cmdstan
# install_cmdstan()  # as needed - will install latest release (as needed)
```

The dataset `scot_connected.data.json` contains the cancer dataset together with the spatial structure. The cancer study data is:

- y : observed outcome - number of cases of lip cancer
- x : single predictor - percent of population working in agriculture, forestry, or fisheries.
- E : population

The spatial structure is comprised of:

- I: `int<lower = 0> I;  // number of nodes`
- J: `int<lower = 0> J;  // number of edges`

- edges: `int<lower = 1, upper = I> edges[2, J];  // node[1, j] adjacent to node[2, j]`
- tau: `real tau; // scaling factor`

The helper function `nb_to_edge_array` takes the `nb` object and returns the 2 × J edge array; the helper function `scaling_factor` uses the edge array to compute the geometric mean of the corresponding adjacency matrix. For the fully connected Scotland graph, the spatial data inputs are:

```
  "I": 56,
  "J": 132,

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

  "tau": 0.4853,
```

In [3]:
```
from cmdstanpy import cmdstan_path, CmdStanModel
bym2_model = CmdStanModel(stan_file='bym2.stan')
bym2_fit = bym2_model.sample(data='scot_connected.data.json')
```

```
INFO:cmdstanpy:found newer exe file, not recompiling
INFO:cmdstanpy:compiled model file: /Users/mitzi/github/stan-dev/example-models/
knitr/car-iar-poisson/update_2021_02/bym2
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:start chain 2
INFO:cmdstanpy:start chain 3
INFO:cmdstanpy:start chain 4
INFO:cmdstanpy:finish chain 3
INFO:cmdstanpy:finish chain 2
INFO:cmdstanpy:finish chain 1
INFO:cmdstanpy:finish chain 4
```

In [4]:
```
bym2_fit.summary()
```

Out[4]:

name	Mean	MCSE	StdDev	5%	50%	95%	N_Eff	N_Eff/s	R_hat
lp__	750.000	0.28000	9.000	740.000	750.000	770.0000	1100.0	95.00	1.0

name	Mean	MCSE	StdDev	5%	50%	95%	N_Eff	N_Eff/s	R_hat
alpha	-0.220	0.00270	0.120	-0.420	-0.220	-0.0081	2200.0	190.00	1.0
beta	0.037	0.00029	0.013	0.014	0.037	0.0570	2000.0	180.00	1.0
rho	0.870	0.00600	0.140	0.580	0.930	1.0000	570.0	50.00	1.0
sigma	0.710	0.00180	0.059	0.620	0.710	0.8100	1000.0	92.00	1.0
...	...	...	...	...	...	...	...	...	...
y_rep[56,7]	1.400	0.01900	1.300	0.000	1.000	4.0000	4234.0	375.00	1.0
y_rep[56,8]	1.400	0.02000	1.300	0.000	1.000	4.0000	3762.0	334.00	1.0
y_rep[56,9]	1.300	0.01900	1.200	0.000	1.000	4.0000	4305.0	382.00	1.0
y_rep[56,10]	1.300	0.02000	1.300	0.000	1.000	4.0000	3958.0	351.00	1.0
logit_rho	2.980	0.08000	2.210	0.300	2.530	7.2200	799.4	70.89	1.0

846 rows × 9 columns

In [5]:
```
bym2_fit.diagnose()
```

```
INFO:cmdstanpy:Processing csv files: /var/folders/db/4jnggnf549s42z50bd61jskm000
0gq/T/tmpid9c9ukr/bym2-202102192035-1-ml782znn.csv, /var/folders/db/4jnggnf549s4
2z50bd61jskm0000gq/T/tmpid9c9ukr/bym2-202102192035-2-f3pjyaw6.csv, /var/folders/
db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid9c9ukr/bym2-202102192035-3-cvjt73a2.cs
v, /var/folders/db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid9c9ukr/bym2-20210219203
5-4-h9k5rqat.csv

Checking sampler transitions treedepth.
Treedepth satisfactory for all transitions.

Checking sampler transitions for divergences.
No divergent transitions found.

Checking E-BFMI - sampler transitions HMC potential energy.
E-BFMI satisfactory for all transitions.

Effective sample size satisfactory.

Split R-hat values satisfactory all parameters.

Processing complete, no problems detected.
```

Out[5]:
```
'Processing csv files: /var/folders/db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid9c9
ukr/bym2-202102192035-1-ml782znn.csv, /var/folders/db/4jnggnf549s42z50bd61jskm00
00gq/T/tmpid9c9ukr/bym2-202102192035-2-f3pjyaw6.csv, /var/folders/db/4jnggnf549s
42z50bd61jskm0000gq/T/tmpid9c9ukr/bym2-202102192035-3-cvjt73a2.csv, /var/folder
s/db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid9c9ukr/bym2-202102192035-4-h9k5rqat.c
sv\n\nChecking sampler transitions treedepth.\nTreedepth satisfactory for all tr
ansitions.\n\nChecking sampler transitions for divergences.\nNo divergent transi
tions found.\n\nChecking E-BFMI - sampler transitions HMC potential energy.\nE-B
FMI satisfactory for all transitions.\n\nEffective sample size satisfactory.\n\n
Split R-hat values satisfactory all parameters.\n\nProcessing complete, no probl
ems detected.'
```

Fit disconnected graphs on Scotland Lip cancer dataset with BYM2 model implemented in Stan, following Freni-

Sterrantino

All components of the BYM2 model remain the same. The same 2 × J edges array encodes the spatial structure of the graph, and as before, the spatial prior `phi` is a vector, one element per node (areal unit). In order to compute the elements of `phi` on a per-component basis, we use Stan's multi-indexing operators. For each component, we provide a vector of indices into `phi` and into the edges_array. Because Stan doesn't have ragged arrays (yet), we construct two square matrices, where the number of rows in each are the number of components (including singletons) in the graph, and the number of columns are the number of nodes and edges, respectively. In addition we compute the per-component scaling factor.

The helper function `index_components` takes an `nb` object as input and returns the list of data structures needed by the `BYM2_islands.stan` model. We use the R package `jsonlite` to save this in JSON format. For the Scotland map with 3 components, in file `scotland_3_comps_nbs.data.R`, function `index_components` and `write_json` produce the following input data, used in addition to the number of nodes, edges, and edge array:

```
    "K":3,
    "K_node_cts":[53,2,1],
    "K_edge_cts":[126,1,0],
    "K_node_idxs":
[[1,2,3,4,5,7,9,10,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,


[6,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,


[11,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

    "K_edge_idxs":
[[1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28


[13,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0


[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,


  "tau": [0.4504, 0.25, 1],
```

In [6]:
```
from cmdstanpy import cmdstan_path, CmdStanModel
bym2_islands_model = CmdStanModel(stan_file='bym2_islands.stan')
```

```
INFO:cmdstanpy:found newer exe file, not recompiling
INFO:cmdstanpy:compiled model file: /Users/mitzi/github/stan-dev/example-models/
knitr/car-iar-poisson/update_2021_02/bym2_islands
```

In [7]:
```
print(bym2_islands_model.code())
```

```
functions {
  /**
```

```
   * Return the log probability density of the specified vector of
   * coefficients under the ICAR model with unit variance, where
   * adjacency is determined by the adjacency array and the spatial
   * structure is a disconnected graph which has at least one
   * connected component.  Each connected component has a
   * soft sum-to-zero constraint.

   * The spatial structure is described by a 2-D adjacency array
   * over the all edges in the areal map and a arrays of the
   * indices of per-component nodes and edges which are used as
   * masks into phi and the adjacency matrix.   Because the Stan
   * language lacks ragged arrays, these are all square matrices,
   * padded out with zeros; additional vectors record the number
   * of nodes and edges in each component.
   *
   * @param phi vector of varying effects
   * @param adjacency parallel arrays of indexes of adjacent elements of phi
   * @param node_cts array of sizes of per-component nodes
   * @param edge_cts array of sizes of per-component edges
   * @param node_idxs array of arrays of per_component node indexes.
   * @param edge_idxs array of arrays of per_component edge indexes.
   *
   * @return ICAR log probability density
   *
   * @reject if the the adjacency matrix does not have two rows
   * @reject if size mismatch between indexing arrays
   * @reject if size mismatch between phi and node indexes columns.
   */
  real standard_icar_disconnected_lpdf(vector phi,
                                       int[ , ] adjacency,
                                       int[ ] node_cts,
                                       int[ ] edge_cts,
                                       int[ , ] node_idxs,
                                       int[ , ] edge_idxs) {
    if (size(adjacency) != 2)
      reject("require 2 rows for adjacency array;",
             " found rows = ", size(adjacency));

    if (!(size(phi) == dims(node_idxs)[2]
          && size(node_cts) == size(edge_cts)
          && size(node_cts) == size(node_idxs)
          && size(edge_cts) == size(edge_idxs)))
      reject("bad graph indexes, expecting ",
             size(node_cts),
             " rows for node and edge index matrices;",
             " inputs have ",
             size(node_cts),
             " and ",
             size(edge_cts),
             " respectively.");

    real total = 0;
    for (n in 1:size(node_cts)) {
      if (node_cts[n] > 1)
        total += -0.5 * dot_self(phi[adjacency[1, edge_idxs[n, 1:edge_cts[n]]]]
-
                                 phi[adjacency[2, edge_idxs[n, 1:edge_cts[n]]]])
          + normal_lpdf(sum(phi[node_idxs[n, 1:node_cts[n]]]) | 0, 0.001 * node_
cts[n]);
    }
    return total;
  }
}
data {
  // spatial structure
```

```
  int<lower = 0> I;  // number of nodes
  int<lower = 0> J;  // number of edges
  int<lower = 1, upper = I> edges[2, J];  // node[1, j] adjacent to node[2, j]

  int<lower=0, upper=I> K;  // number of components in spatial graph
  int<lower=0, upper=I> K_node_cts[K];   // per-component nodes
  int<lower=0, upper=J> K_edge_cts[K];   // per-component edges
  int<lower=0, upper=I> K_node_idxs[K, I];  // rows contain per-component node i
ndexes
  int<lower=0, upper=J> K_edge_idxs[K, J];  // rows contain per-component edge i
ndexes

  vector[K] tau; // scaling factor

  int<lower=0> y[I];              // count outcomes
  vector<lower=0>[I] E;           // exposure
  vector[I] x;                 // predictor
}
transformed data {
  vector[I] log_E = log(E);
}
parameters {
  real alpha;            // intercept
  real beta;       // covariates

  // spatial effects
  real<lower=0, upper=1> rho; // proportion unstructured vs. spatially structure
d variance
  real<lower = 0> sigma;  // scale of spatial effects
  vector[I] theta;  // standardized heterogeneous spatial effects
  vector[I] phi;  // standardized spatially smoothed spatial effects
}
transformed parameters {
  vector[I] gamma;
  // each component has its own spatial smoothing
  // singleton nodes have distribution normal(0, 1/sqrt(K))
  for (k in 1:K) {
    if (K_node_cts[k] == 1) {
      gamma[K_node_idxs[k,1]] =
        theta[K_node_idxs[k,1]] + normal_lpdf(phi[K_node_idxs[k,1]] | 0, inv_sqr
t(K));
    } else {
      gamma[K_node_idxs[k, 1:K_node_cts[k]]] =
            (sqrt(1 - rho) * theta[K_node_idxs[k, 1:K_node_cts[k]]]
             + (sqrt(rho) * sqrt(1 / tau[k])
                * phi[K_node_idxs[k, 1:K_node_cts[k]]]) * sigma);
    }
  }
}
model {
  y ~ poisson_log(log_E + alpha + x * beta + gamma * sigma);  // co-variates

  alpha ~ normal(0, 1);
  beta ~ normal(0, 1);

  // spatial hyperpriors and priors
  sigma ~ normal(0, 1);
  rho ~ beta(0.5, 0.5);
  theta ~ normal(0, 1);
  phi ~ standard_icar_disconnected(edges, K_node_cts, K_edge_cts, K_node_idxs, K
_edge_idxs);
}
generated quantities {
  // posterior predictive checks
  vector[I] eta = log_E + alpha + x * beta + gamma * sigma;
```

```
  vector[I] y_prime = exp(eta);
  int y_rep[I,10];
  for (j in 1:10) {
    if (max(eta) > 20) {
      // avoid overflow in poisson_log_rng
      print("max eta too big: ", max(eta));
      for (i in 1:I)
        y_rep[i,j] = -1;
    } else {
      for (i in 1:I)
        y_rep[i,j] = poisson_log_rng(eta[i]);
    }
  }
  real logit_rho = log(rho / (1.0 - rho));
}
```

In [8]:
```
import json
with open('scot_3_comp.data.json') as fd:
    scot_data = json.load(fd)
# print(scot_data)
```

In [9]:
```
bym2_islands_fit = bym2_islands_model.sample(data=scot_data, max_treedepth=11)
bym2_islands_fit.summary()
```

```
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:start chain 2
INFO:cmdstanpy:start chain 3
INFO:cmdstanpy:start chain 4
INFO:cmdstanpy:finish chain 4
INFO:cmdstanpy:finish chain 2
INFO:cmdstanpy:finish chain 3
INFO:cmdstanpy:finish chain 1
```

Out[9]:

name	Mean	MCSE	StdDev	5%	50%	95%	N_Eff	N_Eff/s	R_hat
lp__	760.000	0.30000	9.200	740.000	760.000	770.000	930.0	15.00	1.0
alpha	-0.320	0.00260	0.130	-0.530	-0.330	-0.120	2300.0	37.00	1.0
beta	0.048	0.00029	0.013	0.026	0.048	0.069	2000.0	32.00	1.0
rho	0.850	0.00660	0.150	0.560	0.890	1.000	490.0	7.80	1.0
sigma	0.680	0.00240	0.068	0.580	0.680	0.800	780.0	12.00	1.0
...	...	...	...	...	...	...	...	...	...
y_rep[56,7]	1.400	0.02000	1.300	0.000	1.000	4.000	4113.0	65.00	1.0
y_rep[56,8]	1.400	0.02000	1.300	0.000	1.000	4.000	4386.0	69.00	1.0
y_rep[56,9]	1.400	0.02000	1.300	0.000	1.000	4.000	4284.0	67.00	1.0
y_rep[56,10]	1.500	0.02000	1.300	0.000	1.000	4.000	4175.0	66.00	1.0
logit_rho	2.540	0.08000	1.960	0.260	2.100	6.410	570.6	8.97	1.0

846 rows × 9 columns

In [10]:
```
bym2_islands_fit.diagnose()
```

```
INFO:cmdstanpy:Processing csv files: /var/folders/db/4jnggnf549s42z50bd61jskm000
```

```
0gq/T/tmpid9c9ukr/bym2_islands-202102192036-1-lbgptmrv.csv, /var/folders/db/4jng
gnf549s42z50bd61jskm0000gq/T/tmpid9c9ukr/bym2_islands-202102192036-2-favo4t3a.cs
v, /var/folders/db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid9c9ukr/bym2_islands-202
102192036-3-zfrov9au.csv, /var/folders/db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid
9c9ukr/bym2_islands-202102192036-4-hcsc8dpn.csv

Checking sampler transitions treedepth.
Treedepth satisfactory for all transitions.

Checking sampler transitions for divergences.
No divergent transitions found.

Checking E-BFMI - sampler transitions HMC potential energy.
E-BFMI satisfactory for all transitions.

Effective sample size satisfactory.

Split R-hat values satisfactory all parameters.

Processing complete, no problems detected.
```

Out[10]:
```
'Processing csv files: /var/folders/db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid9c9
ukr/bym2_islands-202102192036-1-lbgptmrv.csv, /var/folders/db/4jnggnf549s42z50bd
61jskm0000gq/T/tmpid9c9ukr/bym2_islands-202102192036-2-favo4t3a.csv, /var/folder
s/db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid9c9ukr/bym2_islands-202102192036-3-zf
rov9au.csv, /var/folders/db/4jnggnf549s42z50bd61jskm0000gq/T/tmpid9c9ukr/bym2_is
lands-202102192036-4-hcsc8dpn.csv\n\nChecking sampler transitions treedepth.\nTr
eedepth satisfactory for all transitions.\n\nChecking sampler transitions for di
vergences.\nNo divergent transitions found.\n\nChecking E-BFMI - sampler transit
ions HMC potential energy.\nE-BFMI satisfactory for all transitions.\n\nEffectiv
e sample size satisfactory.\n\nSplit R-hat values satisfactory all parameters.\n
\nProcessing complete, no problems detected.'
```

In [ ]: