RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of the Art

Random intercept models with INLA

Thierry Onkelinx

Concept of the workshop

- 5 generic challenges
- everybody tries to tackle the challenge with own data
- stuck? ask your neighbour for help
- still stuck? ask me

slides, code, data and HackMD:
`https://inbo.github.io/tutorials/tutorials/r-inla`

RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of
the Art

Fixed effect model

Challenge 1

1. fit fixed effect model
2. extract *WAIC* from the model
3. display fixed effect parameters in a table

Prepare data

```
cars <- data.frame(
  fuel_consumption = 3.785411784 / 0.01609344 / mtcars$mpg, # liter / 100 km
  cc = mtcars$disp * 2.54 ^ 3, # engine displacement in cm³
  gearbox = factor(mtcars$am, levels = 0:1, labels = c("auto", "manual"))
)
summary(cars)
```

```
##  fuel_consumption       cc          gearbox
##  Min.   : 6.938   Min.   :1165   auto  :19
##  1st Qu.:10.316   1st Qu.:1980   manual:13
##  Median :12.251   Median :3217
##  Mean   :12.755   Mean   :3781
##  3rd Qu.:15.250   3rd Qu.:5342
##  Max.   :22.617   Max.   :7735
```

Solution 1

```r
library(INLA, quietly = TRUE)
```

```
##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

## This is INLA_18.07.12 built 2019-01-21 15:20:52 UTC.
## See www.r-inla.org/contact-us for how to get help.
## To enable PARDISO sparse library; see inla.pardiso()
```

```r
model <- inla(fuel_consumption ~ cc * gearbox, data = cars,
              control.compute = list(waic = TRUE))
model$waic$waic
```

```
## [1] 140.4934
```

Parameters

Table: model parameters

	mean	lcl	ucl
(Intercept)	6.96863	4.46417	9.47100
cc	0.00157	0.00108	0.00207
gearboxmanual	-0.92728	-4.18081	2.32436
cc:gearboxmanual	0.00023	-0.00068	0.00114

Scaling

- based on mean and standard deviation (standardise)

```
cars$cc_std <- scale(cars$cc)
attr(cars$cc_std, "scaled:center") # reference (cc)
```

```
## [1] 3780.854
```

```
attr(cars$cc_std, "scaled:scale") # scale (cc)
```

```
## [1] 2030.991
```

```
model_std <- inla(fuel_consumption ~ cc_std * gearbox, data = cars,
                  control.compute = list(waic = TRUE))
```

- based on carefully picked values

```
cars$liter <- cars$cc / 1000
cars$liter_c <- cars$liter - 4 # reference = 4 liter = 4000 cc
model_liter <- inla(fuel_consumption ~ liter_c * gearbox, data = cars,
                    control.compute = list(waic = TRUE))
```

Effect on parameters

Table: model parameters for different scaling

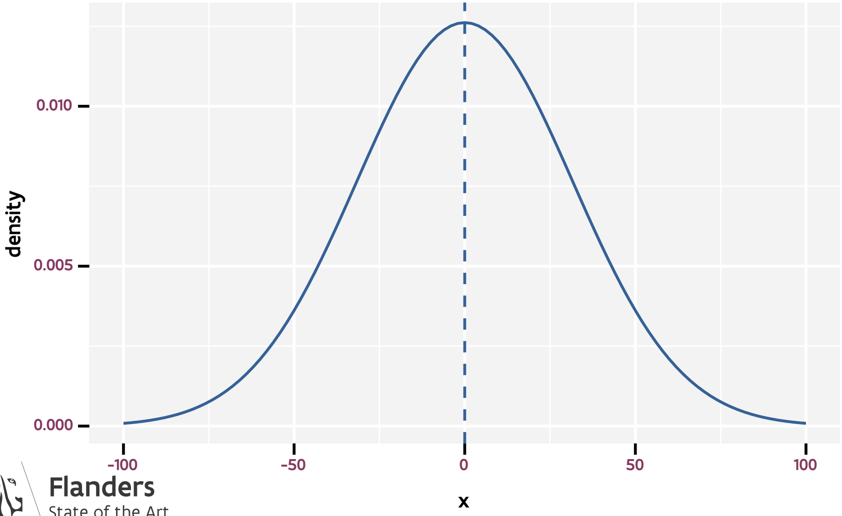
parameter	mean	mean_std	mean_liter
(Intercept)	6.96863	12.91607	13.26043
cc	0.00157	3.19316	1.57248
gearboxmanual	-0.92728	-0.07157	-0.02142
cc:gearboxmanual	0.00023	0.46172	0.22717

Challenge 2

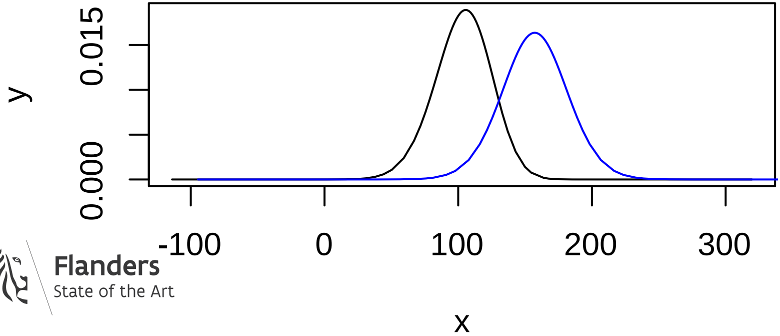
1. what is the default prior for a fixed effect (`?control.fixed`)
2. use a custom prior for a fixed effect (`?inla`)
3. specify two linear combinations r-inla.org, FAQ 17

Default prior for fixed effect

- $\mu = 0$
- $\tau = 0.001 \Rightarrow \sigma^2 = 1/\tau = 1000 \Rightarrow \sigma = 31.63$

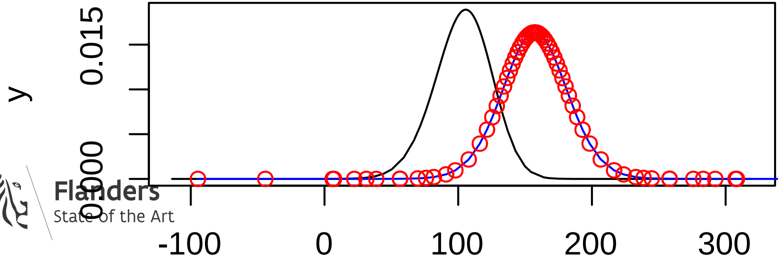


```r
cars$extreme <- cars$liter / 100
model_extreme <- inla(fuel_consumption ~ extreme * gearbox, data = cars)
z <- model_liter$marginals.fixed$liter_c
z[, "x"] <- z[, "x"] * 100
z[, "y"] <- z[, "y"] / 100
plot(model_extreme$marginals.fixed$extreme, type = "l")
lines(z, col = "blue")
```



Change fixed effect prior

```
model_extreme2 <- inla(fuel_consumption ~ extreme * gearbox, data = cars,
  control.fixed = list(
    mean = c(extreme = 100),
    prec = c(extreme = 1e-7, "extreme:gearboxmanual" = 1e-7)))
plot(model_extreme$marginals.fixed$extreme, type = "l")
lines(z, col = "blue")
points(model_extreme2$marginals.fixed$extreme, col = "red")
```



Linear combinations with fixed effects

```
combinations <- expand.grid(liter = pretty(cars$liter),
                            gearbox = unique(cars$gearbox)) %>%
  mutate(liter_c = liter - 4)
model.matrix(~ liter_c * gearbox, combinations) %>%
  as.data.frame() %>%
  inla.make.lincombs() %>%
  setNames(paste(combinations$gearbox, combinations$liter, sep = ":")) -> lc
model_lc <- inla(fuel_consumption ~ liter_c * gearbox, data = cars,
                 lincomb = lc, control.compute = list(waic = TRUE))
```

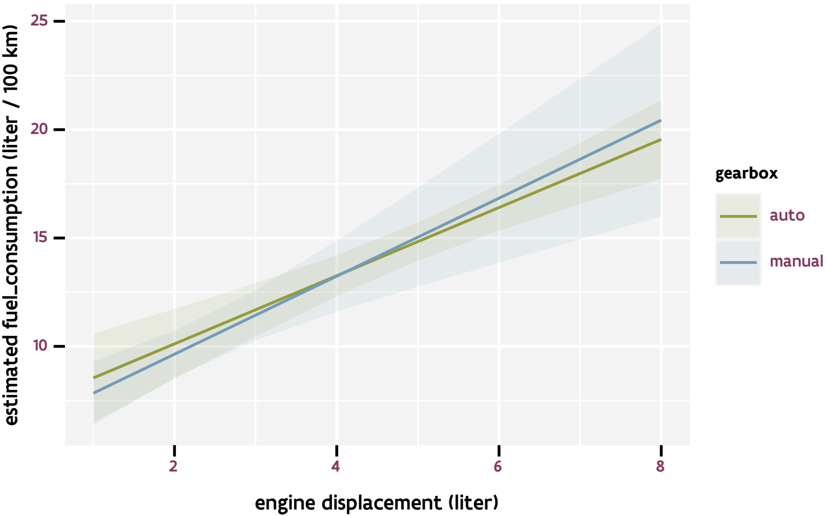
```r
model_lc$summary.lincomb #see ?control.inla
```

```
## data frame with 0 columns and 0 rows
```

```r
model_lc$summary.lincomb.derived
```

```
##           ID      mean        sd 0.025quant  0.5quant 0.975quant      mode kld
## manual:1   1  7.840054 0.7453199   6.366800  7.840029   9.312251  7.840050   0
## manual:2   2  9.639706 0.5482119   8.556069  9.639688  10.722562  9.639704   0
## manual:3   3 11.439357 0.5870562  10.278942 11.439339  12.598931 11.439358   0
## manual:4   4 13.239009 0.8293357  11.599689 13.238984  14.877135 13.239013   0
## manual:5   5 15.038660 1.1532669  12.759036 15.038626  17.316623 15.038667   0
## manual:6   6 16.838312 1.5070884  13.859298 16.838267  19.815153 16.838321   0
## manual:7   7 18.637964 1.8739451  14.933794 18.637908  22.339430 18.637976   0
## manual:8   8 20.437615 2.2474629  15.995123 20.437549  24.876865 20.437630   0
## auto:1     9  8.542993 1.0370579   6.493097  8.542950  10.591464  8.542964   0
## auto:2    10 10.115472 0.8175051   8.499552 10.115439  11.730265 10.115451   0
## auto:3    11 11.687951 0.6213738  10.459711 11.687927  12.915329 11.687938   0
## auto:4    12 13.260430 0.4783945  12.314805 13.260413  14.205381 13.260424   0
## auto:5    13 14.832909 0.4433763  13.956495 14.832895  15.708684 14.832911   0
## auto:6    14 16.405387 0.5378374  15.342246 16.405374  17.467741 16.405398   0
## auto:7    15 17.977866 0.7119837  16.570485 17.977850  19.384194 17.977884   0
## auto:8    16 19.550345 0.9217037  17.728407 19.550325  21.370915 19.550371   0
```

Plot linear combinations



www.INBO.be

Flanders
State of the Art

RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of
the Art

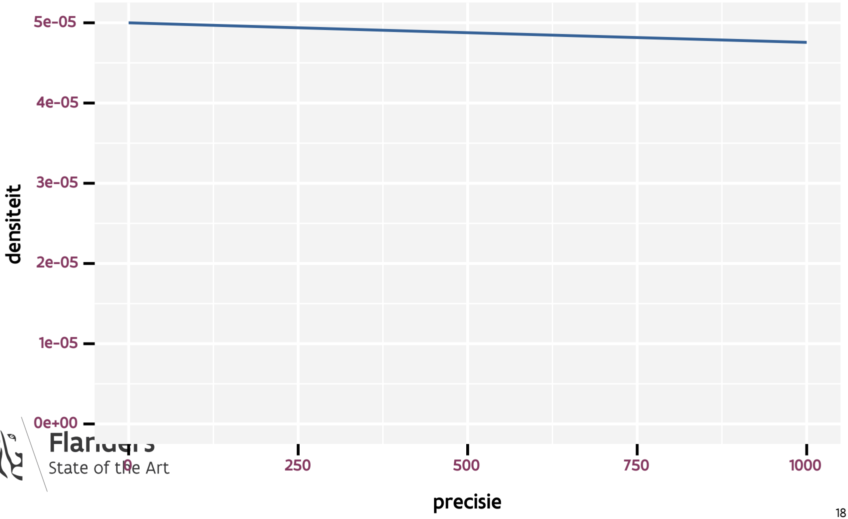
Random intercept model ('iid')

Challenge 3

1. fit a model with one or more random intercepts (`model = 'iid'`)
2. what is the default prior for `'iid'` (`inla.doc('iid')`)
3. calculate σ for the random intercept
4. display the BLUP the random intercept

Default prior

$$\log \tau \sim logGamma(1, 0.00005)$$
$$\tau \sim Gamma(1, 0.00005)$$



σ random intercept

```r
read.delim("ButterfliesNEggs_V4.txt") %>%
  mutate(TreeHeight = TreeHeight / 100 - 1,
         Distance2Edge = Distance2Edge / 10 - 1,
         SmallOakAbundance = SmallOakAbundance / 10 - 0.2) -> butterfly
model <- inla(NEggs ~ NLowBranches + TreeHeight + SmallOakAbundance +
                f(Area, model = "iid"), family = "poisson", data = butterfly)
model$summary.hyperpar
```

```
##                        mean        sd 0.025quant 0.5quant 0.975quant      mode
## Precision for Area 1.154638 0.6190456  0.3574407 1.023218   2.714477 0.7995142
```

```r
to_sigma <- function(tau){sqrt(1/tau)}
model$marginals.hyperpar$`Precision for Area` %>%
  inla.tmarginal(fun = to_sigma) %>%
  inla.zmarginal()
```

```
## Mean              1.02743
## Stdev             0.272553
## Quantile  0.025 0.607468
## Quantile  0.25  0.83313
## Quantile  0.5   0.98818
## Quantile  0.75  1.17742
## Quantile  0.975 1.67119
```

Best Linear Unbiased Predictor (BLUP)

```
glimpse(model$summary.random$Area)
```

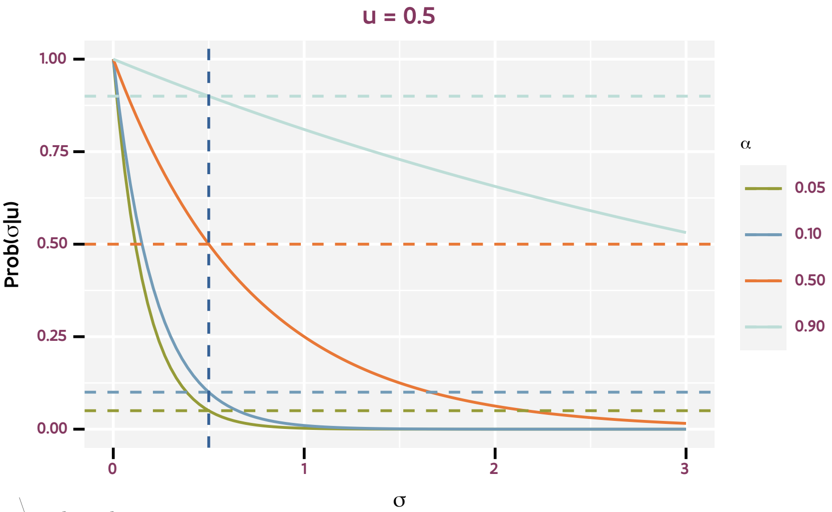
```
## Observations: 22
## Variables: 8
## $ ID           <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,...
## $ mean         <dbl> -0.70876380, 0.89228904, -0.35082396, -0.67654342, -0....
## $ sd           <dbl> 0.5554291, 0.4237639, 0.6676452, 0.8143332, 0.8270883,...
## $ `0.025quant` <dbl> -1.87289647, 0.10071051, -1.76737211, -2.48509876, -2....
## $ `0.5quant`   <dbl> -0.68655883, 0.87658241, -0.31935080, -0.61132373, -0....
## $ `0.975quant` <dbl> 0.3289457, 1.7741420, 0.8840476, 0.7496895, 0.8490009,...
## $ mode         <dbl> -0.64488601, 0.84680983, -0.26408689, -0.50262798, -0....
## $ kld          <dbl> 4.131327e-05, 1.788192e-04, 3.417066e-05, 9.033027e-05...
```

Challenge 4

1. Think about the relevant magnitude of σ for your random effect
2. Use a custom “pc.prec” prior with that σ (`inla.doc("pc.prec")`)

Penalised Complexity prior

$Prob(\sigma > u) = \alpha$ met $u > 0$ en $0 < \alpha < 1$



inlatools package

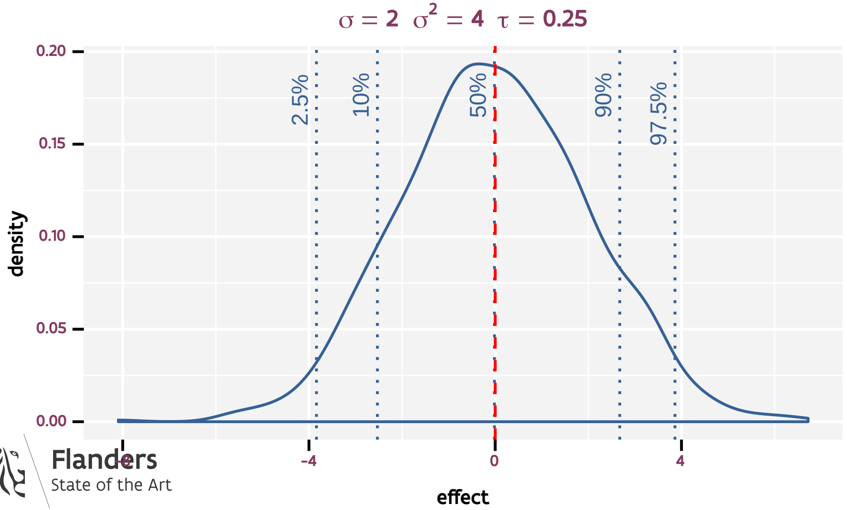
https://inlatools.netlify.com

```
remotes::install_github("inbo/inlatools")
```

- assessing σ random intercept
- assessing σ random walk
- check dispersion
- check distribution
- extract fitted values, observed values, Pearson residuals

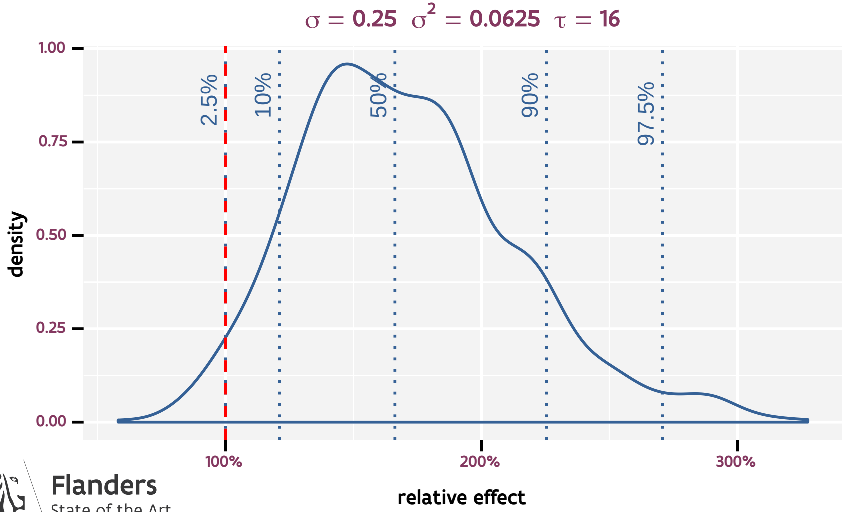
Assessing σ random intercept

```r
library(inlatools)
plot(simulate_iid(sigma = 2))
```



Assessing σ random intercept with `link` and `center`

```
plot(simulate_iid(tau = 16), link = "log", center = "bottom")
```



Solution 4

```r
model_pc <- inla(NEggs ~ NLowBranches + TreeHeight + SmallOakAbundance +
                  f(Area, model = "iid",
                    hyper = list(
                      theta = list(prior = "pc.prec", param = c(0.1, 0.05)))),
                 family = "poisson", data = butterfly)
model_pc$marginals.hyperpar$`Precision for Area` %>%
  inla.tmarginal(fun = to_sigma) %>%
  inla.zmarginal()
```

```
## Mean              0.491791
## Stdev             0.0881641
## Quantile  0.025 0.335646
## Quantile  0.25  0.429468
## Quantile  0.5   0.48571
## Quantile  0.75  0.547405
## Quantile  0.975 0.681398
```

Check dispersion

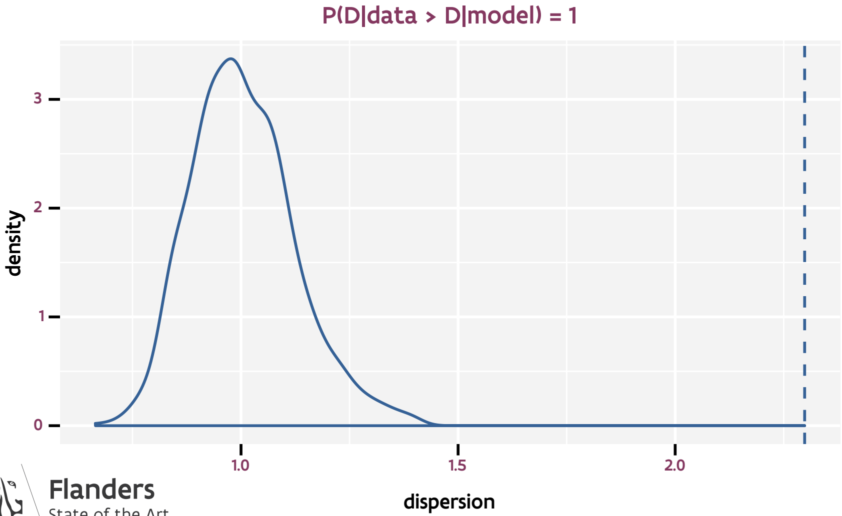
```r
plot(dispersion_check(model_pc))
```

```
## Error in fitted(object): no fitted values in object.
## Refit the object with 'control.predictor = list(compute = TRUE)'
```

```r
model_pc <- inla(NEggs ~ NLowBranches + TreeHeight + SmallOakAbundance +
                   f(Area, model = "iid",
                     hyper = list(
                       theta = list(prior = "pc.prec", param = c(0.1, 0.05)))),
                 family = "poisson", data = butterfly,
                 control.predictor = list(compute = TRUE))
```

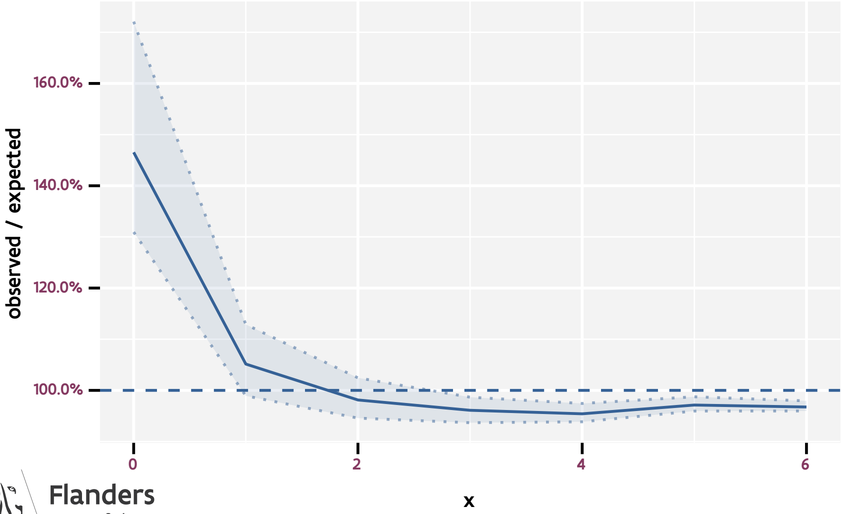
Clear overdispersion

```
plot(dispersion_check(model_pc))
```



Excess of zeros

```
plot(fast_distribution_check(model_pc))
```



Linear combinations including random intercept

Create a matrix for each random effect

- 1 row per linear combination
- 1 column per random effect level

```
Area <- matrix(0, nrow = 3, ncol = 22)
Area[1, 5] <- 1
Area[2, c(3, 4)] <- c(1, -1)
Area[3, ] <- 1 / 22
lc1 <- inla.make.lincombs(NLowBranches = c(2, 0, 1), Area = Area)
names(lc1) <- c("Area 5", "Area 3 - Area 4", "average area")
lc2 <- inla.make.lincombs(NLowBranches = 1)
names(lc2) <- "fixed only"
lc <- c(lc1, lc2)
str(lc, 1)
```

```
## List of 4
##  $ Area 5         :List of 2
##  $ Area 3 - Area 4:List of 2
##  $ average area   :List of 2
##  $ fixed only     :List of 1
```

Flanders
State of the Art

```r
model_lc <- inla(NEggs ~ NLowBranches + TreeHeight + SmallOakAbundance +
                  f(Area, model = "iid",
                    hyper = list(
                      theta = list(prior = "pc.prec", param = c(0.1, 0.05)))),
                 family = "poisson", data = butterfly, lincomb = lc)
model_lc$summary.lincomb.derived # estimate are always on the link scale!
```

```
##                   ID       mean         sd 0.025quant   0.5quant 0.975quant
## Area 5             1 0.75712122 0.47095877 -0.1984497 0.76622380  1.6625693
## Area 3 - Area 4    2 0.04482048 0.58720259 -1.1049607 0.04018034  1.2205331
## average area       3 0.54350370 0.13484111  0.2784505 0.54317773  0.8100311
## fixed only         4 0.52613809 0.08243795  0.3642870 0.52613415  0.6878707
##                         mode kld
## Area 5            0.78398220   0
## Area 3 - Area 4 0.03151894   0
## average area      0.54263242   0
## fixed only        0.52613313   0
```

Back transform to natural scale

```r
exp(model_lc$summary.lincomb.derived["average area", 4:6])
```

```
##              0.025quant 0.5quant 0.975quant
## average area   1.321081 1.721469   2.247978
```

```r
inla.tmarginal(exp, model_lc$marginals.lincomb.derived$`average area`) %>%
  inla.zmarginal()
```

```
## Mean              1.73754
## Stdev             0.234051
## Quantile  0.025 1.32279
## Quantile  0.25  1.57399
## Quantile  0.5   1.72106
## Quantile  0.75  1.88237
## Quantile  0.975 2.24489
```

RESEARCH INSTITUTE
NATURE AND FOREST

Flanders
State of
the Art

First order random walk model (‘rw1’)

Definition

$$\Delta x_i = x_i - x_{i-1} \sim \mathcal{N}(0, \sigma^2)$$

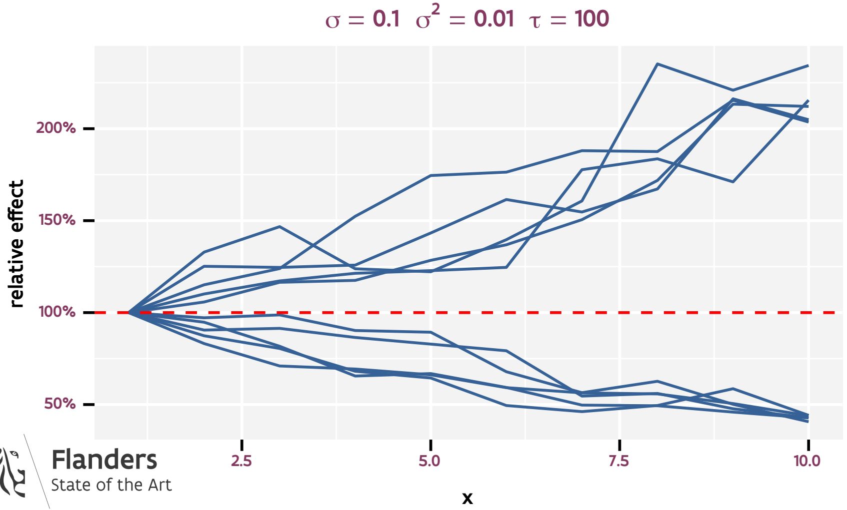
$$x_i \sim \mathcal{N}(x_{i-1}, \sigma^2)$$

```
inla.doc("rw1")
```

- Useful in case of non-linear patterns in discrete variables (year, day, ...)
- Works with discretised continuous variables (e.g. after rounding)

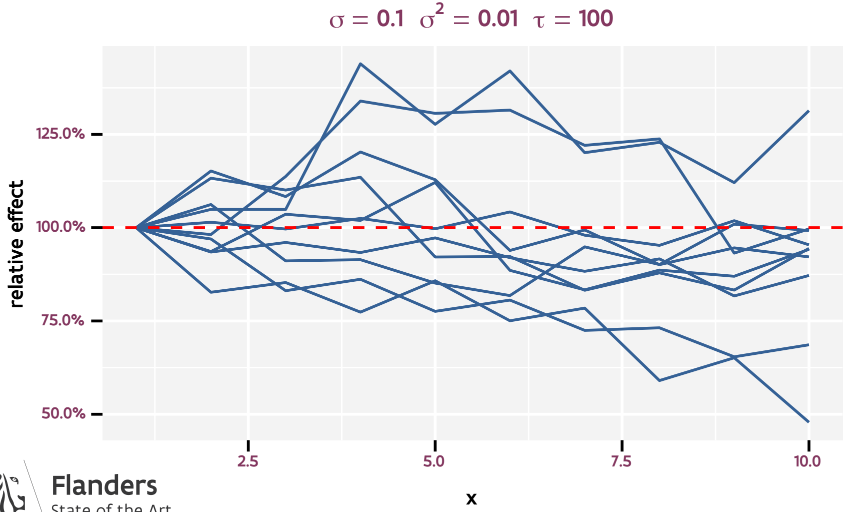
Divergent series

```
rw1 <- simulate_rw(sigma = 0.1)
plot(select_divergence(rw1), link = "log")
```



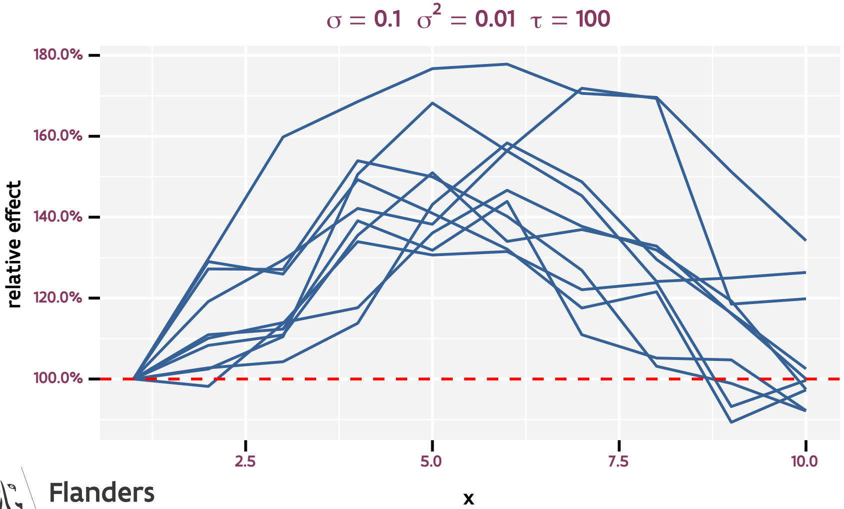
Series with frequency change in direction

```
plot(select_change(rw1), link = "log")
```



Series with central maximum

```
plot(select_poly(rw1, coefs = c(0, -1)), link = "log")
```



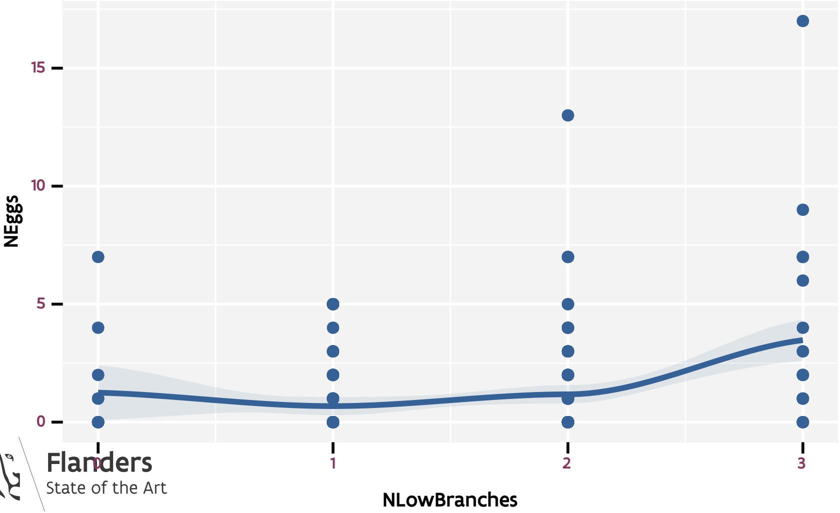
Flanders
State of the Art

Challenge 5

1. pick a relevant variable for an ‘rw1’ model
2. ponder on a relevant σ for that model
3. fit model with ‘rw1’ component and `pc.prec` prior

Data exploration NLowBranches

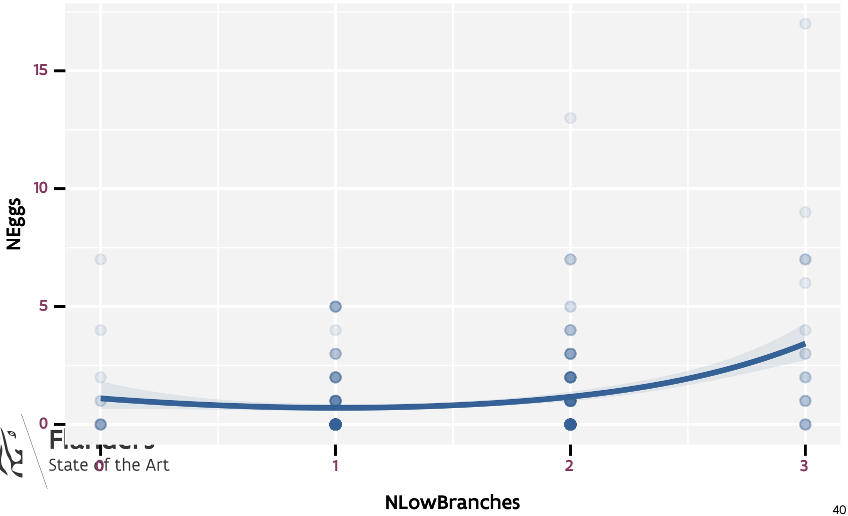
```r
ggplot(butterfly, aes(x = NLowBranches, y = NEggs)) +
  geom_smooth() + geom_point()
```



Flanders
State of the Art

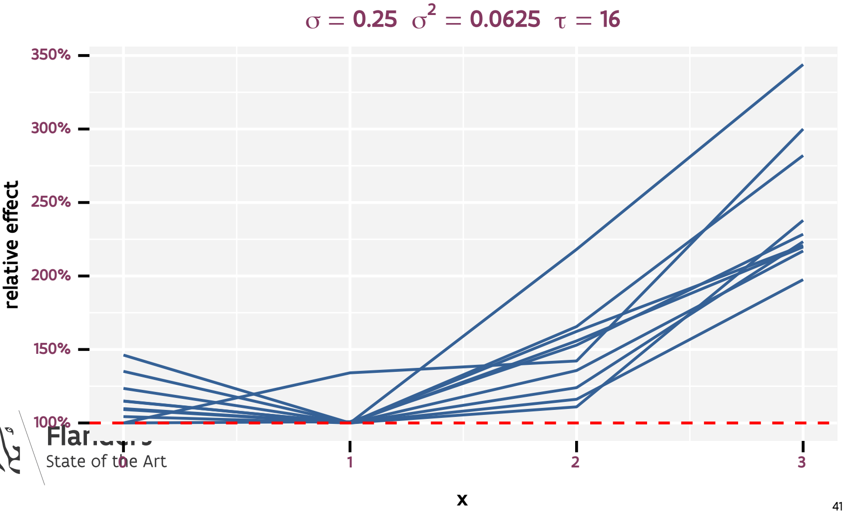
Data exploration NLowBranches

```
ggplot(butterfly, aes(x = NLowBranches, y = NEggs)) +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs", k = 4),
              method.args = list(family = poisson)) + geom_point(alpha = 0.1)
```



Relevant σ for prior NLowBranches

```
simulate_rw(sigma = 0.25, start = 0, length = 4) %>%
  select_poly(coef = c(1, 1)) %>%
  plot(link = "log", center = "bottom")
```



Solution 5

```
model_rw1 <- inla(NEggs ~ TreeHeight + SmallOakAbundance +
                    f(NLowBranches, model = "rw1",
                      hyper = list(
                        theta = list(prior = "pc.prec", param = c(0.25, 0.05)))) +
                    f(Area, model = "iid",
                      hyper = list(
                        theta = list(prior = "pc.prec", param = c(0.1, 0.05)))),
                  family = "poisson", data = butterfly,
                  control.predictor = list(compute = TRUE))
```

Non-linear pattern NLowBranches