THE USE OF R
IN OFFICIAL STATISTICS

12th INTERNATIONAL CONFERENCE

Statistical disclosure control for Census data with sdcMicro package: the computational challenges

Vasiliki Spiliopoulou (Hellenic Statistical Authority)

Dionysios Fragkopoulos (Hellenic Statistical Authority)

HELLENIC STATISTICAL AUTHORITY

Outline

- Why Statistical Disclosure Control is necessary?
- Census data protection methods and tools
- Computation time for implementing CKM Hypercubes with different hierarchies.
- Challenges of CKM in R and computational resource constraints

Confidentiality

- Sensitive or personal information refers to any data that can identify or be linked to an individual (statistical unit).
- Confidentiality is a core principle of official statistics, ensuring that sensitive information about individuals remains protected and undisclosed.
- Statistical disclosure control encompasses various methods to prevent the identification of individuals, households, or their characteristics in published datasets.

Why Statistical Disclosure Control is Necessary

- We are committed to ensuring that respondents' information is used solely for statistical purposes and that confidentiality is upheld.
- Honoring this commitment is an ethical obligation to protect the information entrusted to us.
- A breach of confidentiality may result in penal and administrative sanctions, as outlined in Articles 8-9 of Greek Statistical Law 3832/2010.

Census Data Protection Methods and Tools

- To ensure output harmonization, the EU recommended specific Statistical Disclosure Control (SDC) methods for Census 2021, facilitating cross-country comparisons of census data.
- In the project Harmonized Protection of Census Data in the ESS (2016–2017), involving six EU countries, best practices for Census 2021 were defined and tested. The recommended methods included:
  - a) Targeted Record Swapping (TRS)
  - b) The Cell Key Method (CKM)
  - c) A combination of TRS and CKM for enhanced data protection.
- Another EU project, Open-Source Tools for Perturbative Confidentiality Methods (2018–2019), involved seven EU countries. This project produced user-friendly, open-source software packages (ARGUS and R) to implement these confidentiality methods.

ELSTAT Decision on 2021 Census Data

- After testing the recommended methods and software packages, the Cell Key Method (CKM) in R was selected for the 2021 Population-Housing Census data.
- This method was applied to both EU and national tables to ensure data confidentiality while preserving the characteristics and usefulness of the data.
- The CKM's impact on the data was minimal, making it an ideal choice for maintaining data integrity.

R and System Setup

- R setup:
  a. data.table (1.15.2)
  b. dplyr (1.1.4)
  c. doParallel (1.0.17)
  d. sdcMicro (5.7.7)
  e. cellKey (1.0.2)
  f. ptable (1.0.0)
  g. RhpcBLASctl (0.23-42 - OpenBlas Version 0.3.26+ds-1)
- Infrastructure Requirements: Processing 10,482,487 records required substantial infrastructure. Ultimately, we utilized Oracle Cloud resources with 1,65TB of RAM on AMD EPYC.

Computation Time for Implementing CKM Hypercubes with 2 to 7 Variables

Hypercubes with less than 6 variables	HOURS	MINUTES	SECONDS
5 variables			
HC03		7	37
HC11			46
HC40			5
4 variables			
HC08			37
HC33			5
HC37			9
3 variables			
HC34			3
HC35			15
HC36			15
HC38			15
2 variables			
HC41			13

Hypercubes with 6 variables	HOURS	MINUTES	SECONDS
HC01			29
HC04		3	37
HC07			30
HC09	2	16	51
HC19		1	7
HC20			53
HC22			29
HC27			42
HC28			38
HC39		1	57

Hypercubes with 7 variables	HOURS	MINUTES	SECONDS
HC05		7	55
HC10		3	15
HC17		18	47
HC18		1	59
HC23		1	6
HC24		1	32
HC25		4	21
HC26		9	1

It was observed that hypercubes with the same number of variables exhibited variability in computation times. This variability can be attributed to the hierarchy of the variables within the hypercubes.

Challenges of CKM in R

- Processing Time: Implementing CKM for hypercubes with more than eight variables significantly increased processing time.
- Resources optimization: Using parallel computation we applied CKM in a time efficient way.
- System Failures: For two hypercubes (HC12 and HC13), system failures occurred, necessitating the development of an alternative approach for applying the CKM method.

Comparison of 3 HC by Levels of Hierarchy

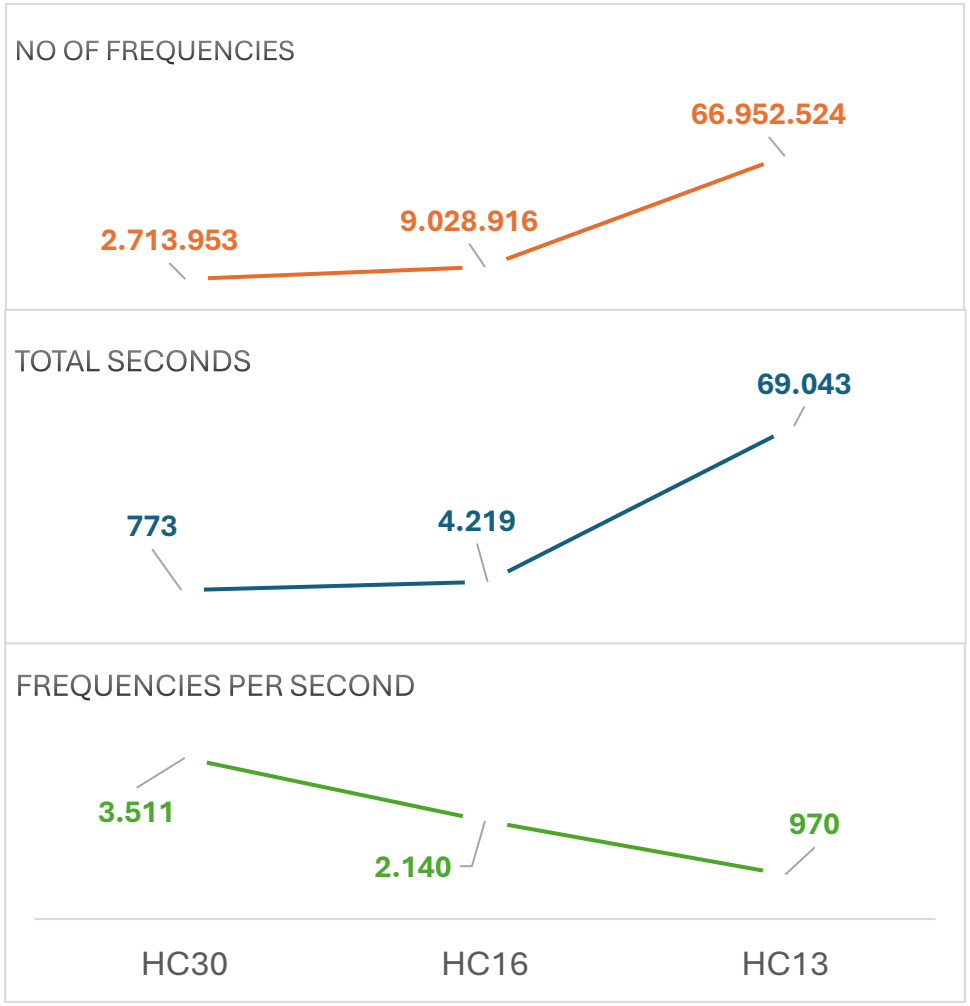
HC30		Hierarchies		
		LEVEL 1	*LEVEL 2*	*LEVEL 3*
GEO	L	5	14	
SEX		2		
AGE	M	6	21	
LMS	L	5		
FST	L	5		
HST	M	2	5	
CAS	L	3	2	
COC	L	4	2	

HC16		Hierarchies		
		LEVEL 1	*LEVEL 2*	*LEVEL 3*
GEO	L	5	14	
SEX		2		
AGE	M	6	21	
OCC		12		
COC	L	4	2	
POB	L	4	2	
YAE	L	3	9	
ROY		4	2	2

HC13		Hierarchies		
		LEVEL 1	*LEVEL 2*	*LEVEL 3*
GEO	M	5	14	53
SEX		2		
AGE	M	6	21	
COC	M	4	2	6
POB	M	4	2	6
YAE	H	3	9	17
ROY		4	2	2
HAR		3	2	

- HC16 has a more extensive hierarchies than HC30. For instance, the variable ROY in HC16 has three levels of hierarchy, along with an additional variable with two levels.
- HC13 contains variables with even more complex hierarchies, including five variables with three levels each.

Comparison of 3 HC by no of Frequencies, Time and Computation Performance



	NO OF FREQUENCIES	TOTAL SECONDS	FREQUENCIES PER SECOND
HC30	2.713.953	773	3.511
HC16	9.028.916	4.219	2.140
HC13	66.952.524	69.043	970

- As the frequency of CKM applications increases, the computation time grows exponentially.
- However, the computation performance defined as the ratio of the number of frequencies to the total processing time in seconds declines.
- The increased complexity of hierarchies' results in reduction of the computational efficiency.

Computation Time for Implementing CKM Hypercubes with 8 and 9 Variables

Hypercubes with 8 variables

	HOURS	MINUTES	SECONDS
HC02	4	7	6
HC06	1	18	6
HC15	1	5	43
HC16	1	10	19
HC21		58	34
HC29		13	29
HC30		12	53
HC31		25	50
HC32		24	32

Hypercubes with 8 variables

	PART*	HOURS	MINUTES	SECONDS
HC12		**8**	**4**	**13**
	EL		6	5
	3		44	40
	4	1	21	7
	5	3	3	38
	6	2	43	57
	9		4	42
HC13		**19**	**10**	**43**
	EL		12	54
	3	2	19	38
	4	3	50	48
	5	6	28	8
	6	6	8	43
	9		10	32

Hypercubes with 9 variables

	HOURS	MINUTES	SECONDS
HC14	8	3	8

* The Cell Key Method (CKM) was implemented separately for HC12 and HC13, for each NUTS1 region.

HELLENIC STATISTICAL AUTHORITY

Solution for HC12, HC13

- As mentioned earlier, implementing the CKM in HC12 and HC13 resulted in system failures. To address this, we tested the alternative method on a smaller hypercube, HC11, to evaluate the results.
- Specifically, we split the geographic variable NUTS3 by NUTS1 regions to reduce the hypercube's size. The CKM method was then applied to each NUTS1 region separately, removing the NUTS3 totals from each part.
- Additionally, the method was applied to the hypercube without the geographic variable to obtain the total values.
- Finally, all the pieces were merged, and the results matched exactly with those of the fully executed HC11, confirming the validity of the approach.

R Code (1) – Hierarchies and Dimensions

```r
# Load hierarchy files
hrc_names <- c("age_m", "coc_h", "pob_l", "yae_l",
               "geo_m_nuts3_3", "geo_m_nuts3_4", "geo_m_nuts3_5",
               "geo_m_nuts3_6", "geo_m_nuts3_9")
for (i in hrc_names) {
  print(paste("Loading hierarchy ", i, ".hrc", sep = ''))
  assign(i, hier_import(inp = paste(hrcdirectory, i, ".hrc", sep = ''), from = "hrc"))
}

d_sex <- hier_create(nodes = c("1", "2"), root = "T")

# Unified cube object
cubes <- list(
  HC11 = list(
    cubeNUTS = "NUTS3",
    dimensions = list(
      HC11_3 = list(NUTS3 = geo_m_nuts3_3, SEX = d_sex, AGE_M = age_m, COC_H = coc_h, YAE_L = yae_l),
      HC11_4 = list(NUTS3 = geo_m_nuts3_4, SEX = d_sex, AGE_M = age_m, COC_H = coc_h, YAE_L = yae_l),
      HC11_5 = list(NUTS3 = geo_m_nuts3_5, SEX = d_sex, AGE_M = age_m, COC_H = coc_h, YAE_L = yae_l),
      HC11_6 = list(NUTS3 = geo_m_nuts3_6, SEX = d_sex, AGE_M = age_m, COC_H = coc_h, YAE_L = yae_l),
      HC11_9 = list(NUTS3 = geo_m_nuts3_9, SEX = d_sex, AGE_M = age_m, COC_H = coc_h, YAE_L = yae_l),
      HC11_EL = list(SEX = d_sex, AGE_M = age_m, COC_H = coc_h, YAE_L = yae_l)
    )
  )
)
```

R Code (2) – DoParallel Loop / CKM

```r
# Process cubes in parallel
temp_list <- foreach(cube_name = names(cubes), .combine = 'c', .packages = c("data.table", "cellKey", "ptable")) %dopar% {
  local_log <- c()  # Local log collector
  result <- list()

  cube <- cubes[[cube_name]]
  cubeNUTS <- cube$cubeNUTS
  dimensions <- cube$dimensions

  regions <- sub("^[^_]*_", "", names(dimensions))

  for (region in regions) {
    tempfilename <- paste("out", cube_name, "_", region, "_", format(Sys.time(), "%Y-%m-%d_%H:%M"), sep = '')
    output_csv <- paste(outputdirectory, tempfilename, ".csv", sep = '')
    filtered_csv <- paste(outputdirectory, tempfilename, "_filtered2.csv", sep = '')

    # Log start locally
    local_log <- c(local_log, paste(as.character(Sys.time()), "Start cube:", cube_name, "region:", region, "\n"))

    # Perturb data
    dim_list <- cubes[[cube_name]]$dimensions[[paste(cube_name, region, sep = "_")]]

    if (region != "EL") {
      temp_data <- ck_setup(
        x = dat_persons[NUTS1 == as.numeric(region)],
        rkey = "RKEY",
        dims = dim_list
      )
    } else {
      temp_data <- ck_setup(
        x = dat_persons,
        rkey = "RKEY",
        dims = dim_list
      )
    }
    temp_data$params_cnts_set(val = ck_params_cnts(ptab = ptab_persons), v = "total")
    temp_data$perturb(v = "total")
```

R Code (3) – Handling the Totals

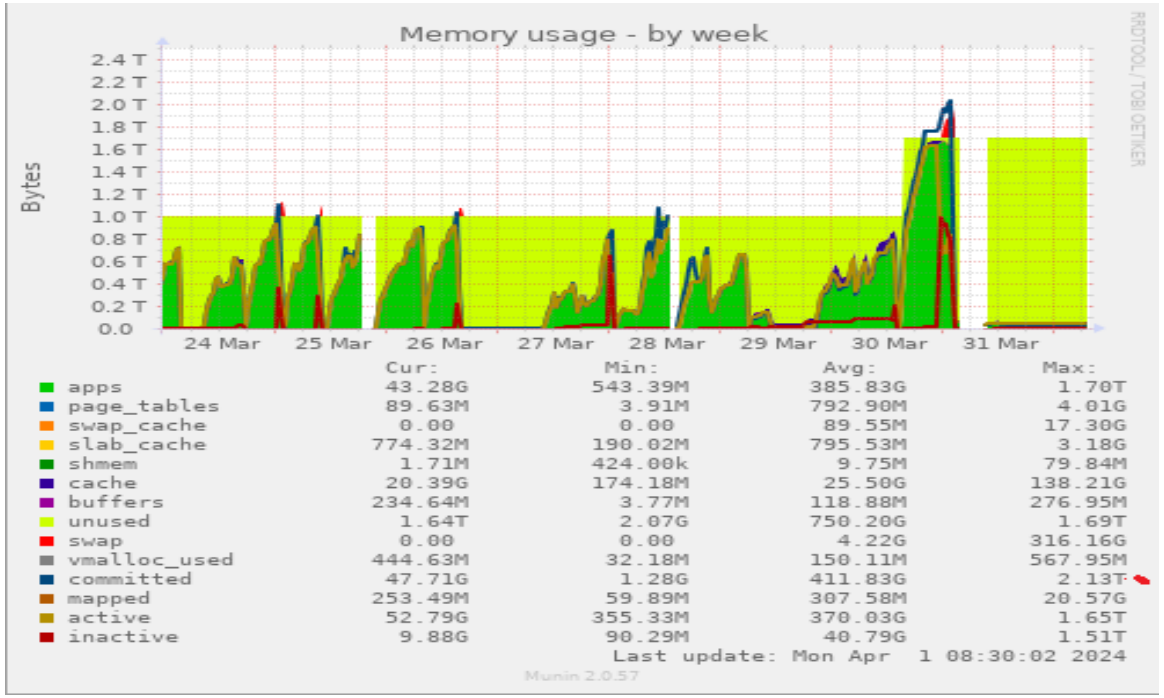
```r
# Process each result in `results`
for (region_name in names(results)) {
  # Identify the current cube and region
  cube_name <- sub("_.*", "", region_name)  # Extract the cube name (e.g., "HC11")
  region <- sub("^[^_]*_", "", region_name)  # Extract the region name (e.g., "3", "EL")

  # Read the corresponding file
  file_path <- results[[region_name]]
  if (!file.exists(file_path)) next  # Skip if the file does not exist

  data <- fread(file_path)

  # Handle "Total" (EL) and non-total regions
  if (region == "EL") {
    # Add "Total" row dynamically
    dat_temp_T <- cbind(setNames(data.frame(rep("Total", nrow(data))), cubes[[cube_name]]$cubeNUTS), data)
  } else {
    if (!exists("dat_temp_nonT")) {
      dat_temp_nonT <- data
    } else {
      dat_temp_nonT <- rbind(dat_temp_nonT, data, fill = TRUE)
    }
  }
}
```

HELLENIC STATISTICAL AUTHORITY

R Code (4) – Parts Merging

```
165 for (cube_name in names(cubes)) {
166   # Extract the column name dynamically
167   cube_nuts_col <- cubes[[cube_name]]$cubeNUTS
168   # Filter rows where the column is not "Total"
169   dat_temp_nonT_clean <- dat_temp_nonT[dat_temp_nonT[[cube_nuts_col]] != "Total"]
170
171   # Combine totals with non-totals if both exist
172   dat_temp_final <- rbindlist(list(dat_temp_T, dat_temp_nonT_clean))
173
174   # Generate a unique filename for the output
175   tempfilename <- paste("out", cube_name, "_", format(Sys.time(), "%Y-%m-%d_%H:%M"), sep = '')
176
177   # Write the final combined data to a CSV file
178   final_csv_path <- file.path(outputdirectory, paste(tempfilename, "_pert2.csv", sep = ''))
179   fwrite(dat_temp_final, final_csv_path, sep = ";")
180 }
```

System Memory Usage



- The diagram illustrates the system's burden during the application of the CKM method for EU hypercubes.
- The process utilized 1.65TB of RAM, and the most complex hypercube (HC13) required an additional 400GB swap file. It run in sequence of two NUTS3 parts in parallel.

HELLENIC STATISTICAL AUTHORITY

THE USE OF R
IN OFFICIAL STATISTICS

12th INTERNATIONAL CONFERENCE

Thank you

* Special thanks and lots of love to Voula Kouziou, Nikos Lagkouvardos and Panos Koulas

HELLENIC STATISTICAL AUTHORITY