# The **n**ew **F**orest **I**nventory **Est**imation and **A**nalysis system

Jiří Fejfar[1],
Radim Adolt[1], Ivo Kohn[1], Adrian Lanz[2]

[1]Forest Management Institute Brandýs nad Labem (ÚHÚL)
[2]Federal Institute for Forest, Snow and Landscape Research WSL

# Outline

1 ) Forest inventory SW

- Existing SW for forest inventory
- What is nFIESTA novelty?
- System overview

2) Implementation

- PostgreSQL extension: with version + tests + CI/CD
- Matrix manipulation

3) Case studies

# Existing SW

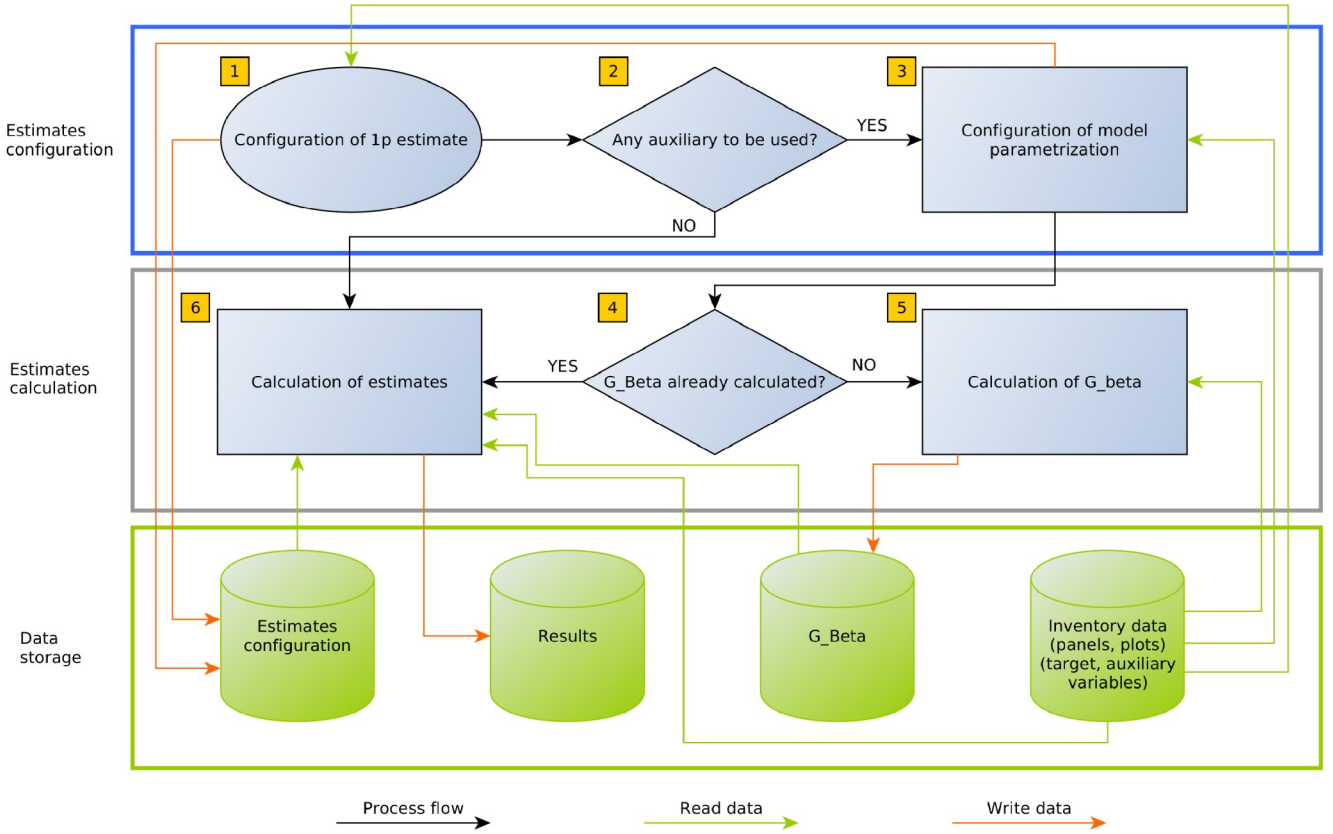
- Ye, R, 1997 Fiesta – A Software tor Forest Inventory (SAS Version). Technical Report, Chair of forest Inventory and Planning, Swiss Federal Institute of Technology (ETH) Zurich. 23 S.

  (eth-24036-02.pdf)
- FIESTA – U.S. Forest Service, 2012.
- Forestinventory package – Andreas Hill, 2017.
- ENFIN e-FOREST (web application + PostgreSQL)

# What is nFIESTA novelty?

- The continuous Horvitz-Thompson theorem:
  - See nfiesta_methods.pdf
  - Unequal inclusion densities
  - Estimation of total without “known forest map”
- DB (PostgreSQL) based

# System overview




- More on nfiesta_overview.pdf

# 2 ) SW implementation

- Software implementation
  - System features (what IS implemented and NOT implemented)
  - DB structure – ERD
  - Functions
  - Matrix representation & SQL implementation of estimators
  - License, Installation

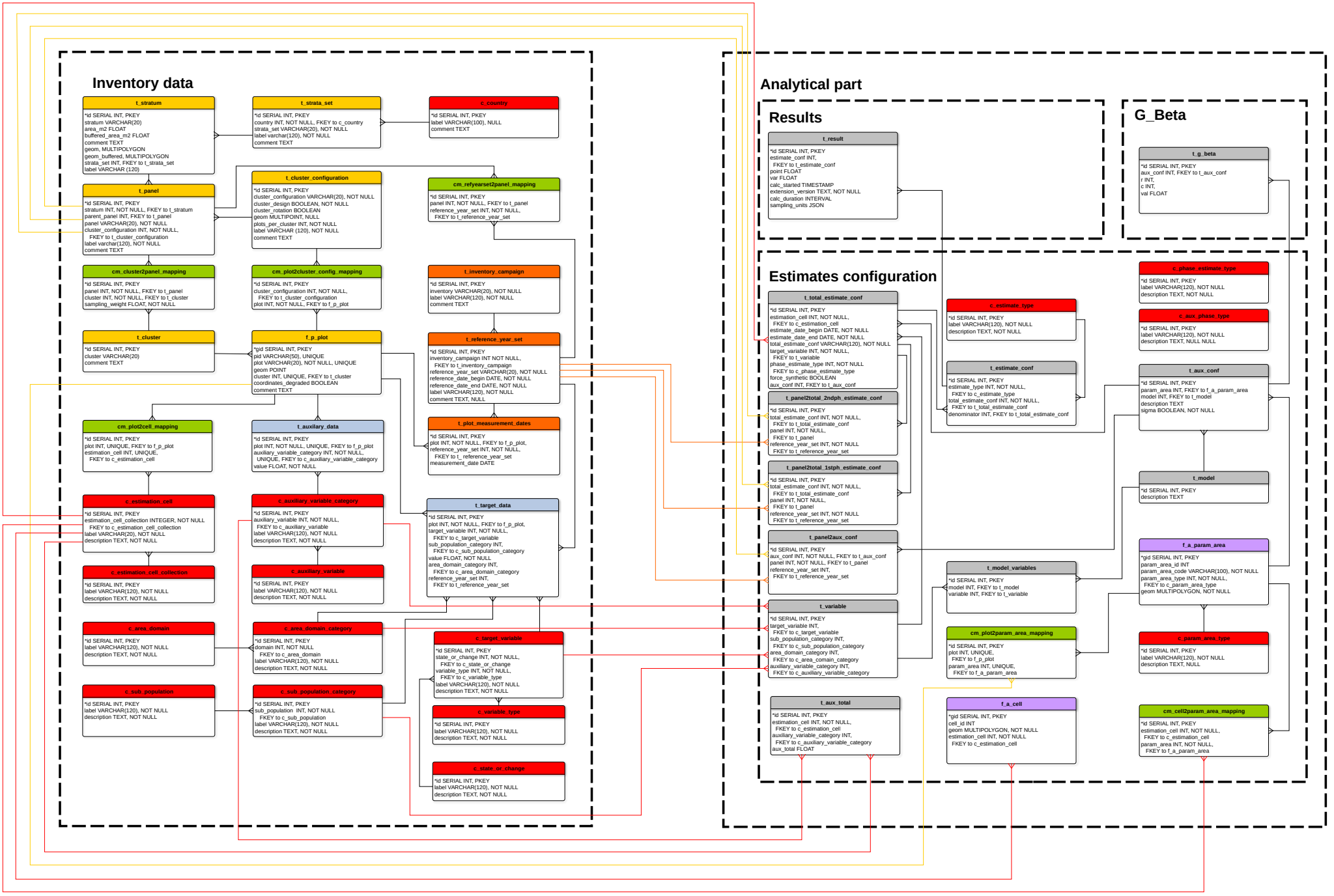
# System features – implemented

- DB (PotgreSQL) structures (**47 tables, 17 functions)** that all together form an NFI estimation device.
- Management of **data** (**field** as well as **auxiliary**) necessary for one-phase and regression estimators of total.
- Configuration of various types of (linear) regression estimators which are specified by different sets of predictors, areas of interest (cells) and possibly larger parameterization areas (in which the model parameters are estimated).
- Processing of **GIS data** (plot positions, geometries of areas of interest, sampling strata, or parametrization areas). But **spatial data is not mandatory** provided all necessary information is available as simple attribute data – e.g. linkage of plots and cells (estimation domains), auxiliary totals.

# System features – not implemented

- **Upload mechanism**. In particular, it does not replace the e-Forest set of tools.
- Generic ETL (**Extract Transform and Load**) mechanism has to be used to get data into the system – from various sources e.g. the "old" e-Forest, or any database / files.
- There are **no plans to implement** any functionality for the end-users to upload their data.
- Any existing estimation (SQL) functionality of e-Forest could be relatively easily incorporated.
- Documentation

# ERD


Inventory data
t_stratum
*id SERIAL INT, PKEY
stratum VARCHAR(20)
area_m2 FLOAT
buffered_area_m2 FLOAT
comment TEXT
geom, MULTIPOLYGON
geom_buffered, MULTIPOLYGON
strata_set INT, FKEY to t_strata_set
label VARCHAR (120)
t_strata_set
*id SERIAL INT, PKEY
country INT, NOT NULL, FKEY to c_country
strata_set VARCHAR(20), NOT NULL
label varchar(120), NOT NULL
comment TEXT
c_country
*id SERIAL INT, PKEY
label VARCHAR(100), NULL
comment TEXT
t_panel
*id SERIAL INT, PKEY
stratum INT, NOT NULL, FKEY to t_stratum
parent_panel INT, FKEY to t_panel
panel VARCHAR(20), NOT NULL
cluster_configuration INT, NOT NULL,
FKEY to t_cluster_configuration
label varchar(120), NOT NULL
comment TEXT
t_cluster_configuration
*id SERIAL INT, PKEY
cluster_configuration VARCHAR(20), NOT NULL
cluster_design BOOLEAN, NOT NULL
cluster_rotation BOOLEAN
geom MULTIPOINT, NULL
plots_per_cluster INT, NOT NULL
label VARCHAR (120), NOT NULL
comment TEXT
cm_refyearset2panel_mapping
*id SERIAL INT, PKEY
panel INT, NOT NULL, FKEY to t_panel
reference_year_set INT, NOT NULL,
FKEY to t_reference_year_set
cm_cluster2panel_mapping
*id SERIAL INT, PKEY
panel INT, NOT NULL, FKEY to t_panel
cluster INT, NOT NULL, FKEY to t_cluster
sampling_weight FLOAT, NOT NULL
cm_plot2cluster_config_mapping
*id SERIAL INT, PKEY
cluster_configuration INT, NOT NULL,
FKEY to t_cluster_configuration
plot INT, NOT NULL, FKEY to f_p_plot
t_inventory_campaign
*id SERIAL INT, PKEY
inventory VARCHAR(20), NOT NULL
label VARCHAR(120), NOT NULL
comment TEXT
t_cluster
*id SERIAL INT, PKEY
cluster VARCHAR(20)
comment TEXT
f_p_plot
*gid SERIAL INT, PKEY
pid VARCHAR(50), UNIQUE
plot VARCHAR(20), NOT NULL, UNIQUE
geom POINT
cluster INT, UNIQUE, FKEY to t_cluster
coordinates_degraded BOOLEAN
comment TEXT
t_reference_year_set
*id SERIAL INT, PKEY
inventory_campaign INT NOT NULL,
FKEY to t_inventory_campaign
reference_year_set VARCHAR(20), NOT NULL
reference_date_begin DATE, NOT NULL
reference_date_end DATE, NOT NULL
label VARCHAR(120), NOT NULL
comment TEXT, NULL
cm_plot2cell_mapping
*id SERIAL INT, PKEY
plot INT, UNIQUE, FKEY to f_p_plot
estimation_cell INT, UNIQUE,
FKEY to c_estimation_cell
t_auxiliary_data
*id SERIAL INT, PKEY
plot INT, NOT NULL, UNIQUE, FKEY to f_p_plot
auxiliary_variable_category INT, NOT NULL,
UNIQUE, FKEY to c_auxiliary_variable_category
value FLOAT, NOT NULL
t_plot_measurement_dates
*id SERIAL INT, PKEY
plot INT, NOT NULL, FKEY to f_p_plot,
reference_year_set INT, NOT NULL,
FKEY to t_reference_year_set
measurement_date DATE
c_estimation_cell
*id SERIAL INT, PKEY
estimation_cell_collection INTEGER, NOT NULL
FKEY to c_estimation_cell_collection
label VARCHAR(20), NOT NULL
description TEXT, NOT NULL
c_auxiliary_variable_category
*id SERIAL INT, PKEY
auxiliary_variable INT, NOT NULL,
FKEY to c_auxiliary_variable
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
t_target_data
*id SERIAL INT, PKEY
plot INT, NOT NULL, FKEY to f_p_plot,
target_variable INT, NOT NULL,
FKEY to c_target_variable
sub_population_category INT,
FKEY to c_sub_population_category
value FLOAT, NOT NULL
area_domain_category INT,
FKEY to c_area_domain_category
reference_year_set INT,
FKEY to t_reference_year_set
c_estimation_cell_collection
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_auxiliary_variable
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_area_domain
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_area_domain_category
*id SERIAL INT, PKEY
domain INT, NOT NULL
FKEY to c_area_domain
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_target_variable
*id SERIAL INT, PKEY
state_or_change INT, NOT NULL,
FKEY to c_state_or_change
variable_type INT, NOT NULL,
FKEY to c_variable_type
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_sub_population
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_sub_population_category
*id SERIAL INT, PKEY
sub_population INT, NOT NULL
FKEY to c_sub_population
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_variable_type
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_state_or_change
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
Analytical part
Results
t_result
*id SERIAL INT, PKEY
estimate_conf INT,
FKEY to t_estimate_conf
point FLOAT
var FLOAT
calc_started TIMESTAMP
extension_version TEXT, NOT NULL
calc_duration INTERVAL
sampling_units JSON
G_Beta
t_g_beta
*id SERIAL INT, PKEY
aux_conf INT, FKEY to t_aux_conf
r INT,
c INT,
val FLOAT
Estimates configuration
t_total_estimate_conf
*id SERIAL INT, PKEY
estimation_cell INT, NOT NULL,
FKEY to c_estimation_cell
estimate_date_begin DATE, NOT NULL
estimate_date_end DATE, NOT NULL
total_estimate_conf VARCHAR(120), NOT NULL
target_variable INT, NOT NULL,
FKEY to t_variable
phase_estimate_type INT, NOT NULL
FKEY to c_phase_estimate_type
force_synthetic BOOLEAN
aux_conf INT, FKEY to t_aux_conf
c_estimate_type
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
t_estimate_conf
*id SERIAL INT, PKEY
estimate_type INT, NOT NULL,
FKEY to c_estimate_type
total_estimate_conf INT, NOT NULL,
FKEY to t_total_estimate_conf
denominator INT, FKEY to t_total_estimate_conf
c_phase_estimate_type
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NOT NULL
c_aux_phase_type
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NULL NULL
t_aux_conf
*id SERIAL INT, PKEY
param_area INT, FKEY to f_a_param_area
model INT, FKEY to t_model
description TEXT
sigma BOOLEAN, NOT NULL
t_panel2total_2ndph_estimate_conf
*id SERIAL INT, PKEY
total_estimate_conf INT, NOT NULL,
FKEY to t_total_estimate_conf
panel INT, NOT NULL,
FKEY to t_panel
reference_year_set INT, NOT NULL
FKEY to t_reference_year_set
t_model
*id SERIAL INT, PKEY
description TEXT
t_panel2total_1stph_estimate_conf
*id SERIAL INT, PKEY
total_estimate_conf INT, NOT NULL,
FKEY to t_total_estimate_conf
panel INT, NOT NULL,
FKEY to t_panel
reference_year_set INT, NOT NULL
FKEY to t_reference_year_set
f_a_param_area
*gid SERIAL INT, PKEY
param_area_id INT
param_area_code VARCHAR(100), NOT NULL
param_area_type INT, NOT NULL,
FKEY to c_param_area_type
geom MULTIPOLYGON, NOT NULL
t_panel2aux_conf
*id SERIAL INT, PKEY
aux_conf INT, NOT NULL, FKEY to t_aux_conf
panel INT, NOT NULL, FKEY to t_panel
reference_year_set INT,
FKEY to t_reference_year_set
t_model_variables
*id SERIAL INT, PKEY
model INT, FKEY to t_model
variable INT, FKEY to t_variable
t_variable
*id SERIAL INT, PKEY
target_variable INT,
FKEY to c_target_variable
sub_population_category INT,
FKEY to c_sub_population_category
area_domain_category INT,
FKEY to c_area_domain_category
auxiliary_variable_category INT,
FKEY to c_auxiliary_variable_category
cm_plot2param_area_mapping
*id SERIAL INT, PKEY
plot INT, UNIQUE,
FKEY to f_p_plot
param_area INT, UNIQUE,
FKEY to f_a_param_area
c_param_area_type
*id SERIAL INT, PKEY
label VARCHAR(120), NOT NULL
description TEXT, NULL
t_aux_total
*id SERIAL INT, PKEY
estimation_cell INT, NOT NULL,
FKEY to c_estimation_cell
auxiliary_variable_category INT,
FKEY to c_auxiliary_variable_category
aux_total FLOAT
f_a_cell
*gid SERIAL INT, PKEY
cell_id INT
geom MULTIPOLYGON, NOT NULL
estimation_cell INT, NOT NULL
FKEY to c_estimation_cell
cm_cell2param_area_mapping
*id SERIAL INT, PKEY
estimation_cell INT, NOT NULL,
FKEY to c_estimation_cell
param_area INT, NOT NULL,
FKEY to f_a_param_area

# Functions

- Configuration
- Estimation
- Run in parallel with python “hack” client

# Inverse of matrix, using R

```
CREATE OR REPLACE FUNCTION
analytical.fn_inverse(input double precision[][])
  RETURNS double precision[][] AS
$BODY$
      return(solve(input))
$BODY$
  LANGUAGE plr VOLATILE STRICT
  COST 100;
```

p x n  n x n  n x p

$$\widetilde{\mathbf{G}}_{\beta_{\mathbf{t}+}} = [\mathbf{X}_{\mathbf{tD}_+}\mathbf{\Pi}_+\mathbf{X}'_{\mathbf{tD}_+}]^{-1}\mathbf{X}_{\mathbf{tD}_+}$$

```
SELECT analytical.fn_inverse (
            '{{222477.383519094,7489538.83305858},
            {7489538.83305858,537848864.980823}}'
      );
```

```
SELECT analytical.fn_inverse (
            array_agg(
                  array_agg(...))
     );
```

# Functions, using R

```
CREATE OR REPLACE FUNCTION
analytical.fn_samplingVarEst__Est_Total_NHT__VE_HT_Total_NHT(
    ldsitys double precision[],
    pixs double precision[],
    pixys double precision[][])
  RETURNS double precision[] AS
$BODY$
        library(samplingVarEst)

        pointEst <- Est.Total.NHT(ldsitys[1==1], pixs[1==1])
        varEst <- VE.HT.Total.NHT(ldsitys[1==1], pixs[1==1], pixys)

        result_df<-c(pointEst, varEst)

return(result_df)
$BODY$
  LANGUAGE plr IMMUTABLE
  COST 100;
```

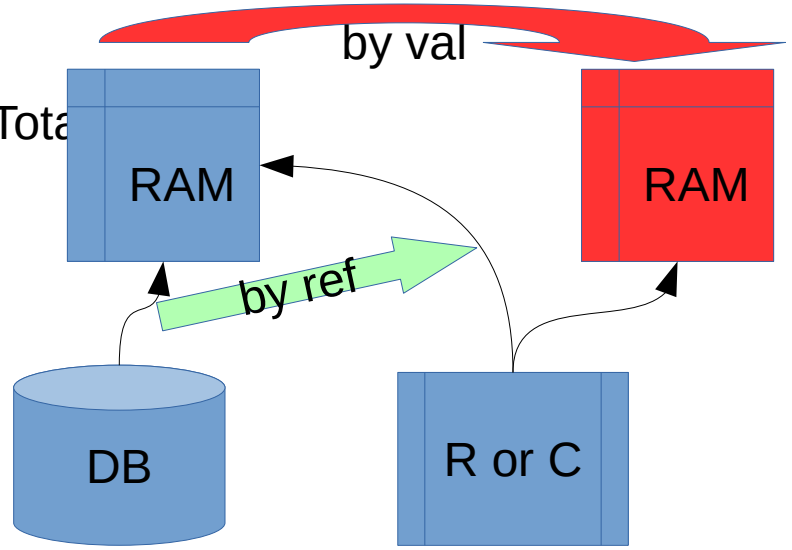
# Functions, using R

```
CREATE OR REPLACE FUNCTION
analytical.fn_samplingVarEst__Est_Total_NHT__VE_HT_Tota
    ldsitys double precision[],
    pixs double precision[],
    pixys double precision[][])
  RETURNS double precision[] AS
$BODY$
        library(samplingVarEst)

        pointEst <- Est.Total.NHT(ldsitys[1==1], pixs[1==1])
        varEst <- VE.HT.Total.NHT(ldsitys[1==1], pixs[1==1], pixys)

        result_df<-c(pointEst, varEst)

return(result_df)
...
```




| n | n^2 | 8 bytes (float) | GB |
|---|---|---|---|
| 40000 | 1 600 000 000 | 12 800 000 000 | 11.92 |

$$\hat{\mathbb{V}}_{HTC}(\hat{Y}_D) = \sum_{i=1}^{k} \left\{ \sum_{\substack{x \in s \\ x \in D_i}} \left[ \frac{Y(x)}{\pi_{S_i}(x)} \right]^2 + \sum_{\substack{x \in s \\ x \in D_i}} \sum_{\substack{x' \in s \\ x' \neq x \\ x' \in D_i}} Y(x) Y(x') \left[ \frac{\pi_S(x,x') - \pi_{S_i}(x)\pi_{S_i}(x')}{\pi_S(x,x')\pi_{S_i}(x)\pi_{S_i}(x')} \right] \right\}$$

# Matrix representation in SQL

- No PostgreSQL type for matrix (PostGIS raster)
- store large matrix on one row is not suitable

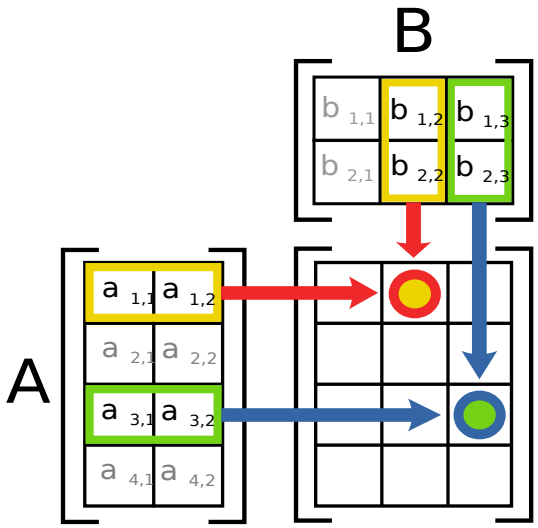
| | 1 | 2 |
|---|---|---|
| 1 | 3.14 | 6.28 |
| 2 | 6.28 | 12.57 |

| r | c | val |
|---|---|---|
| 1 | 1 | 3.14 |
| 1 | 2 | 6.28 |
| 2 | 1 | 6.28 |
| 2 | 2 | 12.57 |

- Sparsity, multidimensionality
- Column-oriented DB (compression, OLAP)
- Apache MAD lib

# SQL implementation

- Matrix inversion (see slide 11)
- Transpose
- Matrix multiplication




By File:Matrix multiplication diagram.svg:User:Bilou
See below. - This file was derived from:
Matrix multiplication diagram.svg, CC BY-SA 3.0, Link

```
WITH w_X_PI AS (
    SELECT
        … AS r,
        … AS c,
        … AS val
    FROM …
), w_XT AS (
    SELECT
        c AS r,
        r AS c,
        Val
    FROM w_X
)
SELECT
    A.r,
    B.c,
    sum(A.val * B.val) as val
FROM
    w_X_PI as A, w_XT as B
WHERE
    A.c = B.r
GROUP BY
    A.r, B.c
ORDER BY r, c
```

# Parallelization with “python hack” (excerpt)

```
…
import psycopg2
import psycopg2.extras
from multiprocessing import Pool

def single_query(_sql, verbose=True):
    ...
    # create connection to DB with psycopg2
    conn = psycopg2.connect(connstr)
    # run single query
    cur.execute(cmd)
    …
    return path_row_list

def run_parallel(_sql, _processes, _verbose=True):
    ...
    with Pool(_processes) as p:
        res_multi = p.map(single_query, _qrs, chunksize=1)
    print(res_multi)
```

# Installation, licence

- Installation

```
git clone https://gitlab.com/nfiesta/nfiesta_pg.git
cd nfiesta/nfiesta_pg
sudo make install && sudo make installcheck

create extension if not exists plr;
create extension nfiesta;
```

- License?
  - GNU GPL
  - European Union Public Licence

# Reference implementation in R

- R package with documentation
- https://gitlab.com/nfiesta/nfiesta_pg/tree/master/r_test

# Development

- PostgreSQL extension
  - Versioning
  - Regression tests
  - Automation: make & make install
  - Standardization: like PostGIS, plR
- virtualization:
  - Docker, (vagrant)
- https://gitlab.com/nfiesta/nfiesta_devenv

# CI / CD

- https://gitlab.com/nfiesta/nfiesta_pg/blob/master/.gitlab-ci.yml
- https://gitlab.com/nfiesta/nfiesta_pg/pipelines

# Next steps

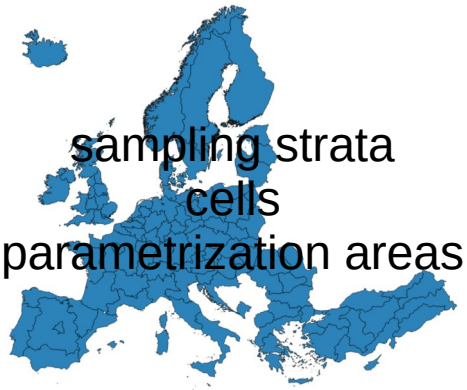
- Custom data type for matrix
  - https://gitlab.com/jurafejfar/pg_matrix
- API to Eigen?
- Can have custom data storage for LARGE matrices
  - FDW
  - Native PostgreSQL (in future)

# 3) Case studies

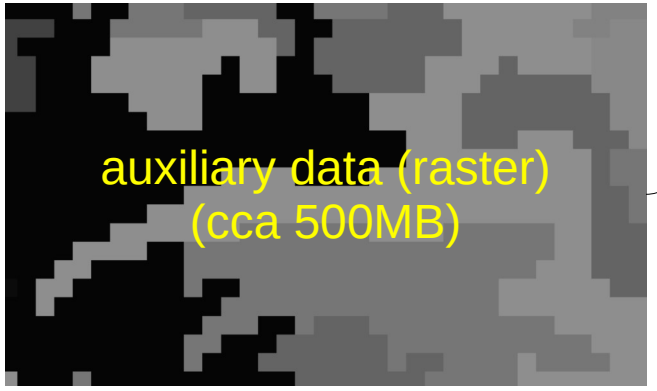
- Testing data (extract from NFI CZ)
- Micro case study (synthetic data)
- Case study (NFI data FR, DE, CH, CZ)

# **Micro**-Case-Study

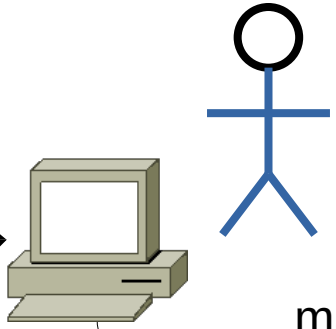
NFI plots & tracts
(1 380 055)



sampling strata
cells
parametrization areas



auxiliary data (raster)
(cca 500MB)



manual
preparation

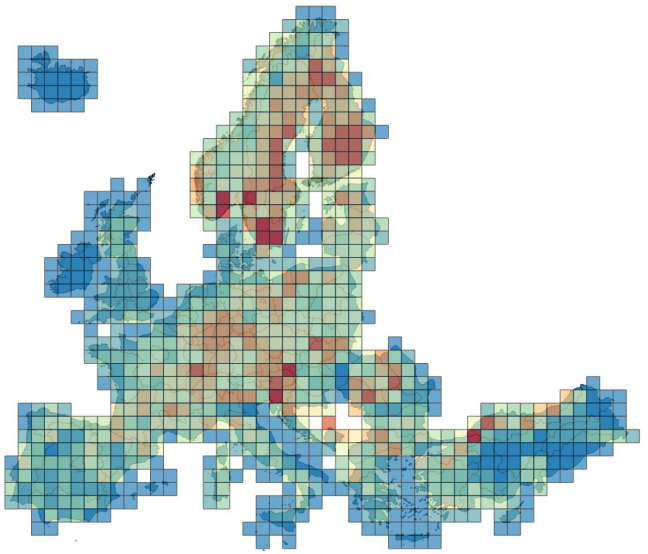
DB + extensions

ETL

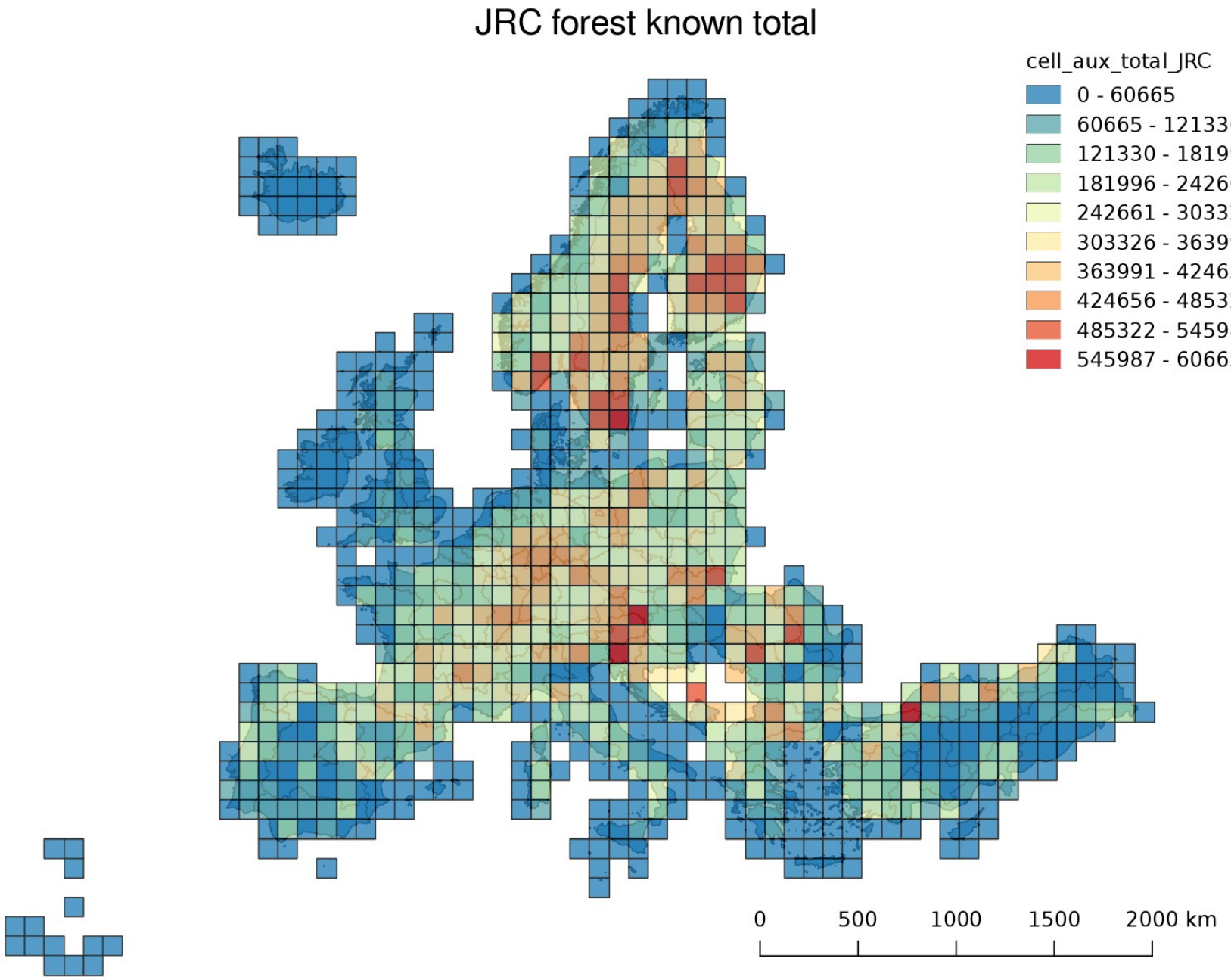
gisdata

analytical

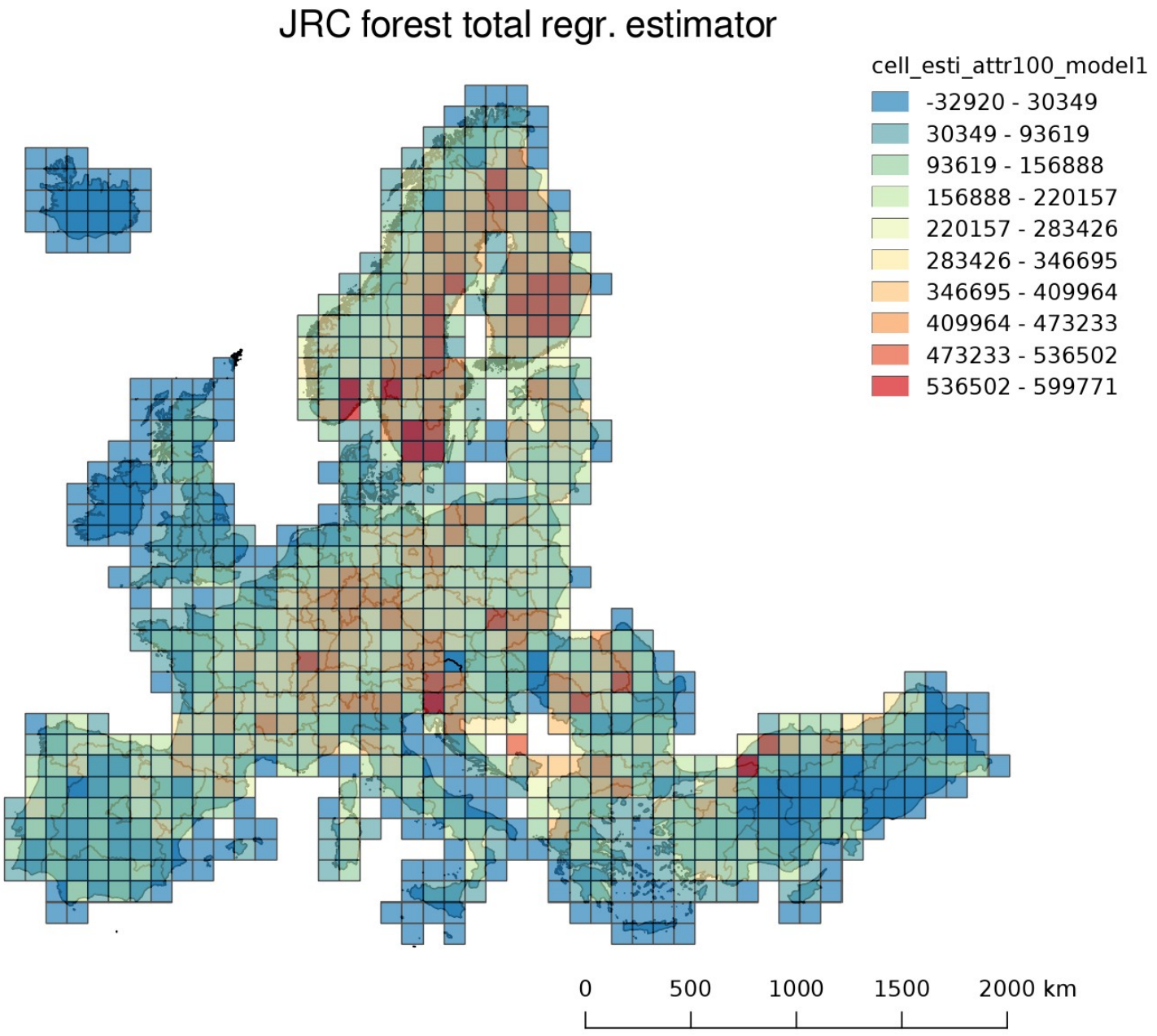
estimates



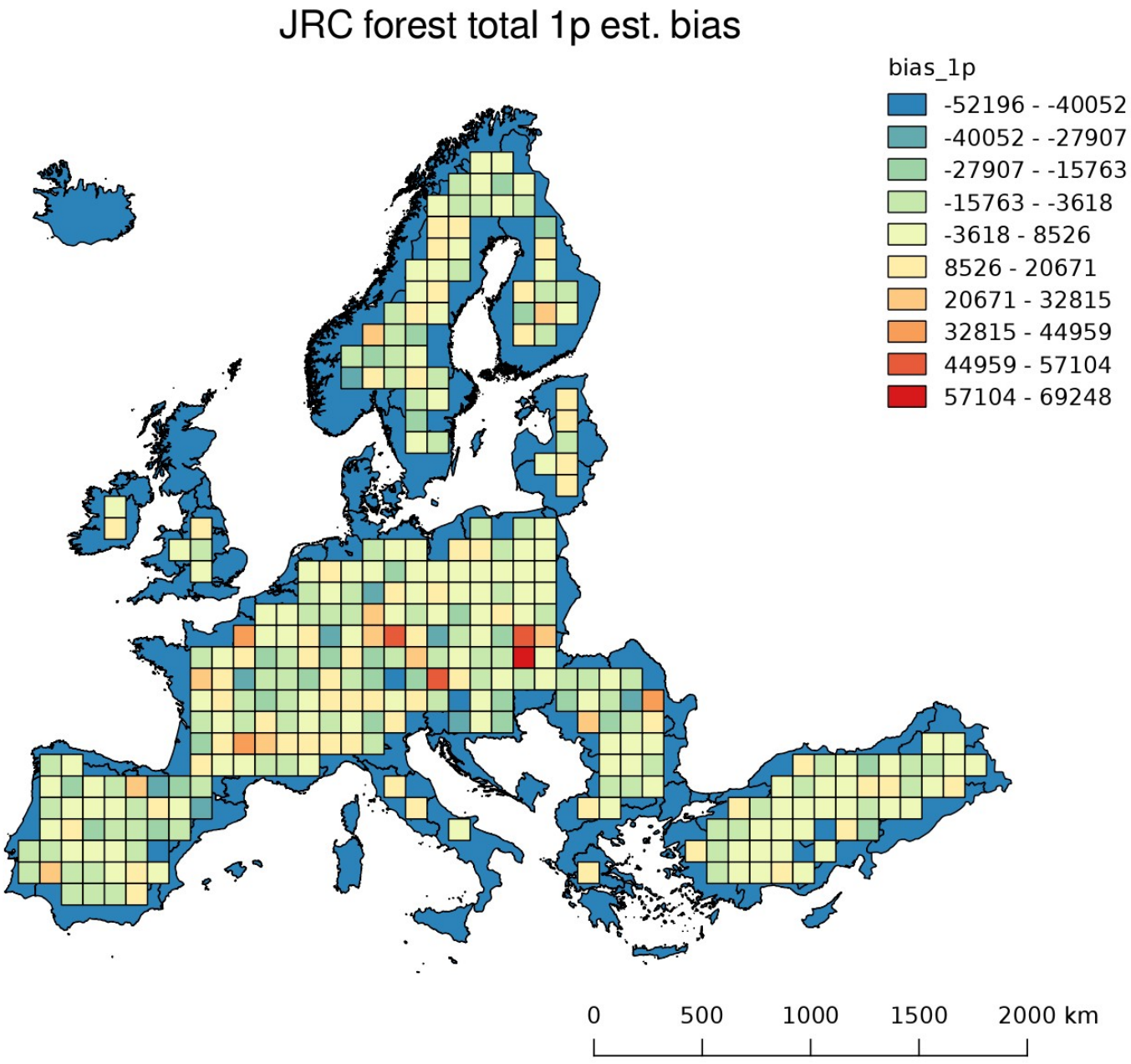
# **Micro-**Case-Study results



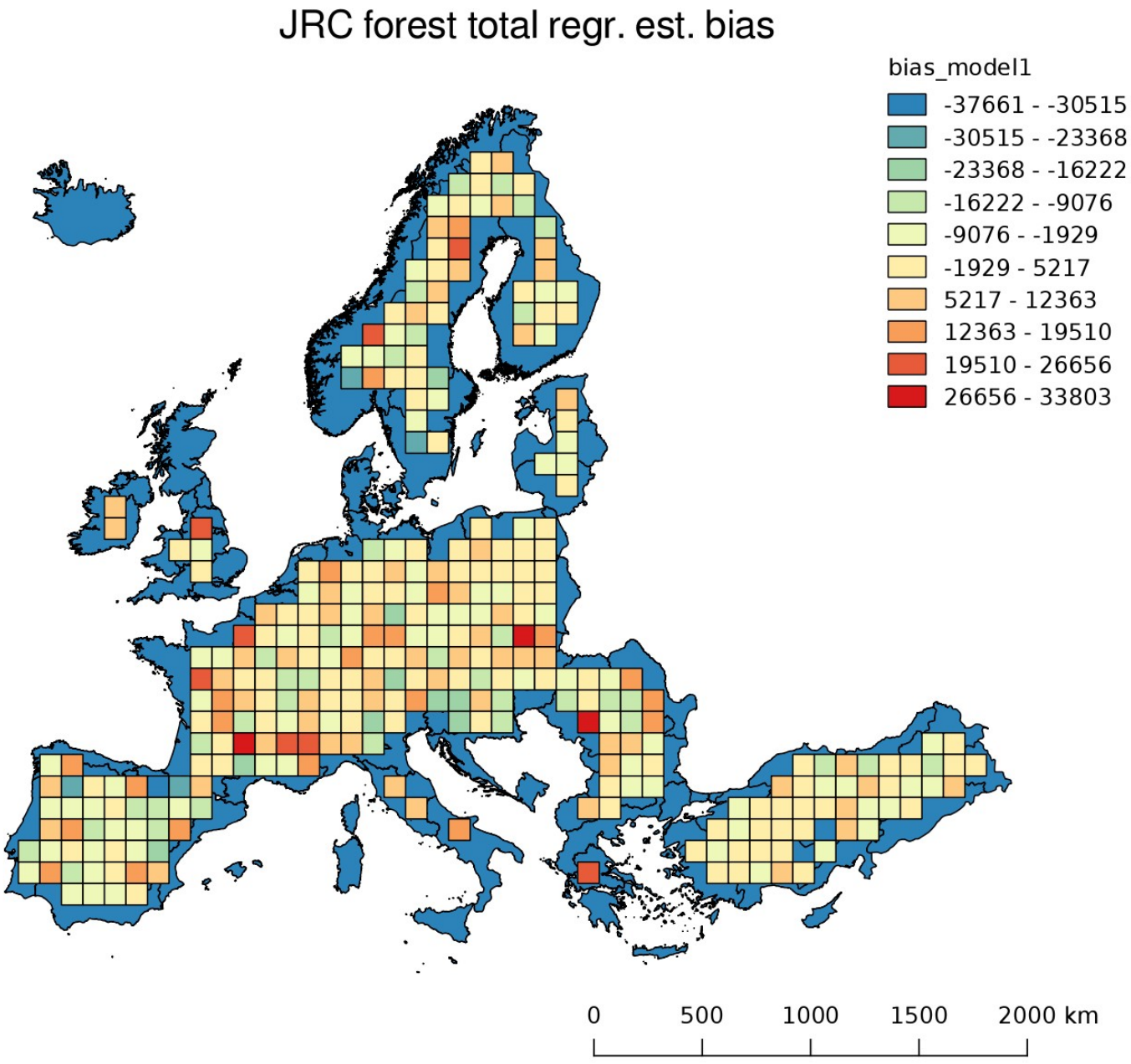
# **Micro-**Case-Study results



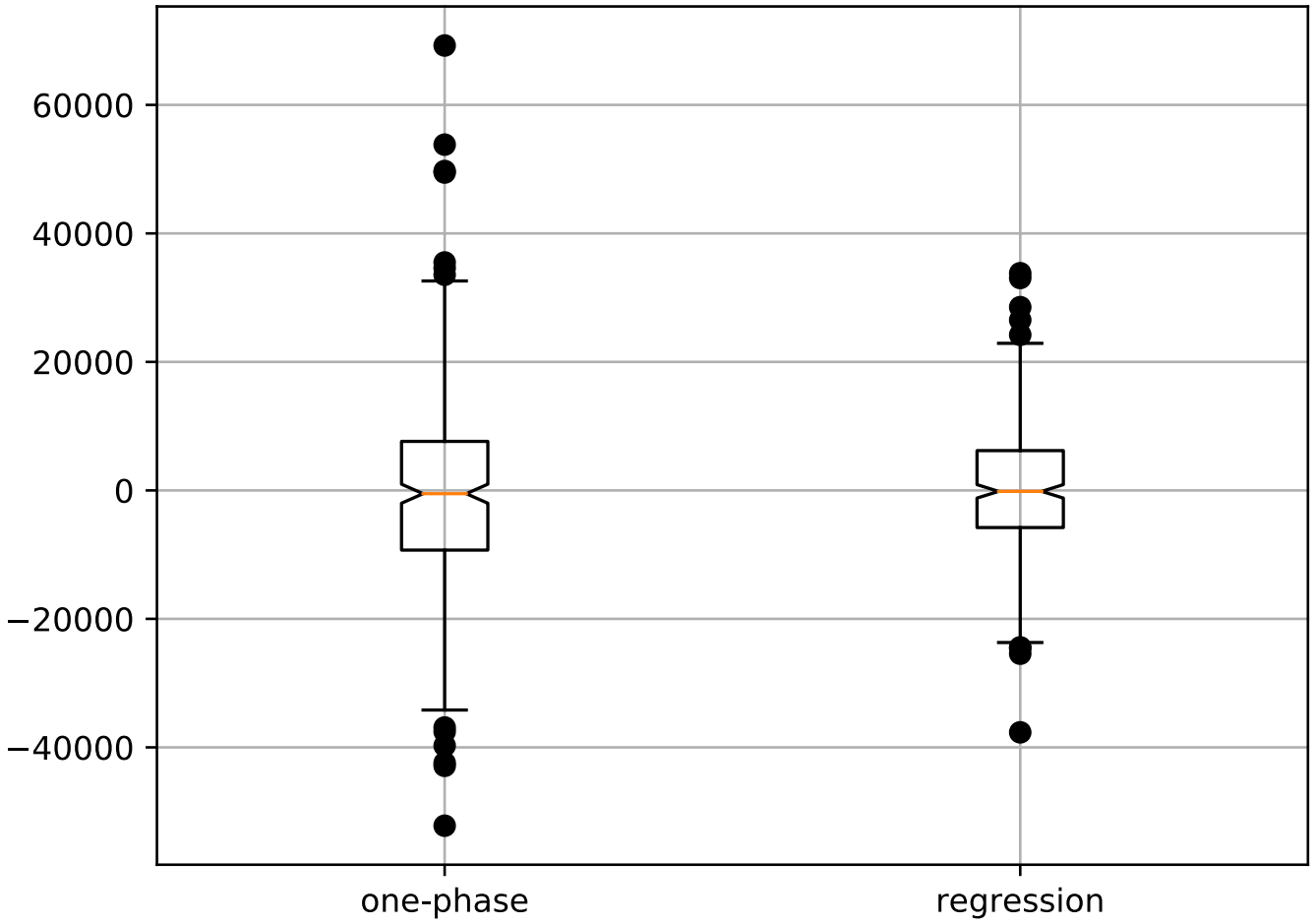
# **Micro-**Case-Study results



# **Micro-**Case-Study results



# **Micro-**Case-Study results



# Case study

- Real NFI data from France, Germany, Switzerland, Czech republic
- See 220-lanz-adolt-v005.pdf

# Thank you!

- https://gitlab.com/nfiesta