RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of the Art

inlabru vs INLA

Thierry Onkelinx

RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of
the Art

Introduction

Concept

- first a comparison between `INLA` and `inlabru`
- then time to redo the challenges from the first workshop using `inlabru`

Slides, code and data available on `https://inbo.github.io/tutorials/tutorials/r_inla/`

INLA or inlabru?

- `inlabru` wrapper around `INLA`
- taylored towards spatial data
  - spatially stuff will handled in the next tutorial
- some stuff is easier / better
- some stuff is harder / more awkward

Toy data set

- Tundra bean goose (*Anser fabalis subsp. rossicus*)
- Subset of wintering waterbirds in Flanders (`https://doi.org/10.15468/lj0udq`)

```
readRDS("anser_fabalis_rossicus.Rds") %>%
  mutate(cyear = year - max(year)) -> goose
glimpse(goose)
```

```
## Observations: 1,502
## Variables: 7
## $ location_id <int> 1010802, 1010803, 1010804, 1011201, 1011203, 1011204, 1...
## $ year        <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2...
## $ month       <fct> nov, nov, nov, nov, nov, nov, nov, nov, nov, nov, nov, ...
## $ count       <int> 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9...
## $ lat         <dbl> 50.97888, 50.99586, 50.97322, 50.96474, 50.94772, 50.93...
## $ long        <dbl> 2.824395, 2.841892, 2.858071, 2.807533, 2.825685, 2.807...
## $ cyear       <int> -15, -15, -15, -15, -15, -15, -15, -15, -15, -15, -15, ...
```

RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of
the Art

Fixed effects only

Similar syntax

- *WAIC* and *DIC* are calculated by default

```
library(INLA)
m0_inla <- inla(count ~ cyear, data = goose, family = "nbinomial",
                control.compute = list(waic = TRUE, dic = TRUE))
library(inlabru)
m0_inlabru <- bru(count ~ cyear, data = goose, family = "nbinomial")
```

inlabru returns augmented INLA object

```r
class(m0_inla)
```

```
## [1] "inla"
```

```r
class(m0_inlabru)
```

```
## [1] "bru"   "iinla" "inla"  "list"
```

```r
all(names(m0_inla) %in% names(m0_inlabru))
```

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

```r
names(m0_inlabru)[!names(m0_inlabru) %in% names(m0_inla)]
```

```
## [1] "stack" "model" "sppa"
```

Careful with factor variables

```
m1_wrong <- bru(count ~ cyear + month, data = goose, family = "nbinomial")
```

```
## Warning in `[<-.factor`(`*tmp*`, is.na(xx), value = 0): invalid factor level, NA
## generated
```

```
m1_wrong$summary.fixed
```

```
##                    mean           sd    0.025quant      0.5quant   0.975quant
## cyear      1.341633e-01 2.599651e-02  8.320225e-02  1.341299e-01 1.852616e-01
## month     -7.062044e-10 4.917725e-09 -1.036138e-08 -7.063511e-10 8.940923e-09
## Intercept  9.363395e-03 3.168145e+01 -6.219220e+01  8.418032e-03 6.215914e+01
##                    mode          kld
## cyear      1.340656e-01 9.206079e-07
## month     -7.062289e-10 1.228088e-09
## Intercept  9.203327e-03 3.563600e-10
```

inlabru requires dummy variables in case of factors

```
model.matrix(~month, goose) %>% # create dummy variable for month
  as.data.frame() %>%
  select(-1) %>% # drop intercept
  bind_cols(goose) -> goose
m1_inlabru <- bru(count ~ cyear + monthdec + monthjan + monthfeb, data = goose,
                  family = "nbinomial")
```

```
m1_inlabru$summary.fixed
```

```
##                mean        sd 0.025quant  0.5quant 0.975quant      mode
## cyear     0.1772642 0.0256012  0.1269679 0.1772695  0.2274817 0.1772822
## monthdec  2.1162171 0.3366472  1.4551835 2.1161638  2.7769379 2.1160844
## monthjan  2.7617469 0.3435883  2.0860347 2.7620561  3.4350695 2.7626995
## monthfeb  2.3542381 0.3316482  1.7019398 2.3545597  3.0041010 2.3552274
## Intercept 2.6787414 0.2592559  2.1959161 2.6691879  3.2154645 2.6501179
##                    kld
## cyear     5.132158e-07
## monthdec  3.306802e-07
## monthjan  8.581558e-07
## monthfeb  8.729663e-07
## Intercept 2.324351e-06
```

RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of
the Art

Random effects

`inlabru` works slightly different

- use any name instead of `f()`
- use `map` to link the random effect to the data (cfr. `ggplot2::aes()`)
- use integer indices in case of a factor
- you need to provide the number of levels in case of a factor

Example of random intercept

```
comp_inla <- count ~ cyear + month + f(location_id, model = "iid")
m2_inla <- inla(comp_inla, data = goose, family = "nbinomial",
                control.compute = list(waic = TRUE, dic = TRUE))
goose <- mutate(goose, loc_id = as.integer(factor(location_id)))
n_loc <- max(goose$loc_id)
comp_inlabru <- count ~ cyear + monthdec + monthjan + monthfeb +
  site(map = loc_id, model = "iid", n = n_loc)
m2_inlabru <- bru(comp_inlabru, data = goose, family = "nbinomial")
```

inlabru allows to reuse a variable

```
mutate(goose, cyear = cyear - min(cyear) + 1, cyear2 = cyear) -> goose2
comp_inla <- count ~ cyear + f(cyear2, model = "iid") + month +
  f(location_id, model = "iid")
m3_inla <- inla(comp_inla, data = goose2, family = "nbinomial",
                control.compute = list(waic = TRUE, dic = TRUE))

n_year <- max(goose2$cyear)
comp_inlabru <- count ~ cyear + rtrend(map = cyear, model = "iid", n = n_year) +
  monthdec + monthjan + monthfeb + site(map = loc_id, model = "iid", n = n_loc)
m3_inlabru <- bru(comp_inlabru, data = goose2, family = "nbinomial")
```

map applies function on the fly

```
comp_inlabru <- count ~  lintrend(map = cyear, model = "linear") +
  quadtrend(map = cyear ^ 2, model = "linear") +
  rtrend(map = cyear, model = "iid", n = n_year) +
  monthdec + monthjan + monthfeb + site(map = loc_id, model = "iid", n = n_loc)
m3_inlabru2 <- bru(comp_inlabru, data = goose2, family = "nbinomial")
```

```
m3_inlabru2$summary.fixed
```

```
##                  mean          sd    0.025quant    0.5quant 0.975quant
## lintrend  -0.04841537 0.125151238 -0.2979217955 -0.04712278 0.19369062
## quadtrend  0.01267985 0.006803506 -0.0005023783  0.01262024 0.02619449
## monthdec   1.76923681 0.354530119  1.0736612839  1.76900958 2.46545729
## monthjan   2.64643485 0.346377499  1.9682546788  2.64573633 3.32788160
## monthfeb   2.57626431 0.336510916  1.9147938213  2.57648588 3.23586704
## Intercept -0.25049045 0.607975049 -1.4152468561 -0.26021782 0.97132768
##                  mode          kld
## lintrend  -0.04455697 5.675233e-08
## quadtrend  0.01250255 9.275217e-08
## monthdec   1.76858637 1.450274e-06
## monthjan   2.64437130 1.299156e-06
## monthfeb   2.57695209 6.557225e-07
## Intercept -0.27948069 1.672619e-07
```

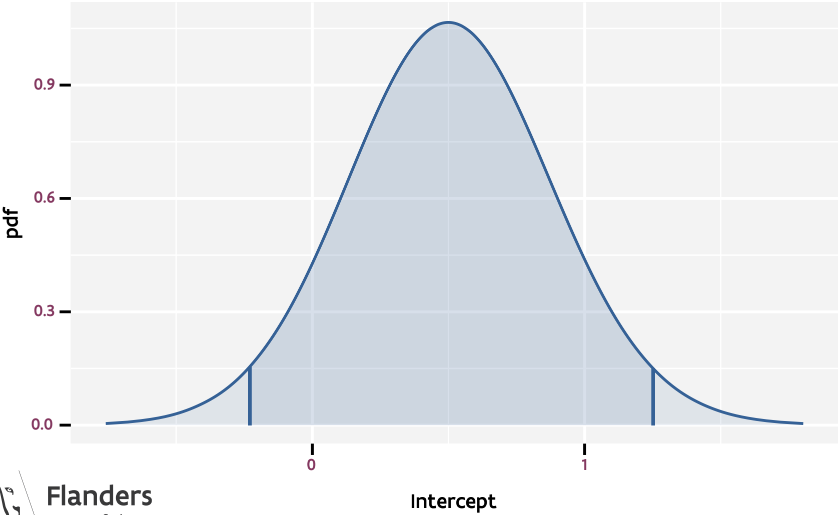
Plotting the model

Prepare a model with rw1 and iid component

```
pc_prior <- list(theta = list(prior = "pc.prec", param = c(1, 0.01)))
goose %>%
  mutate(iyear = cyear - min(cyear) + 1) -> goose
n_year <- max(goose$iyear)
comp_inlabru <- count ~ monthdec + monthjan + monthfeb +
  trend(map = iyear, model = "rw1", n = n_year, hyper = pc_prior) +
  site(map = loc_id, model = "iid", n = n_loc, hyper = pc_prior)
m5_inlabru <- bru(comp_inlabru, data = goose, family = "nbinomial")
```

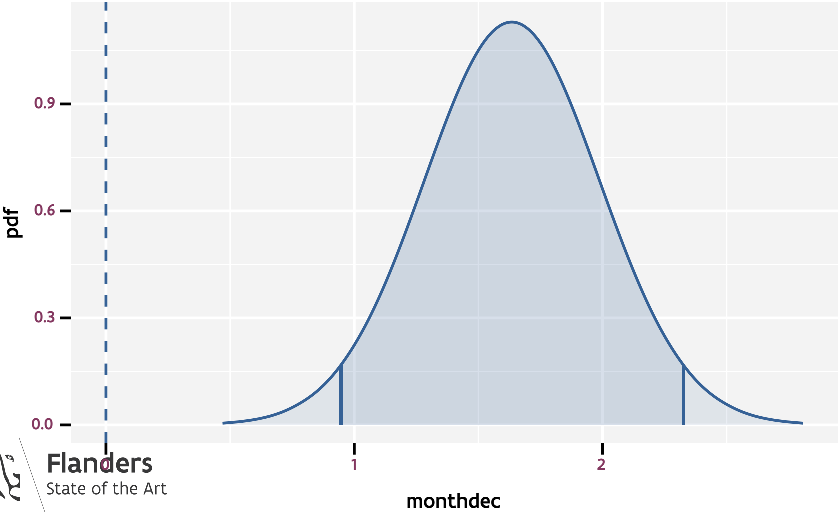
Plot the model

```
plot(m5_inlabru)
```

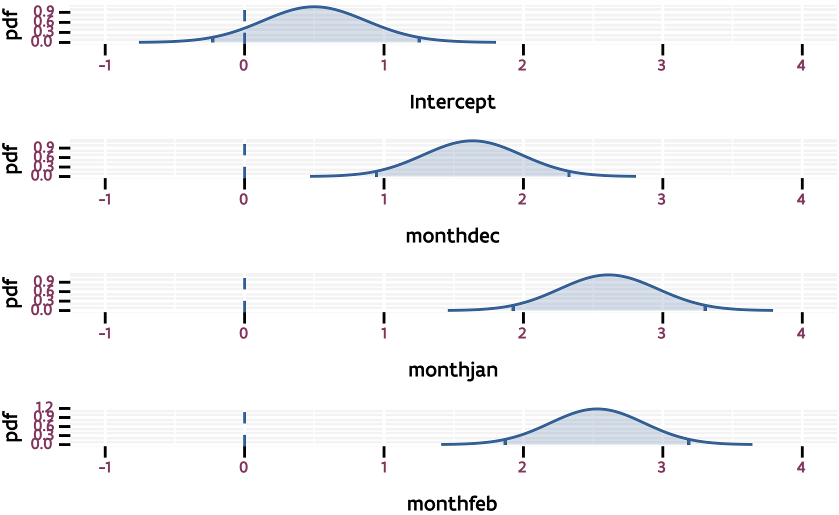


Plotting different fixed effect

```r
plot(m5_inlabru, "monthdec") +
  geom_vline(xintercept = 0, linetype = 2)
```



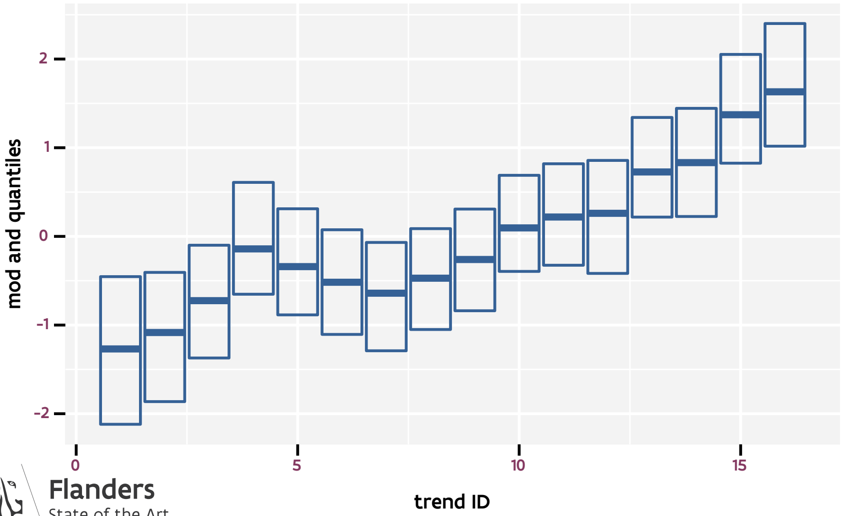
Combined plots with `multiplot()`



Flanders
State of the Art

Plotting a random effect

```r
plot(m5_inlabru, "trend")
```



Flanders
State of the Art

RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of
the Art

Predictions

inlabru has a predict() method

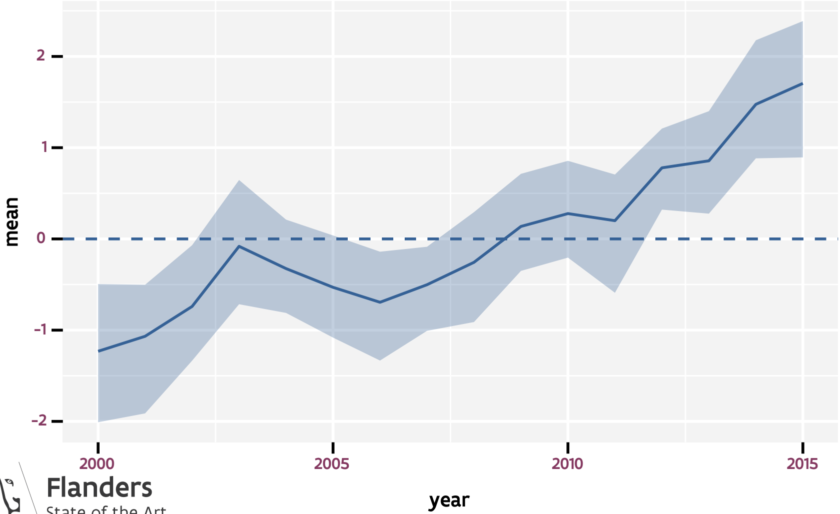
- no need to refit the model for predicting new data!
- works for inla(), bru() and lgcp() models
- you can specify which components of the models to be used

```
goose_trend <- distinct(goose, year, iyear)
pred_trend_log <- predict(m5_inlabru, data = goose_trend, formula = ~ trend)
glimpse(pred_trend_log)
```

```
## Observations: 16
## Variables: 11
## $ year   <int> 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, ...
## $ iyear  <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
## $ mean   <dbl> -1.23175072, -1.06739408, -0.74048282, -0.08115499, -0.32522...
## $ sd     <dbl> 0.4221745, 0.3723906, 0.3322472, 0.3302441, 0.2741831, 0.291...
## $ q0.025 <dbl> -2.0093384, -1.9126213, -1.3352629, -0.7159670, -0.8113579, ...
## $ median <dbl> -1.1970624, -0.9925642, -0.7404002, -0.1274860, -0.3505111, ...
## $ q0.975 <dbl> -0.49516614, -0.50223787, -0.06898507, 0.64582127, 0.2112645...
## $ smin   <dbl> -2.2638541, -2.0880575, -1.3861253, -0.9180012, -0.9115941, ...
## $ smax   <dbl> -0.478575390, -0.419883767, -0.012048394, 1.012219286, 0.438...
## $ cv     <dbl> -0.3427434, -0.3488783, -0.4486899, -4.0693014, -0.8430672, ...
## $ var    <dbl> 0.17823127, 0.13867476, 0.11038818, 0.10906117, 0.07517639, ...
```

Predictions are easy to plot

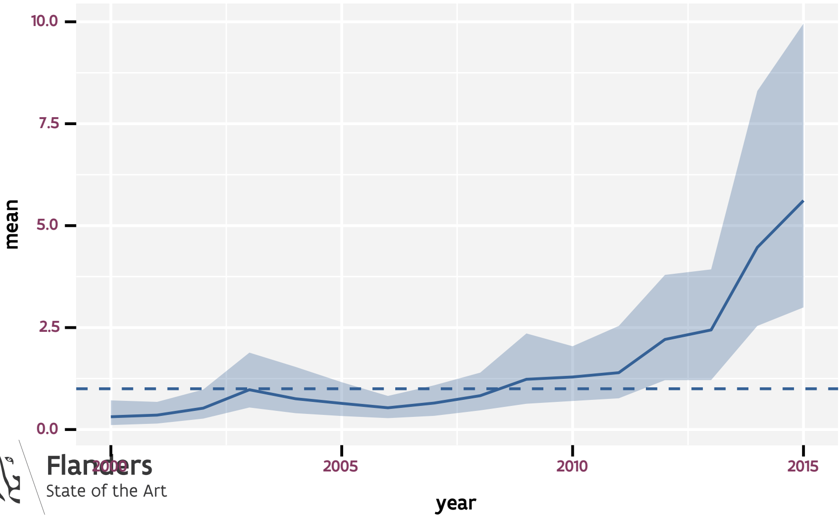
```
ggplot() + gg(pred_trend_log) + geom_hline(yintercept = 0, linetype = 2)
```



Flanders
State of the Art

prediction formula allows functions

```
pred_trend <- predict(m5_inlabru, data = goose_trend, formula = ~ exp(trend))
ggplot() + gg(pred_trend) + geom_hline(yintercept = 1, linetype = 2)
```



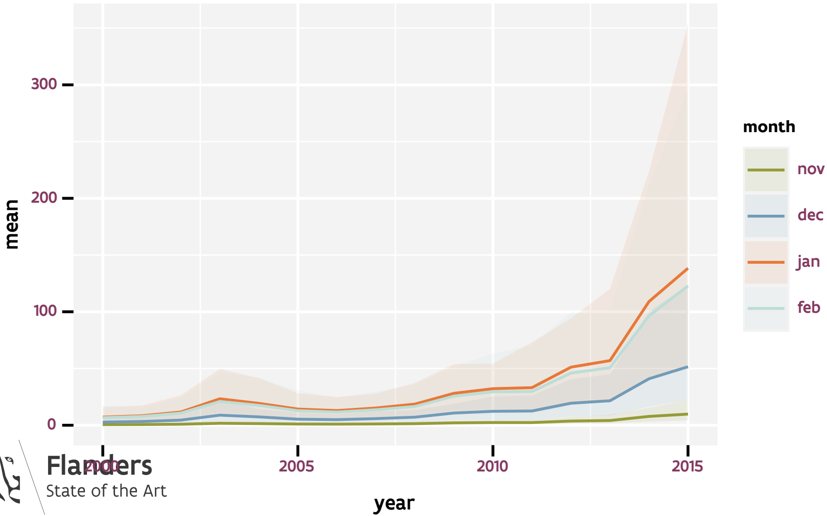
Flanders
State of the Art

Predictions can use multiple components

```r
goose_new <- distinct(goose, year, month, iyear, monthdec, monthjan, monthfeb)
pred_goose <- predict(
  m5_inlabru, data = goose_new,
  formula = ~ exp(Intercept + trend + monthdec + monthjan + monthfeb)
)
```

Multiple covariates require a bit more work to plot

```r
ggplot(pred_goose, aes(x = year, y = mean, ymin = q0.025, ymax = q0.975)) +
  geom_ribbon(aes(fill = month), alpha = 0.1) + geom_line(aes(colour = month))
```



Even aggregations are possible

```
goose_sum <- filter(goose, month == "jan")
predict(m5_inlabru, data = goose_sum,
        formula = ~aggregate(
          exp(Intercept + trend + site),
          by = list(year = goose_sum$year),
          FUN = sum))
```

```
##    year      mean        sd    q0.025    median     q0.975       smin
## 1  2000  23.29620  13.91122  6.364952  18.12885   55.20891   5.366907
## 2  2001  21.79465  12.42084  7.067240  18.69775   48.49705   6.351220
## 3  2002  43.95603  25.25318 14.519125  36.28225  100.09531  12.835054
## 4  2003  78.74028  35.14959 32.218192  70.91131  155.52017  29.078083
## 5  2004  86.13504  38.81957 33.648486  75.70753  184.38188  26.271995
## 6  2005  55.21897  25.48996 23.396520  50.55510  120.61886  17.472150
## 7  2006  59.89855  31.76588 19.826647  57.26883  114.31995  17.033754
## 8  2007  78.28305  38.53550 32.592463  72.59995  148.55553  25.287925
## 9  2008  76.24787  32.63353 30.396130  70.75631  153.71535  18.792296
## 10 2009 134.16117  62.54971 54.048634 121.78106  293.18191  49.028293
## 11 2010 130.25813  55.79101 55.570069 116.22095  288.89363  43.257109
## 12 2011 124.92821  53.75480 49.158919 110.79208  250.06548  38.384446
## 13 2012 207.29861  81.37327 97.590934 184.48784  389.60956  78.476653
## 14 2013 248.75104 100.89817 109.582875 217.65699  491.07363 106.031807
## 15 2014 481.83100 198.15584 249.149339 431.39198 1001.05298 230.478874
## 16 2015 643.68210 278.30712 327.711596 565.62176 1338.41897 262.665597
##            smax        cv       var
```