LIITE / BILAGA 2

Kokeilutoimipaikkojen otannan ohjelmointikomennot
Programmeringskommandon för urvalet av försöksverksamhetsställen

Ohjelmointikomennot koskevat toimipaikkamäärältään isompia kokeilukuntia (kuntapäätöksen VN/583/2021 liitteen 1 s. 1):

Programmeringskommandon gäller försökskommuner med ett större antal verksamhetsställen (kommunbeslutet VN/583/2021 bilaga 1 s. 1):

```
# Code author:
# Ramin Izadi
# ramin.izadi@aalto.fi

rm(list=ls())
library(tidyverse)
library(haven)
library(gtools) # for odd
library(rstatix) # for t_test
library(future.apply) # for future_replicate
library(psych)
source("funktiot.R")

#sessionInfo()
set.seed(105012) #ssmmhh time just before executing

toimipaikka_muuttujat = read_csv("D:/Data/Output/toimipaikat_FINAL.csv")

#  Switch oid to numeric -------------------------------------------------
--

toimipaikka_muuttujat = toimipaikka_muuttujat %>%
  mutate(numeric_oid = row_number())

# Explore balancing covariates -------------------------------------------
--

toimipaikka_muuttujat %>% select(oid_ncoh16:oid_loedusha) %>% summary

corPlot(toimipaikka_muuttujat %>% select(oid_ncoh16:oid_loedusha))
ggplot(gather(toimipaikka_muuttujat %>% select(oid_ncoh16:oid_loedusha)),
aes(value)) +
  geom_histogram(bins = 20) +
  facet_wrap(~key, scales = "free_x")

# Number of potential allocations in each municipality
```

```
toimipaikka_muuttujat %>% group_by(toimipaikan_kuntanimi) %>%
  summarise(n = n(),
            comb = choose(n(), round(n()*0.4))) %>%
  arrange(-n)


# Simulate assignment vectors and calculate distance --------------------
--

# Continuous covariates to balance
covariate_vector = c("oid_cost",
                     "oid_invaka",
                     "oid_athome",
                     #"oid_ncoh16",
                     "oid_hhi_lang",
                     "oid_mekvinc",
                     "oid_singlesha",
                     "oid_meedusha", "oid_loedusha")

# Calculate covariance matrix for Mahalanobis distance
covariance = toimipaikka_muuttujat %>%
  select(all_of(covariate_vector)) %>% cov

# Rerandomization function
mahalanobis_distance = function(assignment_share = 0.5, ...){

  # From each group, randomly choose half into treatment (rounded down)
  koeryhma = toimipaikka_muuttujat %>% group_by(toimipaikan_kuntanimi)
%>%
    slice_sample(prop = assignment_share) %>% # slice_sample rounds down
    pull(numeric_oid)

  # Choose extra treatment units from groups with non-divisible N
  misfits1 = toimipaikka_muuttujat %>% group_by(toimipaikan_kuntanimi)
%>%
    filter((n()*assignment_share)%%1 != 0 & n()*assignment_share > 1) %>%
# munis with non-divisible number of units
    filter(!numeric_oid %in% koeryhma) %>% # available units (not yet
chosen for treatment)
    slice_sample(n = 1) %>% ungroup %>% # choose randomly one unit from
each non_divisible group
    slice_sample(prop = assignment_share) %>% # choose a share into the
treatment group
    pull(numeric_oid)

  # Choose at least one from too small munis
  misfits2 = toimipaikka_muuttujat %>% group_by(toimipaikan_kuntanimi)
%>%
    filter((n()*assignment_share) < 1) %>% # munis with N smaller than
assignment share
    slice_sample(n = 1) %>% ungroup %>% # choose one unit from each small
group
    pull(numeric_oid)

  koeryhma = c(koeryhma, misfits1, misfits2)
  verrokkiryhma = toimipaikka_muuttujat %>% filter(!numeric_oid %in%
koeryhma) %>% pull(numeric_oid)

  # Mean of the covariates in the treatment group
  treat = toimipaikka_muuttujat %>%
    filter(numeric_oid %in% koeryhma) %>%
    select(all_of(...)) %>%
    summarise_all(mean) %>% as.matrix

  # Mean of the covariates in the control group
  control = toimipaikka_muuttujat %>%
    filter(numeric_oid %in% verrokkiryhma) %>%
    select(all_of(...)) %>%
    summarise_all(mean) %>% as.matrix

  # Mahalnobis distance between the mean vectors
  # (Euclidian distance between the two means in a transformed space)
  distance = mahalanobis(x = treat, center = control, cov = covariance)
%>% sqrt

  # Save into a tibble
  tibble(distance = distance, koeryhma = list(koeryhma), verrokkiryhma =
list(verrokkiryhma))
}

# Generate admissible allocations
# Number of replications: 1 000 000 (2.5h with parallel processing, 25Gb
of RAM)

plan(multisession)
detach("package:tidylog", unload = T)

system.time(
  # Parallel processing function that replicates the above function
  (assignment_vectors = future_replicate(1000000,
mahalanobis_distance(assignment_share = 0.4,

all_of(covariate_vector)),
                                         simplify = F,
                                         future.seed = TRUE))
)

assignment_vectors = bind_rows(assignment_vectors)
assignment_vectors

# Choose the most balanced subset and pick the final treatment assignment
vector ------------------------------------------------------------------
-------

# Discard assignments vectors that exceed the budget

# Add data columns and cost column to 10k best assignment vectors
assignments_within_budget =
  assignment_vectors %>% arrange(distance) %>% filter(row_number() %in%
1:10000) %>%
  mutate(koeryhma_data = map(koeryhma, ~toimipaikka_muuttujat %>%
filter(numeric_oid %in% .x)),
         verrokkiryhma_data = map(verrokkiryhma, ~toimipaikka_muuttujat
%>% filter(numeric_oid %in% .x)),
         cost = map2(koeryhma_data, verrokkiryhma_data,
                     ~sum(.x$oid_cost) + sum(.y$oid_invaka*50) +
sum(.y$oid_athome*50))) %>%
  unnest(cols = cost)

# Filter out potential assignments that exceed the budget
assignments_within_budget = assignments_within_budget %>% filter(cost <
9600000)

# Choose the subset of admissible allocations (Use this for exact p-value
calculations)
# If Fisher exact p-value < 0.01 then H > 200

H = assignments_within_budget %>% arrange(distance) %>%
  filter(row_number() %in% 1:2000)

# Choose final treatment assignment vector and save it
H = H %>% mutate(chosen_assignment = row_number() == sample(1:2000, 1))

# Unnest chosen assignment vector
assignment_final = H %>% filter(chosen_assignment ==  T) %>%
  gather(key = treat, value = numeric_oid, koeryhma, verrokkiryhma) %>%
  unnest(cols = numeric_oid)

assignment_final = assignment_final %>%
  left_join(select(toimipaikka_muuttujat, toimipaikan_kuntanimi,
                   toimipaikan_kunta_koodi, numeric_oid,
toimipaikan_oid)) %>%
  arrange(toimipaikan_kuntanimi) %>%
  select(toimipaikan_kuntanimi, toimipaikan_kunta_koodi, toimipaikan_oid,
numeric_oid, treat)


# Save

#save(assignment_vectors, file = "D:/Data/Output/final
randomization/all_assignment_vectors.Rdata")
#save(H, file = "D:/Data/Output/final
randomization/permissible_assignment_vectors.Rdata")
#save(assignment_final, file ="D:/Data/Output/final
randomization/final_assignment_vector.Rdata")

#write_excel_csv(assignment_final, file ="D:/Data/Output/final
randomization/final_assignment_vector.csv")
```