

```

1  ### SCRIPT 9 - R ###
2  # Tratamiento y limpieza de datos global de la base de datos
3  # hasta obtener la base de datos principal (3900 casos)
4  #####
5
6  ### Lectura de Librerias -----
7  library(lubridate)
8  library(XML)
9  library(corrplot)
10 library(dplyr)
11 library(foreach)
12 library(stringr)
13 library(stringi)
14 library(doParallel)
15 library(DBI)
16 library(ggplot2)
17 library(dplyr)
18 library(openxlsx)
19 library(RODBC)
20 library(quanteda) #paquete recomendado en todos laos
21 library(quanteda.textmodels) #aux
22 library(quanteda.textstats) # aux
23 library(readtext) # sencilla manera de leer data de texto en R
24 library(spacyr) # NLP usando la libreria spaCy, incluyendo etiquetado part-of-speech,
entity recognition y dependency parsing.
25 library(zoo)
26 library(DBI)
27 library(tidyr)
28 library(readr)
29 #setup parallel backend to use many processors
30 cores=detectCores()
31 cl <- makeCluster(cores[1]-1, type = "PSOCK", outfile = "log.txt") #not to overload
your computer #,outfile="log.txt"
32 registerDoParallel(cl)
33
34 f_ini <- Sys.time()
35
36
37 ## Directorio ORIGEN----
38
39 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP")
40
41
42 ### Lectura de datos -----
43
44 #Awarding
45 load("DF_Awarding_Final.RData")
46
47 #Modificaciones
48 load("DF_Modis_FinalPrice.RData")
49
50 #Base de datos completa
51 load("BD_PLACSP.RData")
52
53
54 #Base de datos final de AEIPRO 2025, casos mas limpios
55
56 load("DF_Modis_Award_clean.RData")
57
58 ##Limpieza y transformacion de DF_Modis_Award_clean ----
59 DF_Modis_Award_clean <- DF_Modis_Award_clean |>
60   # Transformar y seleccionar columnas con renombramiento
61   transmute(
62     Identificador,
63     Identificador_lici,
64     Identificador_lote,
65
66     CPV_lote = CPV_division,
67     Ultima_actualizacion=Last_updated,
68     Modificacion_coste=Modification_cost,
69     Modificacion_p=Modification_p,
70     Baja_porcentaje=Bid_reduction_p,

```

```

71     Intervalo_hasta_modi_m=Interval_time_m,
72     C_precio_p=Price_p,
73     C_otros_objetivos_p=Other_objective_p,
74     C_juicios_valor_p=Value_judgement_p,
75
76     Precio_final = Final_Price
77 ) |>
78 filter(!is.na(Precio_final)) #Quitamos casos con NA en precio final
79
80 DF_Modis_Award_clean <- DF_Modis_Award_clean |>
81 mutate(
82     Ultima_actualizacion = as.Date(paste0(Ultima_actualizacion, "-01"))
83 )
84
85
86 ### Limpieza y transformacion DF de PLACSP -----
87
88 BD_PLACSP <- BD_PLACSP |> #Seteamos el formato de fecha de nuevo
89 mutate(
90     Primera_publicacion = as.Date(as.numeric(Primera_publicacion), origin =
91         "1899-12-30")
92 )
93
94 BD_PLACSP_clean <- BD_PLACSP |> # Quitamos error de cargar XML
95 mutate(across(
96     where(is.character),
97     ~ str_replace_all(.x, 'xml:space="preserve">', "")
98 ))
99
100
101 BD_PLACSP_clean <- BD_PLACSP_clean |>
102 # Crear columnas Identificador
103 mutate(
104     Identificador_lici = str_c(Identificador, coalesce(as.character(
105         Numero.de.expediente), "NA"), sep = "//"),
106     Identificador_lote = str_c(Identificador, coalesce(as.character(
107         Numero.del.contrato.licitacion_lote), "NA"), sep = "//")
108 ) |>
109
110 # Transformar y seleccionar columnas con renombramiento
111 transmute(
112     Identificador,
113     Identificador_lici,
114     Identificador_lote,
115     Primera_publicacion=Primera_publicacion,
116     Objeto_del_Contrato=Objeto.del.Contrato,
117
118     Presupuesto_licitacion = parse_number(Presupuesto.base.sin.impuestos),
119     CPV = CPV,
120
121     Tipo_de_contrato = Tipo.de.contrato,
122     Tipo_de_Administracion = Tipo.de.Administracion,
123     Tipo_de_procedimiento = Tipo.de.procedimiento,
124     Tramitacion = Tramitacion,
125     Lote = Lote,
126     Codigo_Postal = Codigo.Postal,
127     Tipo_ganador_lote=Tipo.de.identificador.de.adjudicatario.por.licitacion_lote,
128     Ganador_NIF = Identificador.Adjudicatario.de.la.licitacion_lote,
129
130     Presupuesto_licitacion_lote = parse_number(
131         Presupuesto.base.sin.impuestos.licitacion_lote),
132     CPV_lote = CPV.licitacion_lote,
133
134     N_ofertantes = parse_number(Numero.de.ofertas.recibidas.por.licitacion_lote),
135     Precio_adjudicacion = parse_number(
136         Importe.adjudicacion.sin.impuestos.licitacion_lote)
137 ) |>
138 filter(!is.na(Precio_adjudicacion)) #Quitamos casos con NA en precio final
139
140 ## Eliminamos casos duplicados
141 BD_PLACSP_clean<-unique(BD_PLACSP_clean)
142

```

```

139 ## Eliminamos casos duplicados menos en la fecha, que escogemos el más reciente
140
141 BD_PLACSP_clean <- BD_PLACSP_clean |>
142   group_by(across(-Primera_publicacion)) |>
143   slice_max(order_by = Primera_publicacion, n = 1, with_ties = FALSE) |>
144   ungroup()
145
146 # --- Función auxiliar: divide CPV por ":" y chequea si alguno cumple un patrón ---
147 tiene_algun_cpv_que_cumple <- function(cpv_string, patron_regex) {
148   cpvs <- str_split(cpv_string, ":", simplify = TRUE)
149   any(str_detect(cpvs, patron_regex))
150 }
151
152 # --- 1. Casos con al menos un CPV de construcción (empieza por 45) ---
153 bd_placsp_constru <- BD_PLACSP_clean |>
154   filter(
155     !is.na(CPV) &
156     sapply(CPV, tiene_algun_cpv_que_cumple, patron_regex = "^45")
157   )
158
159 # --- 2. Casos con al menos un CPV de ingeniería civil (71311) ---
160 bd_placsp_ingecivil <- BD_PLACSP_clean |>
161   filter(
162     !is.na(CPV) &
163     sapply(CPV, tiene_algun_cpv_que_cumple, patron_regex = "^71311")
164   )
165
166 # --- 3. Filtro extra: quitar casos donde todos los CPVs son de arquitectura ---
167 # Arquitectura: 4521 (excepto 133), 452625, 4526261, ..., 452629
168 # 1. Lista explícita de CPVs de arquitectura (puede ampliarse según necesidad)
169 cpvs_arquitectura <- c(
170   "45210000", "45262500", "45262610", "45262630", "45262650", "45262660",
171   "45262670", "45262680", "45262690", "45262700", "45262800", "45262900"
172 )
173
174 # 2. Función: TRUE si todos los CPVs están en la lista de arquitectura
175
176 solo_arquitectura <- function(cpv_string) {
177   cpvs <- str_split(cpv_string, ":", simplify = TRUE)
178   cpvs <- str_trim(cpvs) # limpiar espacios
179   cpvs <- cpvs[cpvs != ""] # quitar vacíos
180   all(cpvs %in% cpvs_arquitectura)
181 }
182
183 # 3. Aplicamos filtro final: nos quedamos con casos que NO son solo arquitectura
184 bd_placsp_civil <- bd_placsp_constru |>
185   filter(!sapply(CPV, solo_arquitectura))
186
187 # 4. Contamos casos de tipo Obra
188 n_casos_obra_civil <- bd_placsp_civil |>
189   distinct(Identificador) |>
190   filter("Obras" %in% bd_placsp_civil$Tipo_de_contrato) |>
191   nrow()
192
193
194 ## Hacemos un unique de bd_placsp_civil
195 bd_placsp_civil <- unique(bd_placsp_civil)
196
197 # Cruce por entryID (modificaciones) y Identificador (PLACSP civil)
198 n_casos_modi_civil <- DF_Modis_FinalPrice |>
199   distinct(entryID) |>
200   filter(entryID %in% bd_placsp_civil$Identificador) |>
201   nrow()
202
203
204 ##### CRUZAMOS BASES DE DATOS RESULTADO -----
205
206 # La idea general es buscar los casos que DF_Modis_Award_clean tiene dentro de
207 bd_placsp_civil
208 # Luego creamos un data frame nuevo llamado bd_civil_modis_award, con los datos de
209 ambos
210 # Buscamos Identificador_lote de bd_placsp_civil dentro de DF_Modis_Award_clean

```

```

210 # Si existen más de una coincidencia, quedarnos con los casos (cuantos haya) de
211 DF_Modis_Award_clean,
212 # y rellenar con los datos de bd_placsp_civil del caso con mayor valor en la variable
213 Primera_publicacion
214 # Añade una columna Fuente_info_civil como primera columna indicando si el dato fue
215 "Match 1 a 1" o "Match múltiple"
216
217 # Lista para guardar resultados finales del cruce
218 lista_resultados <- list()
219
220 # Iterar por cada Identificador_lote único
221 lotes_unicos <- unique(DF_Modis_Award_clean$Identificador_lote)
222
223 for (lote_id in lotes_unicos) {
224
225   # Casos correspondientes en ambas tablas
226   modi_casos <- DF_Modis_Award_clean |> filter(Identificador_lote == lote_id)
227   civil_casos <- bd_placsp_civil |> filter(Identificador_lote == lote_id)
228
229   # Si no hay match en civil, saltar al siguiente
230   if (nrow(civil_casos) == 0) next
231
232   # === CASO 1: 1x1 ===
233   if (nrow(modi_casos) == 1 && nrow(civil_casos) == 1) {
234     tipo_match <- "Match 1 a 1"
235     civil_sel <- civil_casos
236   } else {
237     # === CASO 2: múltiples coincidencias ===
238     tipo_match <- "Match múltiple (más reciente)"
239     civil_sel <- civil_casos |>
240       arrange(desc(Primera_publicacion)) |>
241       slice(1)
242   }
243
244   # Repetir civil_sel tantas veces como haya filas en modi_casos
245   civil_expandido <- civil_sel[rep(1, nrow(modi_casos)), ]
246
247   # Detectar columnas de civil que NO están ya en modi
248   columnas_nuevas <- setdiff(names(civil_expandido), names(modi_casos))
249
250   # Unión: primero el tipo de match, luego columnas de modi, luego solo las nuevas de
251   civil
252   combinado <- bind_cols(
253     tibble(Fuente_info_civil = tipo_match),
254     modi_casos,
255     civil_expandido[, columnas_nuevas, drop = FALSE]
256   )
257
258   # Agregar a la lista de resultados
259   lista_resultados[[length(lista_resultados) + 1]] <- combinado
260 }
261
262 # Combinar todo en un único data frame final
263 bd_civil_modis_award <- bind_rows(lista_resultados)
264
265 # 4. Contamos casos de tipo Obra
266 n_casos_final_civil <- bd_civil_modis_award |>
267   distinct(Identificador) |>
268   filter("Obras" %in% bd_civil_modis_award$Tipo_de_contrato) |>
269   nrow()
270
271 ## Directorio GUARDAR ----
272
273 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
274 atom/PLACSP/_final_conjunto_v4")
275
276 ##### Guardamos bases de datos resultado 1 -----
277
278 save(bd_civil_modis_award, file="bd_civil_modis_award.RData")
279 save(bd_placsp_civil, file="bd_placsp_civil.RData")
280 save(bd_placsp_ingecivil, file="bd_placsp_ingecivil.RData")

```

```

278 ##Exporto csv Inge Civil
279 write.xlsx(bd_civil_modis_award,"bd_civil_modis_award.xlsx", colNames = T, rowNames=F)
280 write.xlsx(bd_placsp_civil,"bd_placsp_civil.xlsx", colNames = T, rowNames=F)
281 write.xlsx(bd_placsp_ingecivil,"bd_placsp_ingecivil.xlsx", colNames = T, rowNames=F)
282
283 # Borramos variables auxiliares temporales
284 rm(civil_casos)
285 rm(civil_expandido)
286 rm(civil_sel)
287 rm(combinado)
288 rm(columnas_nuevas)
289 rm(lote_id)
290 rm(tipo_match)
291 rm(f_ini)
292 rm(lotes_unicos)
293 rm(modi_casos)
294 rm(lista_resultados)
295
296
297
298 ### Lectura de Librerias -----
299 library(readxl)
300 library(lubridate)
301 library(purrr)
302 library(XML)
303 library(corrplot)
304 library(dplyr)
305 library(foreach)
306 library(stringr)
307 library(stringi)
308 library(doParallel)
309 library(DBI)
310 library(ggplot2)
311 library(dplyr)
312 library(openxlsx)
313 library(RODBC)
314 library(quanteda) #paquete recomendado en todos laos
315 library(quanteda.textmodels) #aux
316 library(quanteda.textstats) # aux
317 library(readtext) # sencilla manera de leer data de texto en R
318 library(spacyr) # NLP usando la libreria spaCy, incluyendo etiquetado part-of-speech,
entity recognition y dependency parsing.
319 library(zoo)
320 library(DBI)
321 library(tidyr)
322 library(readr)
323 #setup parallel backend to use many processors
324 cores=detectCores()
325 cl <- makeCluster(cores[1]-1, type = "PSOCK", outfile = "log.txt") #not to overload
your computer #,outfile="log.txt"
326 registerDoParallel(cl)
327
328 f_ini <- Sys.time()
329
330
331 ## Directorio lectura ----
332
333 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP")
334
335
336 ### Lectura de datos ----
337
338 #Awarding
339 load("DF_Awarding_Final.RData")
340
341 #Modificaciones
342 load("DF_Modis_FinalPrice.RData")
343
344 #Base de datos completa
345 load("BD_PLACSP.RData")
346
347 #Base de datos plazo

```

```

348 load("DF_plazo_u.RData")
349
350 #Base de datos PLACSP de requerimientos
351 load("BD_placsp_quali.RData")
352
353 ## Directorio guardar ----
354
355 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
356
357
358
359 ### Limpieza y transformacion DF de PLACSP -----
360
361 BD_PLACSP <- BD_PLACSP |> #Seteamos el formato de fecha de nuevo
362 mutate(
363   Primera.publicacion = as.Date(as.numeric(Primera.publicacion), origin =
"1899-12-30")
364 )
365
366
367 BD_PLACSP_clean <- BD_PLACSP |> # Quitamos error de cargar XML
368 mutate(across(
369   where(is.character),
370   ~ str_replace_all(.x, 'xml:space="preserve">', '')
371 ))
372
373
374 BD_PLACSP_clean <- BD_PLACSP_clean |>
375 # Crear columnas Identificador
376 mutate(
377   Identificador_lici = str_c(Identificador, coalesce(as.character(
Numero.de.expediente), "NA"), sep = "//"),
378   Identificador_lote = str_c(Identificador, coalesce(as.character(
Numero.del.contrato.licitacion_lote), "NA"), sep = "//")
379 ) |>
380
381 # Transformar y seleccionar columnas con renombramiento
382 transmute(
383   Identificador,
384   Identificador_lici,
385   Identificador_lote,
386   Primera_publicacion=Primera.publicacion,
387   Fecha_actualizacion=Fecha.actualizacion,
388   Objeto_del_Contrato=Objeto.del.Contrato,
389
390   Presupuesto_licitacion = parse_number(Presupuesto.base.sin.impuestos),
391   CPV = CPV,
392
393   Tipo_de_contrato = Tipo.de.contrato,
394   Tipo_de_Administracion = Tipo.de.Administracion,
395   Tipo_de_procedimiento = Tipo.de.procedimiento,
396   Tramitacion = Tramitacion,
397   Lote = Lote,
398  Codigo_Postal = Codigo.Postal,
399   Tipo_ganador_lote=Tipo.de.identificador.de.adjudicatario.por.licitacion_lote,
400   Ganador_NIF = Identificador.Adjudicatario.de.la.licitacion_lote,
401
402   Presupuesto_licitacion_lote = parse_number(
Presupuesto.base.sin.impuestos.licitacion_lote),
403   CPV_lote = CPV.licitacion_lote,
404
405   N_ofertantes = parse_number(Numero.de.ofertas.recibidas.por.licitacion_lote),
406   Precio_adjudicacion = parse_number(
Importe.adjudicacion.sin.impuestos.licitacion_lote)
407 ) |>
408 filter(!is.na(Precio_adjudicacion)) #Quitamos casos con NA en precio final
409
410 ## Eliminamos casos duplicados
411 BD_PLACSP_clean<-unique(BD_PLACSP_clean)
412
413 ## Eliminamos casos duplicados menos en la fecha, que escogemos el más reciente
414

```

```

415 BD_PLACSP_clean <- BD_PLACSP_clean |>
416   group_by(across(-Fecha_actualizacion)) |>
417   slice_max(order_by = Fecha_actualizacion, n = 1, with_ties = FALSE) |>
418   ungroup()
419
420 # --- Función auxiliar: divide CPV por ":" y chequea si alguno cumple un patrón ---
421 tiene_algun_cpv_que_cumple <- function(cpv_string, patron_regex) {
422   cpvs <- str_split(cpv_string, ":", simplify = TRUE)
423   any(str_detect(cpvs, patron_regex))
424 }
425
426 # --- 1. Casos con al menos un CPV de construcción (empieza por 45) ---
427 bd_placsp_constru <- BD_PLACSP_clean |>
428   filter(
429     !is.na(CPV) &
430     apply(CPV, tiene_algun_cpv_que_cumple, patron_regex = "^45")
431   )
432
433 # --- 2. Casos con al menos un CPV de ingeniería civil (71311) ---
434 # --- Nuevos CPV a incluir explícitamente ---
435 cpv_extras_civil <- c(
436   "71322000", "71322100", "71322200", "71322300",
437   "71322400", "71322500", "71325000"
438 )
439
440
441 ## Para el analisis de buscar pares obra-servicio, en los que las obras ya limitan a
442 Inge Civil
443 cpv_extras_pares <-c(
444   "71322000", "71322100", "71322200", "71322300",
445   "71322400", "71322500", "71325000",
446   "71000000", #Servicios de ingenieria, arquitectura, construcción en general, para
447   recoger casos que han puesto el general
448   "71240000", #Servicios de arquitectura, ingeniería y planificación
449   "71241000", #Estudio de viabilidad, servicios de asesoramiento, análisis
450   "71242000", #Elaboración de proyectos y diseños, presupuestos
451   "71243000", # Anteproyectos (sistemas e integración)
452   "71250000", #Servicios de arquitectura, ingeniería y agrimensura
453   "71300000", #Servicios de ingenieria
454   "71310000", #Servicios de consultoría en ingeniería y construcción
455   "71312000", #Servicios de consultoría en ingeniería de estructuras
456   "71320000", #Servicios de diseño técnico
457   "71335000", #Estudios de ingeniería
458   "71336000" #Servicios complementarios de ingeniería
459 )
460
461 # --- Función que detecta coincidencias múltiples Inge Civil ---
462 tiene_cpv_ingecivil <- function(cpv_string) {
463   cpvs <- str_split(cpv_string, ":", simplify = TRUE)
464   cpvs <- str_trim(cpvs)
465
466   any(str_detect(cpvs, "^71311")) ||      # ingeniería civil clásica
467   any(str_detect(cpvs, "^71313")) ||      # ingeniería ambiental
468   any(substr(cpvs, 1, 8) %in% cpv_extras_civil) # códigos exactos extra inge civil
469 }
470
471 # --- Aplicamos el nuevo filtro ---
472 bd_placsp_ingecivil <- BD_PLACSP_clean |>
473   filter(!is.na(CPV) & apply(CPV, tiene_cpv_ingecivil))
474
475 # --- Función que detecta coincidencias múltiples Inge pares---
476 tiene_cpv_ingepares <- function(cpv_string) {
477   cpvs <- str_split(cpv_string, ":", simplify = TRUE)
478   cpvs <- str_trim(cpvs)
479
480   any(str_detect(cpvs, "^71311")) ||      # ingeniería civil clásica
481   any(str_detect(cpvs, "^71313")) ||      # ingeniería ambiental
482   any(substr(cpvs, 1, 8) %in% cpv_extras_pares) # códigos exactos extra inge civil
483   para pares
484 }
485
486 # --- Aplicamos el nuevo filtro ---
487 bd_placsp_ingepares <- BD_PLACSP_clean |>

```

```

485     filter(!is.na(CPV) & sapply(CPV, tiene_cpv_ingepares))
486
487
488
489
490
491 # --- 3. Filtro extra: quitar casos donde todos los CPVs son de arquitectura ---
492 # Arquitectura: 4521 (excepto 133), 452625, 4526261, ..., 452629
493 # 1. Lista explícita de CPVs de arquitectura (puede ampliarse según necesidad)
494 cpvs_arquitectura <- c(
495   "45210000", "45262500", "45262610", "45262630", "45262650", "45262660",
496   "45262670", "45262680", "45262690", "45262700", "45262800", "45262900"
497 )
498
499 # 2. Función: TRUE si todos los CPVs están en la lista de arquitectura
500
501
502 solo_arquitectura <- function(cpv_string) {
503   cpvs <- str_split(cpv_string, ":", simplify = TRUE)
504   cpvs <- str_trim(cpvs) # limpiar espacios
505   cpvs <- cpvs[cpvs != ""] # quitar vacíos
506   all(cpvs %in% cpvs_arquitectura)
507 }
508
509 # 3. Aplicamos filtro final: nos quedamos con casos que NO son solo arquitectura
510 bd_placsp_civil <- bd_placsp_constru |>
511   filter(!sapply(CPV, solo_arquitectura))
512
513 # 4. Contamos casos de tipo Obra
514 n_casos_obra_civil <- bd_placsp_civil |>
515   distinct(Identificador) |>
516   filter("Obras" %in% bd_placsp_civil$Tipo_de_contrato) |>
517   nrow()
518
519
520 ## Hacemos un unique de bd_placsp_civil
521 bd_placsp_civil <- unique(bd_placsp_civil)
522
523 ##### TRABAJAMOS LOS LOTES EN bd_placsp_civil -----
524 # Casos sin lotes
525 bd_placsp_civil_u <- bd_placsp_civil |>
526   filter(Lote == "Sin lotes")
527
528 ## En los casos con Identificador repetidos, nos quedamos con el de mas reciente fecha
de actualizacion
529 bd_placsp_civil_u <- bd_placsp_civil_u |>
530   group_by(Identificador) |>
531   slice_max(order_by = Fecha_actualizacion, n = 1, with_ties = FALSE) |>
532   ungroup()
533
534 # Casos con lotes
535 bd_placsp_civil_lotes <- bd_placsp_civil |>
536   filter(Lote != "Sin lotes")
537
538 # 1. Convertimos Lote a numérico solo para uso interno
539 bd_placsp_civil_lotes <- bd_placsp_civil_lotes |>
540   filter(!is.na(as.integer(Lote))) |>
541   mutate(Lote = as.integer(Lote)) |>
542
543 # 2. Eliminamos duplicados: nos quedamos con el lote más reciente
544 group_by(Identificador, Lote) |>
545 slice_max(order_by = Fecha_actualizacion, n = 1, with_ties = FALSE) |>
546 ungroup() |>
547
548 # 3. Filtramos los proyectos donde Lote cubre exactamente 1:N sin huecos
549 group_by(Identificador) |>
550 filter(n() == max(Lote)) |>
551 ungroup()
552
553 ## Para cada Identificador, generemos un solo caso, sumando las características de
los lotes como te digo a continuación en cada variable:
554 #- Ganador NIF: Concatenar los casos divididos por ":". Luego, hacer limpieza para
que no haya valores repetidos (entendiendo por valor los caracteres entre cada ":")

```



```

555 #- Presupuesto_licitacion_lote: Sumar los valores de todos
556 #- Precio_adjudicacion: Sumar los valores de todos
557 #- N_ofertantes: Escoger el valor maximo
558 #- CPV_lote: Concatenar los casos divididos por ":". Luego, hacer limpieza para que
no haya valores repetidos (entendiendo por valor los caracteres entre cada ":").
559 #- Lote: Se escoge el valor máximo
560 #- Identificador_lote: Concatenar los casos divididos por ":". Luego, hacer limpieza
para que no haya valores repetidos (entendiendo por valor los caracteres entre cada
":")
561 #- El resto de variables, nos quedamos con los valores del caso con Lote = 1
562
563 # --- Función auxiliar para limpiar concatenaciones ---
564 limpiar_concatenado <- function(x) {
565   x |>
566     str_split(":", simplify = TRUE) |>
567     as.vector() |>
568     str_trim() |>
569     discard(~ .x == "" || is.na(.x)) |>
570     unique() |>
571     paste(collapse = ":")
572 }
573
574 # --- Consolidación de los casos con lotes ---
575 bd_placsp_civil_lotes_resumido <- bd_placsp_civil_lotes |>
576   group_by(Identificador) |>
577   summarise(
578     Ganador_NIF = limpiar_concatenado(Ganador_NIF),
579     CPV_lote = limpiar_concatenado(CPV_lote),
580     Identificador_lote = limpiar_concatenado(Identificador_lote),
581
582     Presupuesto_licitacion_lote = sum(Presupuesto_licitacion_lote, na.rm = TRUE),
583     Precio_adjudicacion = sum(Precio_adjudicacion, na.rm = TRUE),
584
585     N_ofertantes = if (all(is.na(N_ofertantes))) NA_integer_ else max(N_ofertantes,
na.rm = TRUE),
586     Lote = max(as.integer(Lote), na.rm = TRUE)
587   ) |>
588 # --- Añadir las variables del lote 1 ---
589 left_join(
590   bd_placsp_civil_lotes |>
591     filter(Lote == 1) |>
592     dplyr::select(
593       Identificador,
594       Identificador_lici,
595       Primera_publicacion,
596       Fecha_actualizacion,
597       Objeto_del_Contrato,
598       Presupuesto_licitacion,
599       CPV,
600       Tipo_de_contrato,
601       Tipo_de_Administracion,
602       Tipo_de_procedimiento,
603       Tramitacion,
604      Codigo_Postal,
605       Tipo_ganador_lote
606     ),
607   by = "Identificador"
608 )
609
610 ## Juntamos bases de datos tratadas de bd_placsp_civil
611
612 # Asegurar que ambos 'N_lotes' son character
613 bd_placsp_civil_lotes_resumido <- bd_placsp_civil_lotes_resumido |>
614   rename(N_lotes = Lote) |>
615   mutate(N_lotes = as.character(N_lotes))
616
617 bd_placsp_civil_u <- bd_placsp_civil_u |>
618   rename(N_lotes = Lote) |>
619   mutate(N_lotes = as.character(N_lotes))
620
621 # Unir los data frames
622 bd_placsp_civil_globales <- bind_rows(
623   bd_placsp_civil_lotes_resumido,

```

```

624     bd_placsp_civil_u
625 )
626
627 ##### TRABAJAMOS E INTRODUCIMOS EL PLAZO CON DF_plazo_u -----
628
629 #Para cada Identificador de DF_plazo_u, escojas el que tiene fecha de actualizacion
mas tardia
630 DF_plazo_u <- DF_plazo_u |>
631   group_by(Identificador) |>
632   slice_max(order_by = Fecha_actualizacion, n = 1, with_ties = FALSE) |>
633   ungroup()
634
635 #Para cada Identificador de bd_placsp_civil_globales,
636 #busques el caso con Identificador correspondiente en DF_plazo_u,
637 #y le añadas los datos correspondientes de las columnas Plazo_m y Prorroga.
638 #Sino tienen, deja NA. Igual tienes que crear antes las columnas en
bd_placsp_civil_globales
639 #No quiero que hagas un joint del tiron
640
641 bd_placsp_civil_globales <- bd_placsp_civil_globales |>
642   mutate(
643     Plazo_m = NA_real_,
644     Prorroga = NA_character_
645   )
646
647 # Crear índice de matching
648 idx_plazo <- match(bd_placsp_civil_globales$Identificador, DF_plazo_u$Identificador)
649
650 # Asignar solo donde hay match válido
651 bd_placsp_civil_globales$Plazo_m[!is.na(idx_plazo)] <- DF_plazo_u$Plazo_m[idx_plazo[!
is.na(idx_plazo)]]
652 bd_placsp_civil_globales$Prorroga[!is.na(idx_plazo)] <- DF_plazo_u$Prorroga[idx_plazo
[!is.na(idx_plazo)]]
653
654 ##### TRABAJAMOS E INTRODUCIMOS LOS CRITERIOS DE ADJUDICACION CON DF_Awarding_Final
-----
655
656 ## Cambiamos nombre de variables
657 DF_Awarding_Final <- DF_Awarding_Final |>
658   # Crear columnas Identificador
659   mutate(
660     Identificador = entryID,
661     Identificador_lici = str_c(entryID, coalesce(as.character(ContractID), "NA"), sep
= "/"")
662   )
663
664 # 1. Quedarse con el primer caso por Identificador (no hay Fecha_actualizacion)
665 DF_Awarding_Final <- DF_Awarding_Final |>
666   group_by(Identificador) |>
667   slice(1) |>
668   ungroup()
669
670 # 2. Crear columnas vacías en bd_placsp_civil_globales
671 bd_placsp_civil_globales <- bd_placsp_civil_globales |>
672   mutate(
673     C_precio_p = NA_real_,
674     C_resto_objetivos_p = NA_real_,
675     C_juicio_valor_p = NA_real_
676   )
677
678 # 3. Match por Identificador
679 idx_award <- match(bd_placsp_civil_globales$Identificador, DF_Awarding_Final$
Identificador)
680
681 # 4. Asignar valores si hay match
682 bd_placsp_civil_globales$C_precio_p[!is.na(idx_award)] <-
683   DF_Awarding_Final$`%_precio`[idx_award[!is.na(idx_award)]]
684
685 bd_placsp_civil_globales$C_resto_objetivos_p[!is.na(idx_award)] <-
686   DF_Awarding_Final$`%_resto_objetivos`[idx_award[!is.na(idx_award)]]
687
688 bd_placsp_civil_globales$C_juicio_valor_p[!is.na(idx_award)] <-
689   DF_Awarding_Final$`%_juicios_valor`[idx_award[!is.na(idx_award)]]

```

```

690
691
692
693 ##### TRABAJAMOS SOBRE EL DATA FRAME DE MODIFICACIONES ORIGINAL -----
694 ## Directorio carga ----
695
696 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP")
697 # Cargamos el data frame si no está aún en el entorno
698 load("DF_Modificaciones.RData") # Asegurate de tenerlo cargado con este nombre
699
700 ## Directorio GUARDAR ----
701
702 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
703
704 # Cambiamos nombres y generamos el nuevo identificador
705 DF_mods <- DF_all_u |>
706   transmute(
707     Identificador = entryID,
708     ContractID = ContractID, # Necesaria para construir Identificador_lici
709     Identificador_lici = str_c(entryID, coalesce(as.character(ContractID), "NA"), sep
= "//"),
710     Fecha_actualizacion = updated,
711     Modi_temp = ContractModificationDurationMeasure,
712     Modi_temp_Uc = ContractModificationDurationMeasure_Uc,
713     Plazo_final = FinalDurationMeasure,
714     Plazo_final_Uc = FinalDurationMeasure_Uc,
715     Modi_eco_e = ContractModificationLegalMonetaryTotal,
716     Precio_final_e = FinalLegalMonetaryTotal
717   )
718
719 DF_mods <- DF_mods |> #Pasamos todos los plazos a meses
720   mutate(
721     Modi_temp_m = case_when(
722       Modi_temp_Uc == "ANN" ~ floor(Modi_temp * 12),
723       Modi_temp_Uc == "MON" ~ floor(Modi_temp),
724       Modi_temp_Uc == "DAY" ~ floor(Modi_temp / 30),
725       TRUE ~ NA_integer_
726     ),
727
728     Plazo_final_m = case_when(
729       Plazo_final_Uc == "ANN" ~ floor(Plazo_final * 12),
730       Plazo_final_Uc == "MON" ~ floor(Plazo_final),
731       Plazo_final_Uc == "DAY" ~ floor(Plazo_final / 30),
732       TRUE ~ NA_integer_
733     )
734   )
735
736 DF_mods <- DF_mods |> #Nos quedamos con las variables que queremos
737   dplyr::select(
738     Identificador,
739     Identificador_lici,
740     Fecha_actualizacion,
741     Modi_temp_m,
742     Plazo_final_m,
743     Modi_eco_e,
744     Precio_final_e
745   )
746
747 ### CRUZAMOS DF_mods con bd_placsp_civil_glboales -----
748 #Primero vamos a por los casos en que en la variable bd_placsp_civil_globales$N_lotes
corresponde a "Sin lotes".
749 #En estos casos, quiero que para el Identificador de bd_placsp_civil_globales busques
los casos con ese
750 #Identificador en DF_mods y escojas el caso con mayor valor en Precio_final_e.
751
752 # 1. Filtrar casos sin lotes de la base principal
753 bd_globales_u <- bd_placsp_civil_globales |>
754   filter(N_lotes == "Sin lotes") |>
755   mutate(
756     Modi_temp_m = NA_integer_,
757     Plazo_final_m = NA_integer_,

```

```

758     Modi_eco_e = NA_real_,
759     Precio_final_e = NA_real_
760 )
761
762 # 2. Obtener de DF_mods el caso con mayor Precio_final_e por Identificador
763 mods_u_filtrado <- DF_mods |>
764   group_by(Identificador) |>
765   slice_max(order_by = Precio_final_e, n = 1, with_ties = FALSE) |>
766   ungroup()
767
768 # 3. Hacer match y asignar de forma vectorizada
769 idx_mods <- match(bd_globales_u$Identificador, mods_u_filtrado$Identificador)
770
771 bd_globales_u$Modi_temp_m[!is.na(idx_mods)] <- mods_u_filtrado$Modi_temp_m[idx_mods[!
772 is.na(idx_mods)]]
773 bd_globales_u$Plazo_final_m[!is.na(idx_mods)] <- mods_u_filtrado$Plazo_final_m[
774 idx_mods[!is.na(idx_mods)]]
775 bd_globales_u$Modi_eco_e[!is.na(idx_mods)] <- mods_u_filtrado$Modi_eco_e[idx_mods[!
776 is.na(idx_mods)]]
777 bd_globales_u$Precio_final_e[!is.na(idx_mods)] <- mods_u_filtrado$Precio_final_e[
778 idx_mods[!is.na(idx_mods)]]
779
780 # Verificamos la condición de igualdad con tolerancia de Modificaciones en precio
781 coincide <- abs(bd_globales_u$Precio_final_e - bd_globales_u$Precio_adjudicacion -
782 bd_globales_u$Modi_eco_e) < 1e-6
783
784 # Filtrar índices válidos (no NA)
785 idx_corregir <- which(!coincide & !is.na(coincide))
786
787 # Mostrar porcentaje de coincidencia
788 porcentaje_ok <- mean(coincide, na.rm = TRUE) * 100
789 cat(sprintf("Coincidencia exacta en %.2f%% de los casos con 'Sin lotes'\n",
790 porcentaje_ok))
791
792 # Corregir únicamente los que no coinciden y tienen datos completos
793 bd_globales_u$Modi_eco_e[idx_corregir] <- bd_globales_u$Precio_final_e[idx_corregir] -
794 bd_globales_u$Precio_adjudicacion[idx_corregir]
795
796
797 # Verificar igualdad con tolerancia para plazos
798 coincide_plazo <- (bd_globales_u$Plazo_final_m - bd_globales_u$Plazo_m) ==
799 bd_globales_u$Modi_temp_m
800
801 # Filtrar índices válidos sin NA
802 idx_corregir_plazo <- which(!coincide_plazo & !is.na(coincide_plazo))
803
804 # Porcentaje de coincidencia
805 porcentaje_ok_plazo <- mean(coincide_plazo, na.rm = TRUE) * 100
806 cat(sprintf("Coincidencia exacta en %.2f%% de los casos con 'Sin lotes' (plazos)\n",
807 porcentaje_ok_plazo))
808
809 # Corregir solo donde no coincide y no hay NA
810 bd_globales_u$Modi_temp_m[idx_corregir_plazo] <- bd_globales_u$Plazo_final_m[
811 idx_corregir_plazo] - bd_globales_u$Plazo_m[idx_corregir_plazo]
812
813
814 ## Ahora vamos con los casos que N_lotes != "Sin Lotes", para los que la operativa es
815 diferente:
816 # Para cada Identificador de casos de bd_placsp_civil_globales con lotes, buscamos
817 los casos con mismo Identificador en DF_mods, y hacemos las siguientes
818 transformaciones:
819 #- La variable "Modi_eco_e" del caso en bd_placsp_civil_globales, será la suma de los
820 valores Modi_eco_e de los casos en DF_Modis
821 #- La variable Modi_temp_m del caso en bd_placsp_civil_globales, será el máximo de
822 los valores Modi_temp_m de los casos en DF_Modis
823 #- Las variables Plazo_final_m y Precio_final_e, se calcularán posteriormente en
824 bd_placsp_civil_globales como la suma de Modi_temp_m y Modi_eco_e con Plazo_m y
825 Precio_adjudicacion respectivamente
826
827 # Paso 1: Filtrar los casos con lotes desde la base principal
828 bd_globales_lotes <- bd_placsp_civil_globales |>
829   filter(N_lotes != "Sin lotes") |>

```

```

814     mutate(
815       Modi_temp_m = NA_integer_,
816       Modi_eco_e = NA_real_,
817       Plazo_final_m = NA_integer_,
818       Precio_final_e = NA_real_
819     )
820
821 # Paso 2: Eliminar duplicados por Identificador + importe, conservar el más reciente
822 DF_mods_filtrado <- DF_mods |>
823   group_by(Identificador, Modi_eco_e, Precio_final_e) |>
824   slice_max(order_by = Fecha_actualizacion, n = 1, with_ties = FALSE) |>
825   ungroup()
826
827 # Paso 3: Agrupar por Identificador y calcular agregados únicos
828 mods_lotes_agg <- DF_mods_filtrado |>
829   group_by(Identificador) |>
830   summarise(
831     Plazo_final_m = if (all(is.na(Plazo_final_m))) NA_integer_ else max(Plazo_final_m
832       , na.rm = TRUE),
833     Modi_eco_e = sum(Modi_eco_e, na.rm = TRUE),
834     .groups = "drop"
835   )
836
837 # Paso 4: Match por Identificador
838 idx_lotes <- match(bd_globales_lotes$Identificador, mods_lotes_agg$Identificador)
839
840 # Paso 5: Asignar valores económicos y de plazo final
841 bd_globales_lotes$Plazo_final_m[!is.na(idx_lotes)] <- mods_lotes_agg$Plazo_final_m[
842   idx_lotes[!is.na(idx_lotes)]]
843 bd_globales_lotes$Modi_eco_e[!is.na(idx_lotes)] <- mods_lotes_agg$Modi_eco_e[idx_lotes
844   [!is.na(idx_lotes)]]
845
846 # Paso 6: Calcular plazo modificado (como diferencia entre final y original)
847 bd_globales_lotes <- bd_globales_lotes |>
848   mutate(
849     Modi_temp_m = ifelse(
850       is.na(Plazo_final_m) | is.na(Plazo_m),
851       NA_integer_,
852       Plazo_final_m - Plazo_m
853     ),
854
855     # Precio final = adjudicación + modificación económica
856     Precio_final_e = ifelse(
857       is.na(Precio_adjudicacion) & is.na(Modi_eco_e),
858       NA_real_,
859       coalesce(Precio_adjudicacion, 0) + coalesce(Modi_eco_e, 0)
860     )
861   )
862
863 # --- Verificar y corregir el componente económico ---
864 coincide_econ <- abs(bd_globales_lotes$Precio_final_e - bd_globales_lotes$
865   Precio_adjudicacion - bd_globales_lotes$Modi_eco_e) < 1e-6
866 idx_corregir_econ <- which(!coincide_econ & !is.na(coincide_econ))
867
868 porcentaje_ok_econ <- mean(coincide_econ, na.rm = TRUE) * 100
869 cat(sprintf("Coincidencia exacta en %.2f%% de los casos con lotes (económico)\n",
870   porcentaje_ok_econ))
871
872 bd_globales_lotes$Modi_eco_e[idx_corregir_econ] <- bd_globales_lotes$Precio_final_e[
873   idx_corregir_econ] - bd_globales_lotes$Precio_adjudicacion[idx_corregir_econ]
874
875 # --- Verificar y corregir el componente de plazos ---
876 coincide_plazo <- (bd_globales_lotes$Plazo_final_m - bd_globales_lotes$Plazo_m) ==
877   bd_globales_lotes$Modi_temp_m
878 idx_corregir_plazo <- which(!coincide_plazo & !is.na(coincide_plazo))
879
880 porcentaje_ok_plazo <- mean(coincide_plazo, na.rm = TRUE) * 100
881 cat(sprintf("Coincidencia exacta en %.2f%% de los casos con lotes (plazo)\n",
882   porcentaje_ok_plazo))
883
884 bd_globales_lotes$Modi_temp_m[idx_corregir_plazo] <- bd_globales_lotes$Plazo_final_m[
885   idx_corregir_plazo] - bd_globales_lotes$Plazo_m[idx_corregir_plazo]
886

```

```

878 # --- Unión final: todos los casos con y sin lotes ---
879 bd_civil_todo <- bind_rows(
880   bd_globales_u,
881   bd_globales_lotes
882 )
883
884 # --- Calculo de nuevas variables ---
885 bd_civil_todo <- bd_civil_todo |>
886   mutate(
887     # 1. Baja porcentual (% de descuento sobre presupuesto)
888     Baja_p = ifelse(
889       !is.na(Presupuesto_licitacion) & Presupuesto_licitacion != 0,
890       round(((Presupuesto_licitacion - Precio_adjudicacion) / Presupuesto_licitacion)
891         * 100, 2),
892       NA_real_
893     ),
894     # 2. Modificación económica porcentual
895     Modi_eco_p = ifelse(
896       !is.na(Precio_adjudicacion) & Precio_adjudicacion != 0,
897       round(((Precio_final_e - Precio_adjudicacion) / Precio_adjudicacion) * 100, 2),
898       NA_real_
899     ),
900     # 3. Modificación temporal porcentual
901     Modi_temp_p = ifelse(
902       !is.na(Plazo_final_m) & Plazo_final_m != 0,
903       round(((Plazo_final_m - Plazo_m) / Plazo_final_m) * 100, 2),
904       NA_real_
905     ),
906     # 4. Intervalo entre publicación y última actualización (en meses, redondeado)
907     Intervalo_modi = ifelse(
908       !is.na(Fecha_actualizacion) & !is.na(Primera_publicacion),
909       round(as.numeric(difftime(Fecha_actualizacion, Primera_publicacion, units =
910         "days")) / 30),
911       NA_real_
912     )
913   )
914 # Si se han registrado modificaciones de ese proyecto
915
916 bd_civil_todo <- bd_civil_todo |>
917   mutate(
918     Modis_s_n = factor(
919       if_else(
920         Identificador %in% DF_mods$Identificador,
921         "Sí", "No"
922       ),
923       levels = c("Sí", "No")
924     )
925   )
926
927 # --- Vamos a filtrar los casos de bd_civil_todo con precios de adjudicacion menores
928 # a 10.000 --
929 bd_civil_todo <- bd_civil_todo |>
930   filter(Precio_adjudicacion >= 10000 | is.na(Precio_adjudicacion))
931
932 # --- Subconjunto: solo los casos con modificaciones económicas registradas ---
933 bd_civil_modis <- bd_civil_todo |>
934   filter(!is.na(Modi_eco_e))
935
936 # --- Subconjunto: solo los casos con modificaciones económicas registradas y sin NA
937 # en criterio economico ---
938 bd_civil_sin_NA <- bd_civil_modis |>
939   filter(!is.na(C_precio_p))
940
941 ##### VAMOS A METER LOS DATOS SOBRE REQUISITOS DE LAS LICITACIONES ##### -----
942
943 ### VAMOS A METER LOS DATOS SOBRE REQUISITOS DE LAS LICITACIONES ##### --
944
945
946 # Crear columnas vacías en bd_civil_todo

```

```

947 bd_civil_todo <- bd_civil_todo |>
948   mutate(
949     Clasi_empresa = NA_character_,
950     Requi_adm = NA_character_,
951     Clasi_max_req = NA_real_
952   )
953
954 # Realizar el matching vectorizado por Identificador
955 idx_quali <- match(bd_civil_todo$Identificador, BD_placsp_quali$Identificador)
956
957 # Asignar valores desde BD_placsp_quali
958 bd_civil_todo$Clasi_empresa[!is.na(idx_quali)] <- BD_placsp_quali$Clasi_empresa[
959   idx_quali[!is.na(idx_quali)]]
960 bd_civil_todo$Requi_adm[!is.na(idx_quali)] <- BD_placsp_quali$Requi_adm[idx_quali[!
961   is.na(idx_quali)]]
962 bd_civil_todo$Clasi_max_req[!is.na(idx_quali)] <- BD_placsp_quali$Clasi_max_req[
963   idx_quali[!is.na(idx_quali)]]
964
965 ## Actualizamos subconjuntos
966 # --- Subconjunto: solo los casos con modificaciones económicas registradas ---
967 bd_civil_modis <- bd_civil_todo |>
968   filter(!is.na(Modi_eco_e))
969
970 # --- Subconjunto: solo los casos con modificaciones económicas registradas y sin NA
971 # en criterio economico ---
972 bd_civil_sin_NA <- bd_civil_modis |>
973   filter(!is.na(C_precio_p))
974
975 ### VAMOS A METER LOS DATOS SOBRE FECHAS DEL PROYECTO##### ----
976
977 # Paso 0: Crear vector de identificadores únicos a usar
978 ids_utiles <- unique(c(bd_civil_modis$Identificador, bd_civil_sin_NA$Identificador))
979
980 # Paso 1: Filtrar BD_PLACSP por identificadores relevantes y columnas necesarias
981 BD_PLACSP_filtrado <- BD_PLACSP |>
982   filter(Identificador %in% ids_utiles) |>
983   dplyr::select(
984     Identificador,
985     Fecha.actualizacion,
986     Fecha.entrada.en.vigor.del.contrato.de.licitacion_lote,
987     Fecha.formalizacion.del.contrato.licitacion_lote,
988     Fecha.de.presentacion.de.ofertas
989   )
990
991 # Paso 2: Calcular fechas brutas
992 BD_PLACSP_fechas <- BD_PLACSP_filtrado |>
993   mutate(
994     Fecha_entrada_vigor_raw = as.Date(
995       coalesce(
996         Fecha.entrada.en.vigor.del.contrato.de.licitacion_lote,
997         Fecha.formalizacion.del.contrato.licitacion_lote
998       ),
999       origin = "1899-12-30"
1000     ),
1001     Fecha_presentacion_raw = as.Date(
1002       Fecha.de.presentacion.de.ofertas,
1003       origin = "1899-12-30"
1004     )
1005   )
1006
1007 # Paso 3: Mantener solo la fila más reciente por Identificador
1008 BD_PLACSP_max_fechas <- BD_PLACSP_fechas |>
1009   group_by(Identificador) |>
1010   slice_max(order_by = Fecha.actualizacion, n = 1, with_ties = FALSE) |>
1011   ungroup()
1012
1013 # Paso 4: Formatear las fechas como "YYYY/MM/DD"
1014 BD_PLACSP_max_fechas <- BD_PLACSP_max_fechas |>
1015   mutate(
1016     Fecha_entrada_vigor = format(Fecha_entrada_vigor_raw, "%Y/%m/%d"),
1017     Fecha_presentacion = format(Fecha_presentacion_raw, "%Y/%m/%d")
1018   ) |>
1019   dplyr::select(Identificador, Fecha_entrada_vigor, Fecha_presentacion)

```

```

1016
1017 # Paso 5: Añadir columnas vacías en los data frames destino
1018 bd_civil_modis <- bd_civil_modis |>
1019   mutate(
1020     Fecha_entrada_vigor = NA_character_,
1021     Fecha_presentacion = NA_character_
1022   )
1023
1024 bd_civil_sin_NA <- bd_civil_sin_NA |>
1025   mutate(
1026     Fecha_entrada_vigor = NA_character_,
1027     Fecha_presentacion = NA_character_
1028   )
1029
1030 # Paso 6: Match por Identificador y asignar valores si hay
1031 idx_modis <- match(bd_civil_modis$Identificador, BD_PLACSP_max_fechas$Identificador)
1032 idx_sin_na <- match(bd_civil_sin_NA$Identificador, BD_PLACSP_max_fechas$Identificador)
1033
1034 bd_civil_modis$Fecha_entrada_vigor[!is.na(idx_modis)] <-
1035   BD_PLACSP_max_fechas$Fecha_entrada_vigor[idx_modis[!is.na(idx_modis)]]
1036
1037 bd_civil_modis$Fecha_presentacion[!is.na(idx_modis)] <-
1038   BD_PLACSP_max_fechas$Fecha_presentacion[idx_modis[!is.na(idx_modis)]]
1039
1040 bd_civil_sin_NA$Fecha_entrada_vigor[!is.na(idx_sin_na)] <-
1041   BD_PLACSP_max_fechas$Fecha_entrada_vigor[idx_sin_na[!is.na(idx_sin_na)]]
1042
1043 bd_civil_sin_NA$Fecha_presentacion[!is.na(idx_sin_na)] <-
1044   BD_PLACSP_max_fechas$Fecha_presentacion[idx_sin_na[!is.na(idx_sin_na)]]
1045
1046 # --- Rellenar NA con valores auxiliares (nuevas reglas) ---
1047
1048 bd_civil_modis <- bd_civil_modis |>
1049   mutate(
1050     Fecha_presentacion = if_else(
1051       is.na(Fecha_presentacion) & !is.na(Primera_publicacion),
1052       as.character(Primera_publicacion),
1053       Fecha_presentacion
1054     ),
1055     Fecha_entrada_vigor = if_else(
1056       is.na(Fecha_entrada_vigor) & !is.na(Fecha_presentacion),
1057       Fecha_presentacion,
1058       Fecha_entrada_vigor
1059     )
1060   )
1061
1062
1063 # En bd_civil_sin_NA
1064 bd_civil_sin_NA <- bd_civil_sin_NA |>
1065   mutate(
1066     Fecha_presentacion = if_else(
1067       is.na(Fecha_presentacion) & !is.na(Primera_publicacion),
1068       as.character(Primera_publicacion),
1069       Fecha_presentacion
1070     ),
1071     Fecha_entrada_vigor = if_else(
1072       is.na(Fecha_entrada_vigor) & !is.na(Fecha_presentacion),
1073       Fecha_presentacion,
1074       Fecha_entrada_vigor
1075     )
1076   ) |>
1077   # Eliminar casos con ambas fechas aún en NA
1078   filter(!(is.na(Fecha_entrada_vigor) & is.na(Fecha_presentacion)))
1079
1080
1081 # --- Conversión de fechas a clase Date (YYYY-MM-DD) ---
1082 # Estándar: convertir todo a "YYYY-MM-DD" (ISO 8601)
1083 bd_civil_modis <- bd_civil_modis |>
1084   mutate(
1085     Fecha_presentacion = as.character(Fecha_presentacion),
1086     Fecha_entrada_vigor = as.character(Fecha_entrada_vigor),
1087     Primera_publicacion = as.character(Primera_publicacion)
1088   ) |>

```



```

1089     mutate(
1090       Fecha_presentacion = str_replace_all(Fecha_presentacion, "/", "-"),
1091       Fecha_entrada_vigor = str_replace_all(Fecha_entrada_vigor, "/", "-"),
1092       Primera_publicacion = str_replace_all(Primera_publicacion, "/", "-")
1093     )
1094 bd_civil_sin_NA <- bd_civil_sin_NA |>
1095     mutate(
1096       Fecha_presentacion = as.character(Fecha_presentacion),
1097       Fecha_entrada_vigor = as.character(Fecha_entrada_vigor),
1098       Primera_publicacion = as.character(Primera_publicacion)
1099     ) |>
1100     mutate(
1101       Fecha_presentacion = str_replace_all(Fecha_presentacion, "/", "-"),
1102       Fecha_entrada_vigor = str_replace_all(Fecha_entrada_vigor, "/", "-"),
1103       Primera_publicacion = str_replace_all(Primera_publicacion, "/", "-")
1104     )
1105
1106 convertir_fechas <- function(df) {
1107   df |>
1108     mutate(
1109       Fecha_entrada_vigor = as.Date(Fecha_entrada_vigor),
1110       Fecha_presentacion = as.Date(Fecha_presentacion),
1111       Primera_publicacion = as.Date(Primera_publicacion),
1112       Fecha_actualizacion = as.Date(Fecha_actualizacion)
1113     )
1114 }
1115
1116 bd_civil_modis <- convertir_fechas(bd_civil_modis)
1117 bd_civil_sin_NA <- convertir_fechas(bd_civil_sin_NA)
1118
1119 ## Ultima comprobacion, forzamos a Primer_publicacion a ser la mas temprana
1120
1121 # Aseguramos que Primera_publicacion sea la más temprana de las tres fechas relevantes
1122 bd_civil_modis <- bd_civil_modis |>
1123     mutate(
1124       Primera_publicacion = pmin(
1125         Primera_publicacion,
1126         Fecha_presentacion,
1127         Fecha_entrada_vigor,
1128         na.rm = TRUE
1129       )
1130     )
1131
1132 bd_civil_sin_NA <- bd_civil_sin_NA |>
1133     mutate(
1134       Primera_publicacion = pmin(
1135         Primera_publicacion,
1136         Fecha_presentacion,
1137         Fecha_entrada_vigor,
1138         na.rm = TRUE
1139       )
1140     )
1141
1142
1143
1144 # --- Outliers manuales (reassignar entrada_vigor = presentacion si en lista) ---
1145 ids_outliers_f <- c("1676574", "4051825", "3418862", "2819257", "9856662", "4614642",
1146 "6715534", "5446016")
1147
1148 bd_civil_modis <- bd_civil_modis |>
1149     mutate(
1150       Fecha_entrada_vigor = if_else(
1151         Identificador %in% ids_outliers_f,
1152         Fecha_presentacion,
1153         Fecha_entrada_vigor
1154       )
1155     )
1156
1157 bd_civil_sin_NA <- bd_civil_sin_NA |>
1158     mutate(
1159       Fecha_entrada_vigor = if_else(
1160         Identificador %in% ids_outliers_f,

```

```

1161     Fecha_presentacion,
1162     Fecha_entrada_vigor
1163 )
1164 )
1165
1166 # --- Cálculo de intervalos temporales ---
1167 bd_civil_modis <- bd_civil_modis |>
1168 mutate(
1169     Intervalo_licid = as.integer(Fecha_presentacion - Primera_publicacion),
1170     Intervalo_vigor_d = as.integer(Fecha_entrada_vigor - Fecha_presentacion),
1171     Intervalo_urge_d = as.integer(Fecha_entrada_vigor - Primera_publicacion),
1172     Intervalo_modi_m = round(as.numeric(difftime(Fecha_actualizacion,
1173         Fecha_entrada_vigor, units = "days"))) / 30)
1174 )
1175
1176 bd_civil_sin_NA <- bd_civil_sin_NA |>
1177 mutate(
1178     Intervalo_licid = as.integer(Fecha_presentacion - Primera_publicacion),
1179     Intervalo_vigor_d = as.integer(Fecha_entrada_vigor - Fecha_presentacion),
1180     Intervalo_urge_d = as.integer(Fecha_entrada_vigor - Primera_publicacion),
1181     Intervalo_modi_m = round(as.numeric(difftime(Fecha_actualizacion,
1182         Fecha_entrada_vigor, units = "days"))) / 30)
1183 )
1184
1185 ##### GENERAMOS DATA FRAME DE RELACION OBRA-REDACCION ##### -----
1186
1187 ##### ANÁLISIS SEMÁNTICO DE PARES OBRA-REDACCIÓN (QUANTEDA) ##### -----
1188 library(quanteda)
1189 library(readxl)
1190 library(dplyr)
1191 library(stringr)
1192 library(stringi)
1193
1194 # 1. Leer archivo de coincidencias TF-IDF
1195 coincidencias_tfidf <- read_excel("C:/Users/guillermo.alonso/Desktop/Tesis
Guillermo/_BD actualizada/Datos atom/PLACSP/coincidencias_modis_tfidf_ampliado.xlsx")
1196
1197 # 2. Filtrar coincidencias con score alto (> 0.75)
1198 coincidencias_filtradas <- coincidencias_tfidf |>
1199     filter(precision_score > 0.75)
1200
1201 # 3. Seleccionar mejor par por Identificador_modis
1202 coincidencias_filtradas <- coincidencias_filtradas |>
1203     group_by(Identificador_modis) |>
1204     slice_max(order_by = precision_score, n = 1, with_ties = FALSE) |>
1205     ungroup()
1206
1207 # 4. Seleccionar mejor par por Identificador_redac (resultado: pares únicos)
1208 coincidencias_filtradas <- coincidencias_filtradas |>
1209     group_by(Identificador_redac) |>
1210     slice_max(order_by = precision_score, n = 1, with_ties = FALSE) |>
1211     ungroup()
1212
1213 # 5. Guardamos el resultado inicial
1214 bd_ingeobra_ampliado <- coincidencias_filtradas
1215
1216 # --- NORMALIZACIÓN DE TEXTO Y TOKENIZACIÓN ROBUSTA ---
1217 # Limpieza: eliminar tildes y pasar a minúsculas
1218 bd_ingeobra_ampliado <- bd_ingeobra_ampliado |>
1219     mutate(Objeto_redac_clean = stri_trans_general(Objeto_redac, "Latin-ASCII"))
1220
1221 # Crear tokens limpios con stemming
1222 tokens_redac <- tokens(
1223     bd_ingeobra_ampliado$Objeto_redac_clean,
1224     remove_punct = TRUE,
1225     remove_symbols = TRUE,
1226     remove_numbers = TRUE
1227 ) |>
1228     tokens_tolower() |>
1229     tokens_wordstem(language = "spanish")
1230

```

```

1231 # --- 1. Expresiones clave nuevas (frases comunes en redacción de proyecto) ---
1232 multi_expresiones <- list(
1233   c("direccion", "facultativa"),
1234   c("direccion", "obra"),
1235   c("proyecto", "constructivo"),
1236   c("proyecto", "ejecutivo"),
1237   c("proyecto", "detalle"),
1238   c("plan", "seguridad"),
1239   c("coordinacion", "seguridad"),
1240   c("estudio", "soluciones"),
1241   c("ingenieria", "detalle"),
1242   c("elaboracion", "proyecto"),
1243   c("asistencia", "tecnica"),
1244   c("servicio", "elaboracion"),
1245   c("elaboracion", "integral")
1246 )
1247
1248 tokens_redac <- tokens_compound(tokens_redac, phrase(multi_expresiones))
1249
1250 # --- FILTRADO DE CASOS CON CONTENIDO DE REDACCIÓN ---
1251 # Definir términos básicos de redacción
1252 terminos_redaccion <- c(
1253   "redaccion", "diseno", "elaboracion", "anteproyecto", "proyecto",
1254   "tecnico", "estudio", "ingenieria", "planificacion",
1255   "memoria", "constructiva", "basica", "ejecutiva"
1256 )
1257
1258 # Crear matriz documento-característica
1259 dfm_redac <- dfm(tokens_redac)
1260
1261 # Detectar si hay contenido clave de redacción
1262 dfm_filtro_redac <- dfm_select(dfm_redac, pattern = terminos_redaccion)
1263 filtro_redaccion <- rowSums(dfm_filtro_redac) > 0
1264
1265 # Separar casos
1266 inge_obra_redaccion <- bd_inge_obra_ampliado[filtro_redaccion, ]
1267 inge_obra_no_redaccion <- bd_inge_obra_ampliado[!filtro_redaccion, ]
1268
1269 # --- ANÁLISIS SEMÁNTICO: CATEGORIZACIÓN DE TEMAS ---
1270 # --- . Diccionario mejorado con términos expandidos y compuestos
1271 diccionario_temas <- dictionary(list(
1272   anteproyecto = c("anteproyecto", "preliminar", "viabilidad", "alternativas",
1273     "solucion"),
1274
1275   redaccion_proyecto = c(
1276     "redaccion", "memoria", "documentacion", "constructivo", "detalle", "ejecutivo",
1277     "ingenieria", "proyecto", "elaboracion", "asistencia tecnica", "elaboracion
1278     proyecto",
1279     "elaboracion integral", "servicio elaboracion", "proyecto tecnico"
1280   ),
1281
1282   direccion_facultativa = c("direccion facultativa", "direccion obra", "supervision",
1283     "control", "coordinacion"),
1284
1285   seguridad_salud = c("seguridad", "salud", "riesgo", "plan seguridad", "prevencion"),
1286
1287   construccion = c("ejecucion", "realizacion", "construccion", "obra civil")
1288 ))
1289 # Aplicar diccionario temático
1290 dfm_temas <- dfm_lookup(dfm_redac[filtro_redaccion, ], dictionary = diccionario_temas)
1291
1292 # Convertir a binario: 1 si contiene el tema, 0 si no
1293 temas_binarias <- convert(dfm_temas, to = "data.frame") |>
1294   dplyr::select(-doc_id) |>
1295   mutate(across(everything(), ~ ifelse(. > 0, 1, 0)))
1296
1297 # Añadir columnas temáticas al data frame
1298 inge_obra_redaccion <- bind_cols(inge_obra_redaccion, temas_binarias)
1299
1300 inge_obra_redaccion <- inge_obra_redaccion |>
1301   dplyr::select(-Objeto_redac_clean)
1302
1303 ## JUNTAMOS CON LOS DATOS QUE TENEMOS -----

```

```

1301
1302 # --- 1. Seleccionar columnas clave de temas y scores ---
1303 base_pares <- inge_obra_redaccion |>
1304   dplyr::select(
1305     precision_score,
1306     anteproyecto,
1307     redaccion_proyecto,
1308     direccion_facultativa,
1309     seguridad_salud,
1310     construccion,
1311     Identificador_redac,
1312     Identificador_modis
1313   ) |>
1314   mutate(
1315     Identificador_redac = as.character(Identificador_redac),
1316     Identificador_modis = as.character(Identificador_modis)
1317   )
1318
1319 # --- 2. Traer info de Identificador_redac desde bd_placsp_ingepares ---
1320 info_redac <- bd_placsp_ingepares |>
1321   filter(Identificador %in% base_pares$Identificador_redac) |>
1322   group_by(Identificador) |>
1323   slice_max(order_by = Fecha_actualizacion, n = 1, with_ties = FALSE) |>
1324   ungroup()
1325
1326 info_redac <- info_redac |>
1327   rename_with(~ paste0("redac_", .x), .cols = -Identificador)
1328
1329 # --- 3. Traer info de Identificador_modis desde bd_civil_modis ---
1330 info_modis <- bd_civil_modis |>
1331   filter(Identificador %in% base_pares$Identificador_modis) |>
1332   group_by(Identificador) |>
1333   slice_max(order_by = Fecha_actualizacion, n = 1, with_ties = FALSE) |>
1334   ungroup()
1335
1336 info_modis <- info_modis |>
1337   rename_with(~ paste0("modis_", .x), .cols = -Identificador)
1338
1339 # --- 4. Join completo con claves convertidas ---
1340 bd_inge_obra_pares <- base_pares |>
1341   left_join(info_redac, by = c("Identificador_redac" = "Identificador")) |>
1342   left_join(info_modis, by = c("Identificador_modis" = "Identificador"))
1343
1344
1345
1346 ##### DETECCION DE OUTLIERS #####
1347
1348 ## Por percentiles extremos
1349 vars_percentiles <- c("Presupuesto_licitacion", "Modi_temp_m", "Modi_eco_e",
1350   "Intervalo_vigor_d", "Baja_p", "Precio_final_e")
1351
1352 for (var in vars_percentiles) {
1353   q_low <- quantile(bd_civil_sin_NA[[var]], 0.005, na.rm = TRUE)
1354   q_high <- quantile(bd_civil_sin_NA[[var]], 0.995, na.rm = TRUE)
1355
1356   bd_civil_sin_NA <- bd_civil_sin_NA |>
1357     filter(.data[[var]] >= q_low & .data[[var]] <= q_high | is.na(.data[[var]]))
1358 }
1359
1360 ## Casos Manuales
1361 bd_civil_sin_NA <- bd_civil_sin_NA |>
1362   filter(
1363     Presupuesto_licitacion != 0 | is.na(Presupuesto_licitacion),
1364     Modi_temp_m >= -12 | is.na(Modi_temp_m),
1365     Modi_eco_p >= -80 | is.na(Modi_eco_p),
1366     Intervalo_vigor_d >= 0 | is.na(Intervalo_vigor_d),
1367     Intervalo_modi_m >= 0 | is.na(Intervalo_modi_m)
1368   )
1369
1370 for (var in vars_percentiles) {
1371   q_low <- quantile(bd_civil_modis[[var]], 0.005, na.rm = TRUE)
1372   q_high <- quantile(bd_civil_modis[[var]], 0.995, na.rm = TRUE)
1373

```

```

1374     bd_civil_modis <- bd_civil_modis |>
1375     filter(.data[[var]] >= q_low & .data[[var]] <= q_high | is.na(.data[[var]]))
1376   }
1377
1378   ## Casos Manuales
1379   bd_civil_modis <- bd_civil_modis |>
1380   filter(
1381     Presupuesto_licitacion != 0 | is.na(Presupuesto_licitacion),
1382     Modi_temp_m >= -12 | is.na(Modi_temp_m),
1383     Modi_eco_p >= -80 | is.na(Modi_eco_p),
1384     Intervalo_vigor_d >= 0 | is.na(Intervalo_vigor_d),
1385     Intervalo_modi_m >= 0 | is.na(Intervalo_modi_m)
1386   )
1387
1388   ## Eliminación manual de casos concretos:
1389
1390   # --- Eliminación por Identificador manual ---
1391   ids_excluir <- c("4051825", "1676574", "3418862", "2819257", "6135224",
1392     "4111947", "11544946", "6508881", "10373168", "10614482")
1393
1394   bd_civil_sin_NA <- bd_civil_sin_NA[!bd_civil_sin_NA$Identificador %in% ids_excluir, ]
1395
1396   # --- Forzar a NA los casos con Modi_temp_m > 60 ---
1397   bd_civil_sin_NA <- bd_civil_sin_NA |>
1398   mutate(
1399     Modi_temp_m = ifelse(Modi_temp_m > 60, NA_integer_, Modi_temp_m),
1400     Plazo_final_m = ifelse(is.na(Modi_temp_m), NA_integer_, Plazo_final_m),
1401     Modi_temp_p = ifelse(is.na(Modi_temp_m), NA_real_, Modi_temp_p)
1402   )
1403
1404   bd_civil_modis <- bd_civil_modis[!bd_civil_modis$Identificador %in% ids_excluir, ]
1405
1406   # --- Forzar a NA los casos con Modi_temp_m > 60 ---
1407   bd_civil_modis <- bd_civil_modis |>
1408   mutate(
1409     Modi_temp_m = ifelse(Modi_temp_m > 60, NA_integer_, Modi_temp_m),
1410     Plazo_final_m = ifelse(is.na(Modi_temp_m), NA_integer_, Plazo_final_m),
1411     Modi_temp_p = ifelse(is.na(Modi_temp_m), NA_real_, Modi_temp_p)
1412   )
1413
1414
1415   ##### GUARDAMOS RESULTADOS -----
1416
1417   ## Directorio GUARDAR ----
1418
1419   setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
1420   atom/PLACSP/_final_conjunto_v4")
1421
1422   # Exportar bd_civil_todo
1423   save(bd_civil_todo, file = "bd_civil_todo_v2.RData")
1424   write.xlsx(bd_civil_todo, "bd_civil_todo_v2.xlsx", colNames = TRUE, rowNames = FALSE)
1425
1426   # Exportar bd_civil_modis
1427   save(bd_civil_modis, file = "bd_civil_modis_v2.RData")
1428   write.xlsx(bd_civil_modis, "bd_civil_modis_v2.xlsx", colNames = TRUE, rowNames = FALSE)
1429
1430   # Exportar bd_placsp_ingecivil
1431   save(bd_placsp_ingecivil, file = "bd_placsp_ingecivil_v2.RData")
1432   write.xlsx(bd_placsp_ingecivil, "bd_placsp_ingecivil_v2.xlsx", colNames = TRUE,
1433     rowNames = FALSE)
1434
1435   # Exportar bd_placsp_ingepares
1436   save(bd_placsp_ingepares, file = "bd_placsp_ingepares_v2.RData")
1437   write.xlsx(bd_placsp_ingepares, "bd_placsp_ingepares_v2.xlsx", colNames = TRUE,
1438     rowNames = FALSE)
1439
1440   # Exportar bd_civil_sin_NA
1441   save(bd_civil_sin_NA, file = "bd_civil_sin_NA_v2.RData")
1442   write.xlsx(bd_civil_sin_NA, "bd_civil_sin_NA_v2.xlsx", colNames = TRUE, rowNames =
1443     FALSE)

```

```

1442 # Guardar en Excel
1443 save(bd_ingeobra_pares, file = "bd_ingeobra_pares_v2.RData")
1444 write.xlsx(bd_ingeobra_pares, "bd_ingeobra_pares_v2.xlsx", colNames = TRUE, rowNames
  = FALSE)

1445
1446
1447
1448 # Exportar todo el entorno global
1449 save.image(file = "_Final_v2_entorno_global_completo_v2.RData")
1450
1451 # Limpiamos objetos
1452
1453 rm(list=ls())
1454
1455
1456 ### SCRIPT DE SOM_v1 -----
1457 # En los scripts de regresion y PCA v1 se han hecho los mismos o menos tratamientos
  de datos que aqui
1458 # Se empieza de cero cargando el data frame base
1459 # Se introduce al guardar las variables Identificador y Identificado_lote
1460 ### -----
1461
1462 ## Directorio LEER v1 ----
1463
1464 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
  atom/PLACSP/_final_conjunto_v4")
1465
1466 ## Cargamos datos -----
1467
1468 load("bd_civil_sin_NA_v2.RData")
1469
1470 ## Directorio GUARDAR ----
1471
1472 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
  atom/PLACSP/_final_conjunto_v4")
1473
1474
1475 ## PREPROCESAMIENTO de VARIABLES ##### -----
1476
1477 # Aseguramos que estamos trabajando sobre la base correcta
1478 df <- bd_civil_sin_NA
1479
1480 # Convertir factores de interés
1481 df <- df |>
1482   mutate(
1483     Tipo_de_Administracion = as.factor(Tipo_de_Administracion),
1484     Tipo_de_procedimiento = as.factor(Tipo_de_procedimiento),
1485     Tipo_de_contrato = as.factor(Tipo_de_contrato),
1486     Tramitacion = as.factor(Tramitacion),
1487     Tipo_ganador_lote = as.factor(Tipo_ganador_lote),
1488     Modis_s_n = as.factor(Modis_s_n)
1489   )
1490
1491
1492 # --- Tratamiento de la variable CPV en df ---
1493 df <- df |>
1494   mutate(
1495     CPV_trunks = purrr::map(CPV, function(x) {
1496       if (is.na(x) || stringr::str_trim(x) == "") return(character(0))
1497
1498       x_split <- stringr::str_split(x, ":", simplify = FALSE)[[1]]
1499       x_trimmed <- stringr::str_trim(x_split)
1500       x_valid <- x_trimmed[x_trimmed != ""]
1501       stringr::str_sub(x_valid, 1, 5)
1502     }),
1503
1504     N_CPV = purrr::map_int(CPV_trunks, length),
1505
1506     CPV_main = purrr::map_chr(CPV_trunks, function(cpvs) {
1507       if (length(cpvs) == 0) return(NA_character_)
1508
1509       cpvs_45 <- cpvs[stringr::str_starts(cpvs, "45")]
1510       if (length(cpvs_45) == 0) return(NA_character_)

```

```

1511     ceros_derecha <- stringr::str_length(cpvs_45) -
1512     stringr::str_length(stringr::str_replace(cpvs_45, "0*$", ""))
1513
1514
1515     min_ceros <- min(ceros_derecha)
1516     seleccionados <- cpvs_45[ceros_derecha == min_ceros]
1517
1518     return(seleccionados[1])
1519   })
1520 )
1521
1522
1523 # --- Tratamiento de la variable Clasi_empresa sobre el data frame df ---
1524
1525 # 1. Calcular N_clasi_empresa: número de clasificaciones registradas (separadas por
1526 #:")
1527 df <- df |>
1528   mutate(
1529     N_clasi_empresa = if_else(
1530       is.na(Clasi_empresa) | str_trim(Clasi_empresa) == "",
1531       0L,
1532       str_count(Clasi_empresa, ":") + 1L
1533     )
1534   )
1535
1536 # 2. Función para extraer la letra con el número más alto (o la primera si hay empate)
1537 extraer_letra_dominante <- function(clasi_string) {
1538   if (is.na(clasi_string) || str_trim(clasi_string) == "") {
1539     return("0") # Regla: si no hay clasificaciones, devolver "0"
1540   }
1541
1542   # Separar clasificaciones por ":"
1543   clasificaciones <- unlist(str_split(clasi_string, ":", simplify = FALSE))
1544
1545   # Extraer letra y número si está, o solo letra si número no es válido
1546   letras_valores <- str_match(clasificaciones, "^[A-Z])[^0-9]*([0-9]*)")
1547
1548   # Filtrar filas válidas
1549   letras <- letras_valores[, 2]
1550   numeros <- suppressWarnings(as.numeric(letras_valores[, 3]))
1551
1552   # Si no hay letras válidas, devolver "0"
1553   if (all(is.na(letras))) {
1554     return("0")
1555   }
1556
1557   # En caso de números faltantes (NA), asumir valor 0
1558   numeros[is.na(numeros)] <- 0
1559
1560   # Agrupar por letra y conservar el máximo valor
1561   resumen <- aggregate(numeros, by = list(letras), FUN = max)
1562   colnames(resumen) <- c("letra", "max_valor")
1563
1564   # Escoger la letra con mayor valor, priorizando la primera en caso de empate
1565   maximo <- max(resumen$max_valor, na.rm = TRUE)
1566   letras_max <- resumen$letra[resumen$max_valor == maximo]
1567
1568   return(letras_max[1])
1569 }
1570
1571 # 3. Aplicar función y ajustar según la regla de N_clasi_empresa
1572 df <- df |>
1573   mutate(
1574     Clasi_empresa_main = vapply(Clasi_empresa, extraer_letra_dominante, character(1)),
1575     Clasi_empresa_main = if_else(N_clasi_empresa == 0, "0", Clasi_empresa_main)
1576   )
1577
1578 # --- Tratamiento de la variable Requi_adm: contar requisitos administrativos ---
1579
1580 df <- df |>
1581   mutate(
1582     # Si Requi_adm es NA o está vacía, asignar 0; si no, contar los elementos
1583     # separados por ":"

```

```

1582     N_requi = if_else(
1583       is.na(Requi_adm) | str_trim(Requi_adm) == "",
1584       0L,
1585       str_count(Requi_adm, ":") + 1L
1586     )
1587   )
1588
1589   # --- Tratamiento de la variable Prorroga ---
1590
1591
1592   df <- df |>
1593     mutate(
1594       Prorroga = case_when(
1595         is.na(Prorroga) ~ "No", # Si está vacía → No
1596         str_detect(tolower(Prorroga), "no|sin|ninguna|ningun|ningún|n\\.a") ~ "No", #
1597         # Si contiene negaciones comunes → No
1598         TRUE ~ "Sí" # En cualquier otro caso → Sí
1599       ),
1600       Prorroga = factor(Prorroga, levels = c("No", "Sí"))
1601     )
1602
1603   # --- Tratamiento de la variable N_lotes ---
1604
1605   df <- df |>
1606     mutate(
1607       # Sustituimos "Sin lotes" por 1
1608       N_lotes = if_else(N_lotes == "Sin lotes", "1", N_lotes),
1609
1610       # Convertimos a entero
1611       N_lotes = as.integer(N_lotes)
1612     )
1613
1614   # --- Tratamiento de la variable Clasi_max_req ---
1615   df$Clasi_max_req <- ifelse(is.na(df$Clasi_max_req), 0, df$Clasi_max_req)
1616
1617   # --- Selección de variables para modelado predictivo ---
1618
1619   # Lista de variables seleccionadas para el modelo
1620   vars_modelo <- c(
1621     "Identificador",
1622     "Identificador_lote",
1623     "Precio_adjudicacion",
1624     "N_ofertantes",
1625     "N_lotes",
1626     "Presupuesto_licitacion",
1627     "Tipo_de_contrato",
1628     "Tipo_de_Administracion",
1629     "Tipo_de_procedimiento",
1630     "Tramitacion",
1631     "Tipo_ganador_lote",
1632     "Plazo_m",
1633     #"Prorroga",
1634     "C_precio_p",
1635     "C_resto_objetivos_p",
1636     "C_juicio_valor_p",
1637     "Baja_p",
1638     "Modi_eco_p",
1639     "Clasi_max_req",
1640     "Intervalo_lici_d",
1641     "Intervalo_vigor_d",
1642     "Intervalo_urge_d",
1643     "N_CPV",
1644     "N_clasi_empresa",
1645     "Clasi_empresa_main",
1646     "Codigo_Postal", #-- Lo sacamos
1647     "CPV_main",
1648     "N_requi"
1649   )
1650
1651   # Filtramos el data frame conservando solo esas variables
1652   df <- df |>
1653     dplyr::select(all_of(vars_modelo))

```



```

1654
1655
1656 ## Generamos nuevas variables categoricas a booleanas
1657 df <- df |>
1658   mutate(
1659     Tipo_de_contrato_Obras = if_else(Tipo_de_contrato == "Obras", 1L, 0L),
1660     Tipo_de_Administracion_Autoridad_local = if_else(Tipo_de_Administracion ==
1661       "Autoridad local", 1L, 0L),
1662     Tipo_de_procedimiento_Abierto = if_else(Tipo_de_procedimiento == "Abierto", 1L, 0L
1663   ),
1664     Tramitacion_Ordinaria = if_else(Tramitacion == "Ordinaria", 1L, 0L),
1665     Tipo_ganador_lote_NIF = if_else(Tipo_ganador_lote == "NIF", 1L, 0L)
1666   )
1667 ##Variables categóricas: aseguramos que estén como factor
1668 df$CPV_main <- factor(df$CPV_main)
1669 df$Clasi_empresa_main <- factor(df$Clasi_empresa_main)
1670 df$Codigo_Postal <- factor(df$Codigo_Postal)
1671
1672 # Eliminar variables con muchos NAs o crear versión limpia si es necesario
1673 df_model <- df |>
1674   dplyr::select(
1675     "Identificador",
1676     "Identificador_lote",
1677     "Precio_adjudicacion",
1678     "N_ofertantes",
1679     "N_lotes",
1680     "Presupuesto_licitacion",
1681     #"Tipo_de_contrato",
1682     "Tipo_de_Administracion",
1683     "Tipo_de_procedimiento",
1684     #"Tramitacion",
1685     #"Tipo_ganador_lote",
1686     "Tipo_de_contrato_Obras",
1687     #"Tipo_de_Administracion_Autoridad_local",
1688     #"Tipo_de_procedimiento_Abierto",
1689     "Tramitacion_Ordinaria",
1690     "Tipo_ganador_lote_NIF",
1691     "Plazo_m",
1692     #"Prorroga",
1693     "C_precio_p",
1694     "C_resto_objetivos_p",
1695     "C_juicio_valor_p",
1696     "Baja_p",
1697     "Modi_eco_p",
1698     "Clasi_max_req",
1699     "Intervalo_licid",
1700     "Intervalo_vigor_d",
1701     "Intervalo_urge_d",
1702     "N_CPV",
1703     "N_clasi_empresa",
1704     "Clasi_empresa_main",
1705     "Codigo_Postal", #-- Lo sacamos
1706     "CPV_main",
1707     "N_requi"
1708   ) |>
1709   na.omit() # Quita NAs para el modelo base
1710
1711 ### DIVISIÓN DE DATOS 80/20 ALEATORIA ----
1712 set.seed(321)
1713 n <- nrow(df_model)
1714 train_idx <- sample(seq_len(n), size = 0.8 * n)
1715 df_train <- df_model[train_idx, ]
1716 df_test <- df_model[-train_idx, ]
1717 save(df_train, df_test, file="train_test_v1_g.RData")
1718 write.xlsx(df_train, "df_train_v1_g.xlsx", colNames = TRUE, rowNames = FALSE)
1719 write.xlsx(df_test, "df_test_v1_g.xlsx", colNames = TRUE, rowNames = FALSE)
1720
1721 # Borramos
1722
1723 rm(list=ls())
1724

```

```

1725 ## -----
1726 ## SCRIPT 3 PCA_v2 ## -----
1727 ## -----
1728
1729 ## Directorio LEER v2 ----
1730
1731 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
1732
1733 ## Cargamos datos -----
1734
1735 load("bd_civil_sin_NA_v2.RData")
1736
1737 ## Directorio GUARDAR ----
1738
1739 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
1740
1741 ## PREPROCESAMIENTO de VARIABLES ##### -----
1742
1743 # Aseguramos que estamos trabajando sobre la base correcta
1744 df <- bd_civil_sin_NA
1745
1746 # Convertir factores de interés
1747 df <- df |>
1748   mutate(
1749     Tipo_de_Administracion = as.factor(Tipo_de_Administracion),
1750     Tipo_de_procedimiento = as.factor(Tipo_de_procedimiento),
1751     Tipo_de_contrato = as.factor(Tipo_de_contrato),
1752     Tramitacion = as.factor(Tramitacion),
1753     Tipo_ganador_lote = as.factor(Tipo_ganador_lote),
1754     Modis_s_n = as.factor(Modis_s_n)
1755   )
1756
1757
1758 # --- Tratamiento de la variable CPV en df ---
1759 df <- df |>
1760   mutate(
1761     CPV_truncs = purrr::map(CPV, function(x) {
1762       if (is.na(x) || stringr::str_trim(x) == "") return(character(0))
1763
1764       x_split <- stringr::str_split(x, ":", simplify = FALSE)[[1]]
1765       x_trimmed <- stringr::str_trim(x_split)
1766       x_valid <- x_trimmed[x_trimmed != ""]
1767       stringr::str_sub(x_valid, 1, 5)
1768     }),
1769
1770     N_CPV = purrr::map_int(CPV_truncs, length),
1771
1772     CPV_main = purrr::map_chr(CPV_truncs, function(cpvs) {
1773       if (length(cpvs) == 0) return(NA_character_)
1774
1775       cpvs_45 <- cpvs[stringr::str_starts(cpvs, "45")]
1776       if (length(cpvs_45) == 0) return(NA_character_)
1777
1778       ceros_derecha <- stringr::str_length(cpvs_45) -
1779         stringr::str_length(stringr::str_replace(cpvs_45, "0*$", ""))
1780
1781       min_ceros <- min(ceros_derecha)
1782       seleccionados <- cpvs_45[ceros_derecha == min_ceros]
1783
1784       return(seleccionados[1])
1785     })
1786   )
1787
1788
1789 # --- Tratamiento de la variable Clasi_empresa sobre el data frame df ---
1790
1791 # 1. Calcular N_clasi_empresa: número de clasificaciones registradas (separadas por
":")
1792 df <- df |>
1793   mutate(
1794     N_clasi_empresa = if_else(

```

```

1795     is.na(Clasi_empresa) | str_trim(Clasi_empresa) == "",
1796     0L,
1797     str_count(Clasi_empresa, ":") + 1L
1798   )
1799 )
1800
1801 # 2. Función para extraer la letra con el número más alto (o la primera si hay empate)
1802 extraer_letra_dominante <- function(clasi_string) {
1803   if (is.na(clasi_string) || str_trim(clasi_string) == "") {
1804     return("0") # Regla: si no hay clasificaciones, devolver "0"
1805   }
1806
1807   # Separar clasificaciones por ":"
1808   clasificaciones <- unlist(str_split(clasi_string, ":", simplify = FALSE))
1809
1810   # Extraer letra y número si está, o solo letra si número no es válido
1811   letras_valores <- str_match(clasificaciones, "^[A-Z])[^0-9]*([0-9]*)")
1812
1813   # Filtrar filas válidas
1814   letras <- letras_valores[, 2]
1815   numeros <- suppressWarnings(as.numeric(letras_valores[, 3]))
1816
1817   # Si no hay letras válidas, devolver "0"
1818   if (all(is.na(letras))) {
1819     return("0")
1820   }
1821
1822   # En caso de números faltantes (NA), asumir valor 0
1823   numeros[is.na(numeros)] <- 0
1824
1825   # Agrupar por letra y conservar el máximo valor
1826   resumen <- aggregate(numeros, by = list(letras), FUN = max)
1827   colnames(resumen) <- c("letra", "max_valor")
1828
1829   # Escoger la letra con mayor valor, priorizando la primera en caso de empate
1830   maximo <- max(resumen$max_valor, na.rm = TRUE)
1831   letras_max <- resumen$letra[resumen$max_valor == maximo]
1832
1833   return(letras_max[1])
1834 }
1835
1836 # 3. Aplicar función y ajustar según la regla de N_clasi_empresa
1837 df <- df |>
1838   mutate(
1839     Clasi_empresa_main = vapply(Clasi_empresa, extraer_letra_dominante, character(1)),
1840     Clasi_empresa_main = if_else(N_clasi_empresa == 0, "0", Clasi_empresa_main)
1841   )
1842
1843 # --- Tratamiento de la variable Requi_adm: contar requisitos administrativos ---
1844
1845 df <- df |>
1846   mutate(
1847     # Si Requi_adm es NA o está vacía, asignar 0; si no, contar los elementos
1848     # separados por ":"
1849     N_requi = if_else(
1850       is.na(Requi_adm) | str_trim(Requi_adm) == "",
1851       0L,
1852       str_count(Requi_adm, ":") + 1L
1853     )
1854   )
1855
1856 # --- Tratamiento de la variable Prorroga ---
1857
1858 df <- df |>
1859   mutate(
1860     Prorroga = case_when(
1861       is.na(Prorroga) ~ "No", # Si está vacía → No
1862       str_detect(tolower(Prorroga), "no|sin|ninguna|ningun|ningún|n\\.a") ~ "No", #
1863       # Si contiene negaciones comunes → No
1864       TRUE ~ "Sí" # En cualquier otro caso → Sí
1865     ),
1866     Prorroga = factor(Prorroga, levels = c("No", "Sí"))

```

```

1866     )
1867
1868 # --- Tratamiento de la variable N_lotes ---
1869
1870 df <- df |>
1871   mutate(
1872     # Sustituimos "Sin lotes" por 1
1873     N_lotes = if_else(N_lotes == "Sin lotes", "1", N_lotes),
1874
1875     # Convertimos a entero
1876     N_lotes = as.integer(N_lotes)
1877   )
1878
1879 # --- Tratamiento de la variable Clasi_max_req ---
1880 df$Clasi_max_req <- ifelse(is.na(df$Clasi_max_req), 0, df$Clasi_max_req)
1881
1882
1883 # --- Selección inicial de variables para modelado predictivo ---
1884
1885 # Lista de variables seleccionadas para el modelo
1886 vars_modelo <- c(
1887   "Identificador",
1888   "Identificador_lote",
1889   "Precio_adjudicacion",
1890   "N_ofertantes",
1891   "N_lotes",
1892   "Presupuesto_licitacion",
1893   "Tipo_de_contrato",
1894   "Tipo_de_Administracion",
1895   "Tipo_de_procedimiento",
1896   "Tramitacion",
1897   "Tipo_ganador_lote",
1898   "Plazo_m",
1899   #"Prorroga",
1900   "C_precio_p",
1901   "C_resto_objetivos_p",
1902   "C_juicio_valor_p",
1903   "Baja_p",
1904   "Modi_eco_e", ## Para poder hacer limpieza de casos
1905   "Modi_eco_p",
1906   "Modi_temp_p", ## Para poder hacer limpieza de casos
1907   "Clasi_max_req",
1908   "Intervalo_lici_d",
1909   "Intervalo_vigor_d",
1910   "Intervalo_urge_d",
1911   "N_CPV",
1912   "N_clasi_empresa",
1913   "Clasi_empresa_main",
1914   "Codigo_Postal", #-- Lo sacamos
1915   "CPV_main",
1916   "N_requi"
1917 )
1918
1919 # Filtramos el data frame conservando solo esas variables
1920 df <- df |>
1921   dplyr::select(all_of(vars_modelo))
1922
1923 # --- Transformacion de variables para modelado predictivo ---
1924 ## Generamos nuevas variables categoricas a booleanas
1925 df <- df |>
1926   mutate(
1927     Tipo_de_contrato_Obras = if_else(Tipo_de_contrato == "Obras", 1L, 0L),
1928     Tipo_de_Administracion_Autoridad_local = if_else(Tipo_de_Administracion ==
1929       "Autoridad local", 1L, 0L),
1930     Tipo_de_procedimiento_Abierto = if_else(Tipo_de_procedimiento == "Abierto", 1L, 0L
1931       ),
1932     Tipo_de_procedimiento_Simplificado = if_else(Tipo_de_procedimiento == "Abierto
1933       simplificado", 1L, 0L),
1934     Tramitacion_Ordinaria = if_else(Tramitacion == "Ordinaria", 1L, 0L),
1935     Tipo_ganador_lote_NIF = if_else(Tipo_ganador_lote == "NIF", 1L, 0L)
1936   )
1937
1938 ##Variables categóricas: aseguramos que estén como factor
1939 df$CPV_main <- factor(df$CPV_main)

```

```

1936 df$Clasi_empresa_main <- factor(df$Clasi_empresa_main)
1937 df$Codigo_Postal <- factor(df$Codigo_Postal)
1938
1939
1940 # Seleccion para transformacion de variables ---
1941 df_model <- df |>
1942   dplyr::select(
1943     "Identificador",
1944     "Identificador_lote",
1945     "Precio_adjudicacion",
1946     "N_ofertantes",
1947     "N_lotes",
1948     #"Presupuesto_licitacion",
1949     #"Tipo_de_contrato",
1950     #"Tipo_de_Administracion",
1951     #"Tipo_de_procedimiento",
1952     #"Tramitacion",
1953     #"Tipo_ganador_lote",
1954     "Tipo_de_contrato_Obras",
1955     "Tipo_de_Administracion_Autoridad_local",
1956     "Tipo_de_procedimiento_Abierto",
1957     "Tipo_de_procedimiento_Simplificado",
1958     "Tramitacion_Ordinaria",
1959     "Tipo_ganador_lote_NIF",
1960     "Plazo_m",
1961     #"Prorroga",
1962     "C_precio_p",
1963     #"C_resto_objetivos_p",
1964     #"C_juicio_valor_p",
1965     "Baja_p",
1966     "Modi_eco_e", ## Para poder hacer limpieza de casos
1967     "Modi_eco_p",
1968     "Modi_temp_p", ## Para poder hacer limpieza de casos
1969     "Clasi_max_req",
1970     #"Intervalo_lici_d",
1971     #"Intervalo_vigor_d",
1972     "Intervalo_urge_d",
1973     "N_CPV",
1974     "N_clasi_empresa",
1975     "Clasi_empresa_main",
1976     #"Codigo_Postal", #-- Lo sacamos
1977     #"CPV_main",
1978     "N_requi"
1979   )
1980
1981
1982
1983 ### NUEVO FILTRO DE MODIS ECO 0 ####
1984
1985 # Contar casos que cumplen la condición de eliminación (Modi eco y temp 0)
1986 casos_eliminar <- df_model |>
1987   filter(
1988     Modi_eco_p == 0 &
1989     (is.na(Modi_temp_p) | Modi_temp_p == 0)
1990   )
1991
1992 n_eliminados <- nrow(casos_eliminar)
1993
1994 # Crear nuevo data frame excluyendo esos casos
1995 df_model_v2 <- df_model |>
1996   filter(
1997     !(Modi_eco_p == 0 &
1998       (is.na(Modi_temp_p) | Modi_temp_p == 0))
1999   )
2000
2001 # Mostrar el número de casos eliminados
2002 cat("Número de casos eliminados:", n_eliminados, "\n")
2003
2004 # Crear variable categórica Modi_cat a partir de Modi_eco_p según rangos definidos
2005 df_model_v2 <- df_model_v2 |>
2006   mutate(
2007     Modi_cat = cut(
2008       Modi_eco_p,

```

```

2009     breaks = c(-Inf, 0, 5, 9, 10, 14, 15, 19, 20, 35, 49, 50, Inf),
2010     labels = 0:11,
2011     right = TRUE,
2012     include.lowest = FALSE
2013   ),
2014   Modi_cat = as.integer(as.character(Modi_cat)) # Convertir a entero
2015 )
2016
2017 # Etiquetas legibles para cada categoría de Modi_cat
2018 modi_cat_labels <- c(
2019   "(-Inf, 0]", "(0,5]", "(5,9]", "(9,10]", "(10,14]", "(14,15]",
2020   "(15,19]", "(19,20]", "(20,35]", "(35,49]", "(49,50]", "(50, Inf]"
2021 )
2022
2023
2024
2025
2026 # Seleccion final de variables para entrenamiento ---
2027 df_model_v2 <- df_model_v2 |>
2028   dplyr::select(
2029     "Identificador",
2030     "Identificador_lote",
2031     "Precio_adjudicacion",
2032     "N_ofertantes",
2033     "N_lotes",
2034     #"Presupuesto_licitacion",
2035     #"Tipo_de_contrato",
2036     #"Tipo_de_Administracion",
2037     #"Tipo_de_procedimiento",
2038     #"Tramitacion",
2039     #"Tipo_ganador_lote",
2040     "Tipo_de_contrato_Obras",
2041     "Tipo_de_Administracion_Autoridad_local",
2042     "Tipo_de_procedimiento_Abierto",
2043     "Tipo_de_procedimiento_Simplificado",
2044     "Tramitacion_Ordinaria",
2045     "Tipo_ganador_lote_NIF",
2046     "Plazo_m",
2047     #"Prorroga",
2048     "C_precio_p",
2049     #"C_resto_objetivos_p",
2050     #"C_juicio_valor_p",
2051     "Baja_p",
2052     #"Modi_eco_e", ## Para poder hacer limpieza de casos
2053     #"Modi_eco_p",
2054     #"Modi_temp_p", ## Para poder hacer limpieza de casos
2055     #"Clasi_max_req",
2056     #"Intervalo_lici_d",
2057     #"Intervalo_vigor_d",
2058     "Intervalo_urge_d",
2059     "N_CPV",
2060     "N_clasi_empresa",
2061     #"Clasi_empresa_main",
2062     #"Codigo_Postal", #-- Lo sacamos
2063     #"CPV_main",
2064     "N_requi",
2065     "Modi_cat"
2066   )
2067
2068
2069
2070 save(df_model_v2, file="df_model_v2_g.RData")
2071 write.xlsx(df_model_v2, "df_model_v2_g.xlsx", colNames = TRUE, rowNames = FALSE)
2072
2073 #Limpiamos
2074
2075 rm(list=ls())
2076
2077 # -----
2078 # SCRIPT 4 SOM_v2 -----
2079 # -----
2080
2081

```

```

2082 ## Directorio LEER v2 ----
2083
2084 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
2085
2086 ## Cargamos datos -----
2087
2088 load("bd_civil_sin_NA_v2.RData")
2089
2090 ## Directorio GUARDAR ----
2091
2092 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
2093
2094
2095
2096 ## PREPROCESAMIENTO de VARIABLES ##### -----
2097
2098 # Aseguramos que estamos trabajando sobre la base correcta
2099 df <- bd_civil_sin_NA
2100
2101 # Convertir factores de interés
2102 df <- df |>
2103   mutate(
2104     Tipo_de_Administracion = as.factor(Tipo_de_Administracion),
2105     Tipo_de_procedimiento = as.factor(Tipo_de_procedimiento),
2106     Tipo_de_contrato = as.factor(Tipo_de_contrato),
2107     Tramitacion = as.factor(Tramitacion),
2108     Tipo_ganador_lote = as.factor(Tipo_ganador_lote),
2109     Modis_s_n = as.factor(Modis_s_n)
2110   )
2111
2112
2113 # --- Tratamiento de la variable CPV en df ---
2114 df <- df |>
2115   mutate(
2116     CPV_truncs = purrr::map(CPV, function(x) {
2117       if (is.na(x) || stringr::str_trim(x) == "") return(character(0))
2118
2119       x_split <- stringr::str_split(x, ":", simplify = FALSE)[[1]]
2120       x_trimmed <- stringr::str_trim(x_split)
2121       x_valid <- x_trimmed[x_trimmed != ""]
2122       stringr::str_sub(x_valid, 1, 5)
2123     }),
2124
2125     N_CPV = purrr::map_int(CPV_truncs, length),
2126
2127     CPV_main = purrr::map_chr(CPV_truncs, function(cpvs) {
2128       if (length(cpvs) == 0) return(NA_character_)
2129
2130       cpvs_45 <- cpvs[stringr::str_starts(cpvs, "45")]
2131       if (length(cpvs_45) == 0) return(NA_character_)
2132
2133       ceros_derecha <- stringr::str_length(cpvs_45) -
2134         stringr::str_length(stringr::str_replace(cpvs_45, "0*$", ""))
2135
2136       min_ceros <- min(ceros_derecha)
2137       seleccionados <- cpvs_45[ceros_derecha == min_ceros]
2138
2139       return(seleccionados[1])
2140     })
2141   )
2142
2143
2144 # --- Tratamiento de la variable Clasi_empresa sobre el data frame df ---
2145
2146 # 1. Calcular N_clasi_empresa: número de clasificaciones registradas (separadas por
":")
2147 df <- df |>
2148   mutate(
2149     N_clasi_empresa = if_else(
2150       is.na(Clasi_empresa) | str_trim(Clasi_empresa) == "",
2151       0L,

```

```

2152     str_count(Clasi_empresa, ":") + 1L
2153   )
2154 )
2155
2156 # 2. Función para extraer la letra con el número más alto (o la primera si hay empate)
2157 extraer_letra_dominante <- function(clasi_string) {
2158   if (is.na(clasi_string) || str_trim(clasi_string) == "") {
2159     return("0") # Regla: si no hay clasificaciones, devolver "0"
2160   }
2161
2162   # Separar clasificaciones por ":"
2163   clasificaciones <- unlist(str_split(clasi_string, ":", simplify = FALSE))
2164
2165   # Extraer letra y número si está, o solo letra si número no es válido
2166   letras_valores <- str_match(clasificaciones, "^[A-Z][^0-9]*([0-9]*)")
2167
2168   # Filtrar filas válidas
2169   letras <- letras_valores[, 2]
2170   numeros <- suppressWarnings(as.numeric(letras_valores[, 3]))
2171
2172   # Si no hay letras válidas, devolver "0"
2173   if (all(is.na(letras))) {
2174     return("0")
2175   }
2176
2177   # En caso de números faltantes (NA), asumir valor 0
2178   numeros[is.na(numeros)] <- 0
2179
2180   # Agrupar por letra y conservar el máximo valor
2181   resumen <- aggregate(numeros, by = list(letras), FUN = max)
2182   colnames(resumen) <- c("letra", "max_valor")
2183
2184   # Escoger la letra con mayor valor, priorizando la primera en caso de empate
2185   maximo <- max(resumen$max_valor, na.rm = TRUE)
2186   letras_max <- resumen$letra[resumen$max_valor == maximo]
2187
2188   return(letras_max[1])
2189 }
2190
2191 # 3. Aplicar función y ajustar según la regla de N_clasi_empresa
2192 df <- df |>
2193   mutate(
2194     Clasi_empresa_main = vapply(Clasi_empresa, extraer_letra_dominante, character(1)),
2195     Clasi_empresa_main = if_else(N_clasi_empresa == 0, "0", Clasi_empresa_main)
2196   )
2197
2198 # --- Tratamiento de la variable Requi_adm: contar requisitos administrativos ---
2199
2200 df <- df |>
2201   mutate(
2202     # Si Requi_adm es NA o está vacía, asignar 0; si no, contar los elementos
2203     # separados por ":"
2204     N_requi = if_else(
2205       is.na(Requi_adm) | str_trim(Requi_adm) == "",
2206       0L,
2207       str_count(Requi_adm, ":") + 1L
2208     )
2209   )
2210
2211 # --- Tratamiento de la variable Prorroga ---
2212
2213 df <- df |>
2214   mutate(
2215     Prorroga = case_when(
2216       is.na(Prorroga) ~ "No", # Si está vacía → No
2217       str_detect(tolower(Prorroga), "no|sin|ninguna|ningun|ningún|n\\.a") ~ "No", #
2218       Si contiene negaciones comunes → No
2219       TRUE ~ "Sí" # En cualquier otro caso → Sí
2220     ),
2221     Prorroga = factor(Prorroga, levels = c("No", "Sí"))
2222   )

```



```

2223 # --- Tratamiento de la variable N_lotes ---
2224
2225 df <- df |>
2226   mutate(
2227     # Sustituimos "Sin lotes" por 1
2228     N_lotes = if_else(N_lotes == "Sin lotes", "1", N_lotes),
2229
2230     # Convertimos a entero
2231     N_lotes = as.integer(N_lotes)
2232   )
2233
2234 # --- Tratamiento de la variable Clasi_max_req ---
2235 df$Clasi_max_req <- ifelse(is.na(df$Clasi_max_req), 0, df$Clasi_max_req)
2236
2237
2238 # --- Selección inicial de variables para modelado predictivo ---
2239
2240 # Lista de variables seleccionadas para el modelo
2241 vars_modelo <- c(
2242   "Identificador",
2243   "Identificador_lote",
2244   "Precio_adjudicacion",
2245   "N_ofertantes",
2246   "N_lotes",
2247   "Presupuesto_licitacion",
2248   "Tipo_de_contrato",
2249   "Tipo_de_Administracion",
2250   "Tipo_de_procedimiento",
2251   "Tramitacion",
2252   "Tipo_ganador_lote",
2253   "Plazo_m",
2254   #"Prorroga",
2255   "C_precio_p",
2256   "C_resto_objetivos_p",
2257   "C_juicio_valor_p",
2258   "Baja_p",
2259   "Modi_eco_e", ## Para poder hacer limpieza de casos
2260   "Modi_eco_p",
2261   "Modi_temp_p", ## Para poder hacer limpieza de casos
2262   "Clasi_max_req",
2263   "Intervalo_lici_d",
2264   "Intervalo_vigor_d",
2265   "Intervalo_urge_d",
2266   "N_CPV",
2267   "N_clasi_empresa",
2268   "Clasi_empresa_main",
2269   "Codigo_Postal", #-- Lo sacamos
2270   "CPV_main",
2271   "N_requi"
2272 )
2273
2274 # Filtramos el data frame conservando solo esas variables
2275 df <- df |>
2276   dplyr::select(all_of(vars_modelo))
2277
2278 # --- Transformacion de variables para modelado predictivo ---
2279 ## Generamos nuevas variables categoricas a booleanas
2280 df <- df |>
2281   mutate(
2282     Tipo_de_contrato_Obras = if_else(Tipo_de_contrato == "Obras", 1L, 0L),
2283     Tipo_de_Administracion_Autoridad_local = if_else(Tipo_de_Administracion ==
2284       "Autoridad local", 1L, 0L),
2285     Tipo_de_procedimiento_Abierto = if_else(Tipo_de_procedimiento == "Abierto", 1L, 0L
2286     ),
2287     Tipo_de_procedimiento_Simplificado = if_else(Tipo_de_procedimiento == "Abierto
2288     simplificado", 1L, 0L),
2289     Tramitacion_Ordinaria = if_else(Tramitacion == "Ordinaria", 1L, 0L),
2290     Tipo_ganador_lote_NIF = if_else(Tipo_ganador_lote == "NIF", 1L, 0L)
2291   )
2292   ##Variables categóricas: aseguramos que estén como factor
2293   df$CPV_main <- factor(df$CPV_main)
2294   df$Clasi_empresa_main <- factor(df$Clasi_empresa_main)
2295   df$Codigo_Postal <- factor(df$Codigo_Postal)

```

```

2293
2294
2295 # Seleccion para transformacion de variables ---
2296 df_model <- df |>
2297   dplyr::select(
2298     "Identificador",
2299     "Identificador_lote",
2300     "Precio_adjudicacion",
2301     "N_ofertantes",
2302     "N_lotes",
2303     #"Presupuesto_licitacion",
2304     #"Tipo_de_contrato",
2305     #"Tipo_de_Administracion",
2306     #"Tipo_de_procedimiento",
2307     #"Tramitacion",
2308     #"Tipo_ganador_lote",
2309     "Tipo_de_contrato_Obras",
2310     "Tipo_de_Administracion_Autoridad_local",
2311     "Tipo_de_procedimiento_Abierto",
2312     "Tipo_de_procedimiento_Simplificado",
2313     "Tramitacion_Ordinaria",
2314     "Tipo_ganador_lote_NIF",
2315     "Plazo_m",
2316     #"Prorroga",
2317     "C_precio_p",
2318     #"C_resto_objetivos_p",
2319     #"C_juicio_valor_p",
2320     "Baja_p",
2321     "Modi_eco_e", ## Para poder hacer limpieza de casos
2322     "Modi_eco_p",
2323     "Modi_temp_p", ## Para poder hacer limpieza de casos
2324     "Clasi_max_req",
2325     #"Intervalo_lici_d",
2326     #"Intervalo_vigor_d",
2327     "Intervalo_urge_d",
2328     "N_CPV",
2329     "N_clasi_empresa",
2330     "Clasi_empresa_main",
2331     #"Codigo_Postal", #-- Lo sacamos
2332     #"CPV_main",
2333     "N_requi"
2334   )
2335
2336
2337 ##### HISTOGRAMA PREVIO ##### -----
2338
2339 # Crear cortes de 1% entre -5 y 55, incluyendo extremos
2340 intervalos <- c(-Inf, seq(-5, 55, by = 1), Inf)
2341
2342 # Crear factor por rangos tipo (a, b]
2343 df_model <- df_model |>
2344   mutate(
2345     modi_interval = cut(
2346       Modi_eco_p,
2347       breaks = intervalos,
2348       include.lowest = FALSE,
2349       right = TRUE,
2350       dig.lab = 5
2351     )
2352   )
2353
2354 # Guardar histograma en PDF
2355 pdf("Histograma_previo_v2.pdf", width = 10, height = 6)
2356 ggplot(df_model, aes(x = modi_interval)) +
2357   geom_bar(fill = "steelblue", color = "black") +
2358   labs(
2359     title = "Distribución de Modi_eco_p por Intervalos de 1%",
2360     x = "Intervalo Modi_eco_p (%)",
2361     y = "Frecuencia"
2362   ) +
2363   theme_minimal(base_size = 11) +
2364   theme(
2365     axis.text.x = element_text(angle = 45, hjust = 1, size = 7),

```

```

2366     plot.title = element_text(hjust = 0.5)
2367   )
2368 dev.off()
2369
2370 ### NUEVO FILTRO DE MODIS ECO 0 ####
2371
2372 # Contar casos que cumplen la condición de eliminación (Modi eco y temp 0)
2373 casos_eliminar <- df_model |>
2374   filter(
2375     Modi_eco_p == 0 &
2376     (is.na(Modi_temp_p) | Modi_temp_p == 0)
2377   )
2378
2379 n_eliminados <- nrow(casos_eliminar)
2380
2381 # Crear nuevo data frame excluyendo esos casos
2382 df_model_v2 <- df_model |>
2383   filter(
2384     !(Modi_eco_p == 0 &
2385       (is.na(Modi_temp_p) | Modi_temp_p == 0))
2386   ) |>
2387   mutate(
2388     modi_interval = cut(
2389       Modi_eco_p,
2390       breaks = intervalos,
2391       include.lowest = FALSE,
2392       right = TRUE,
2393       dig.lab = 5
2394     )
2395   )
2396
2397 # Mostrar el número de casos eliminados
2398 cat("Número de casos eliminados:", n_eliminados, "\n")
2399
2400 pdf("Histograma_previo_sin_ceros_v2.pdf", width = 10, height = 6)
2401 ggplot(df_model_v2, aes(x = modi_interval)) +
2402   geom_bar(fill = "steelblue", color = "black") +
2403   labs(
2404     title = "Distribución de Modi_eco_p por Intervalos de 1%_Filtrados 0s",
2405     x = "Intervalo Modi_eco_p (%)",
2406     y = "Frecuencia"
2407   ) +
2408   theme_minimal(base_size = 11) +
2409   theme(
2410     axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
2411     plot.title = element_text(hjust = 0.5)
2412   )
2413
2414 dev.off()
2415
2416 df_model_v2 <- df_model_v2 |>
2417   dplyr::select(- modi_interval)
2418
2419
2420
2421 # Crear variable categórica Modi_cat a partir de Modi_eco_p según rangos definidos
2422 df_model_v2 <- df_model_v2 |>
2423   mutate(
2424     Modi_cat = cut(
2425       Modi_eco_p,
2426       breaks = c(-Inf, 0, 5, 9, 10, 14, 15, 19, 20, 35, 49, 50, Inf),
2427       labels = 0:11,
2428       right = TRUE,
2429       include.lowest = FALSE
2430     ),
2431     Modi_cat = as.integer(as.character(Modi_cat)) # Convertir a entero
2432   )
2433
2434 # Etiquetas legibles para cada categoría de Modi_cat
2435 modi_cat_labels <- c(
2436   "(-Inf, 0]", "(0,5]", "(5,9]", "(9,10]", "(10,14]", "(14,15]",
2437   "(15,19]", "(19,20]", "(20,35]", "(35,49]", "(49,50]", "(50, Inf]"
2438 )

```

```

2439
2440 # Asignar factor con etiquetas para graficar
2441 df_model_v2 <- df_model_v2 |>
2442   mutate(Modi_cat_factor = factor(Modi_cat, levels = 0:11, labels = modi_cat_labels))
2443
2444 # Guardar histograma actualizado en PDF
2445 pdf("Histograma_post_v2.pdf", width = 10, height = 6)
2446 ggplot(df_model_v2, aes(x = Modi_cat_factor)) +
2447   geom_bar(fill = "darkcyan", color = "black") +
2448   labs(
2449     title = "Distribución de Modi_eco_p por Categorías (Modi_cat)",
2450     x = "Categorías de Modi_eco_p (%)",
2451     y = "Frecuencia"
2452   ) +
2453   theme_minimal(base_size = 11) +
2454   theme(
2455     axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
2456     plot.title = element_text(hjust = 0.5)
2457   )
2458 dev.off()
2459
2460
2461
2462 # Selecccion final de variables para entrenamiento ---
2463 df_model_v2 <- df_model_v2 |>
2464   dplyr::select(
2465     "Identificador",
2466     "Identificador_lote",
2467     "Precio_adjudicacion",
2468     "N_ofertantes",
2469     "N_lotes",
2470     #"Presupuesto_licitacion",
2471     #"Tipo_de_contrato",
2472     #"Tipo_de_Administracion",
2473     #"Tipo_de_procedimiento",
2474     #"Tramitacion",
2475     #"Tipo_ganador_lote",
2476     "Tipo_de_contrato_Obras",
2477     "Tipo_de_Administracion_Autoridad_local",
2478     "Tipo_de_procedimiento_Abierto",
2479     "Tipo_de_procedimiento_Simplificado",
2480     "Tramitacion_Ordinaria",
2481     "Tipo_ganador_lote_NIF",
2482     "Plazo_m",
2483     #"Prorroga",
2484     "C_precio_p",
2485     #"C_resto_objetivos_p",
2486     #"C_juicio_valor_p",
2487     "Baja_p",
2488     #"Modi_eco_e", ## Para poder hacer limpieza de casos
2489     #"Modi_eco_p",
2490     #"Modi_temp_p", ## Para poder hacer limpieza de casos
2491     #"Clasi_max_req",
2492     #"Intervalo_lici_d",
2493     #"Intervalo_vigor_d",
2494     "Intervalo_urge_d",
2495     "N_CPV",
2496     "N_clasi_empresa",
2497     #"Clasi_empresa_main",
2498     #"Codigo_Postal", #-- Lo sacamos
2499     #"CPV_main",
2500     "N_requi",
2501     "Modi_cat"
2502   )
2503
2504
2505 #### ENTRENAMIENTO SOM ### -----
2506 ### LIBRERÍAS NECESARIAS ----
2507
2508 # ---- PARTE 1: Preprocesamiento y División Train/Test ----
2509
2510 # División reproducible de los datos en train y test -----
2511

```

```

2512 set.seed(321)
2513
2514 n <- nrow(df_model_v2)
2515 train_idx <- sample(seq_len(n), size = 0.8 * n)
2516
2517 df_train_v2 <- df_model_v2[train_idx, ]
2518 df_test_v2 <- df_model_v2[-train_idx, ]
2519
2520 # Guardamos conjuntos base
2521 saveRDS(list(
2522   df_train = df_train_v2,
2523   df_test = df_test_v2,
2524   vars_predictoras = setdiff(colnames(df_model_v2), "Modi_cat")
2525 ), "00_datos_base_SOM_g.rds")
2526
2527 save(df_train_v2, file="df_train_SOM_v2_g.RData")
2528 write.xlsx(df_train_v2, "df_train_SOM_v2_g.xlsx", colNames = TRUE, rowNames = FALSE)
2529 save(df_test_v2, file="df_test_SOM_v2_g.RData")
2530 write.xlsx(df_test_v2, "df_test_SOM_v2_g.xlsx", colNames = TRUE, rowNames = FALSE)
2531
2532
2533 ## Limpiamos
2534 rm(list=ls())
2535
2536
2537
2538 # -----
2539 # SCRIPT 5 SOM_v3 -----
2540 # -----
2541
2542
2543 ## Directorio LEER v3 ----
2544
2545 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
2546
2547 ## Cargamos datos -----
2548
2549 load("bd_civil_sin_NA_v2.RData")
2550
2551 ## Directorio GUARDAR ----
2552
2553 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
2554
2555
2556 ## PREPROCESAMIENTO de VARIABLES ##### -----
2557
2558 # Aseguramos que estamos trabajando sobre la base correcta
2559 df <- bd_civil_sin_NA
2560
2561 # Convertir factores de interés
2562 df <- df |>
2563   mutate(
2564     Tipo_de_Administracion = as.factor(Tipo_de_Administracion),
2565     Tipo_de_procedimiento = as.factor(Tipo_de_procedimiento),
2566     Tipo_de_contrato = as.factor(Tipo_de_contrato),
2567     Tramitacion = as.factor(Tramitacion),
2568     Tipo_ganador_lote = as.factor(Tipo_ganador_lote),
2569     Modis_s_n = as.factor(Modis_s_n)
2570   )
2571
2572
2573 # --- Tratamiento de la variable CPV en df ---
2574 df <- df |>
2575   mutate(
2576     CPV_truncs = purrr::map(CPV, function(x) {
2577       if (is.na(x) || stringr::str_trim(x) == "") return(character(0))
2578
2579       x_split <- stringr::str_split(x, ":", simplify = FALSE)[[1]]
2580       x_trimmed <- stringr::str_trim(x_split)
2581       x_valid <- x_trimmed[x_trimmed != ""]
2582       stringr::str_sub(x_valid, 1, 5)

```

```

2583   }),
2584
2585   N_CPV = purrr::map_int(CPV_truncs, length),
2586
2587   CPV_main = purrr::map_chr(CPV_truncs, function(cpvs) {
2588     if (length(cpvs) == 0) return(NA_character_)
2589
2590     cpvs_45 <- cpvs[stringr::str_starts(cpvs, "45")]
2591     if (length(cpvs_45) == 0) return(NA_character_)
2592
2593     ceros_derecha <- stringr::str_length(cpvs_45) -
2594       stringr::str_length(stringr::str_replace(cpvs_45, "0*$", ""))
2595
2596     min_ceros <- min(ceros_derecha)
2597     seleccionados <- cpvs_45[ceros_derecha == min_ceros]
2598
2599     return(seleccionados[1])
2600   })
2601 )
2602
2603
2604 # --- Tratamiento de la variable Clasi_empresa sobre el data frame df ---
2605
2606 # 1. Calcular N_clasi_empresa: número de clasificaciones registradas (separadas por
2607 # ":")
2608 df <- df |>
2609   mutate(
2610     N_clasi_empresa = if_else(
2611       is.na(Clasi_empresa) | str_trim(Clasi_empresa) == "",
2612       0L,
2613       str_count(Clasi_empresa, ":") + 1L
2614     )
2615   )
2616
2617 # 2. Función para extraer la letra con el número más alto (o la primera si hay empate)
2618 extraer_letra_dominante <- function(clasi_string) {
2619   if (is.na(clasi_string) || str_trim(clasi_string) == "") {
2620     return("0") # Regla: si no hay clasificaciones, devolver "0"
2621   }
2622
2623   # Separar clasificaciones por ":"
2624   clasificaciones <- unlist(str_split(clasi_string, ":", simplify = FALSE))
2625
2626   # Extraer letra y número si está, o solo letra si número no es válido
2627   letras_valores <- str_match(clasificaciones, "^[A-Z][^0-9]*([0-9]*)")
2628
2629   # Filtrar filas válidas
2630   letras <- letras_valores[, 2]
2631   numeros <- suppressWarnings(as.numeric(letras_valores[, 3]))
2632
2633   # Si no hay letras válidas, devolver "0"
2634   if (all(is.na(letras))) {
2635     return("0")
2636   }
2637
2638   # En caso de números faltantes (NA), asumir valor 0
2639   numeros[is.na(numeros)] <- 0
2640
2641   # Agrupar por letra y conservar el máximo valor
2642   resumen <- aggregate(numeros, by = list(letras), FUN = max)
2643   colnames(resumen) <- c("letra", "max_valor")
2644
2645   # Escoger la letra con mayor valor, priorizando la primera en caso de empate
2646   maximo <- max(resumen$max_valor, na.rm = TRUE)
2647   letras_max <- resumen$letra[resumen$max_valor == maximo]
2648
2649   return(letras_max[1])
2650 }
2651
2652 # 3. Aplicar función y ajustar según la regla de N_clasi_empresa
2653 df <- df |>
2654   mutate(
2655     Clasi_empresa_main = vapply(Clasi_empresa, extraer_letra_dominante, character(1)),

```

```

2655     Clasi_empresa_main = if_else(N_clasi_empresa == 0, "0", Clasi_empresa_main)
2656   )
2657
2658   # --- Tratamiento de la variable Requi_adm: contar requisitos administrativos ---
2659
2660   df <- df |>
2661     mutate(
2662       # Si Requi_adm es NA o está vacía, asignar 0; si no, contar los elementos
2663       # separados por ":"
2664       N_requi = if_else(
2665         is.na(Requi_adm) | str_trim(Requi_adm) == "",
2666         0L,
2667         str_count(Requi_adm, ":") + 1L
2668       )
2669     )
2670
2671   # --- Tratamiento de la variable Prorroga ---
2672
2673   df <- df |>
2674     mutate(
2675       Prorroga = case_when(
2676         is.na(Prorroga) ~ "No", # Si está vacía → No
2677         str_detect(tolower(Prorroga), "no|sin|ninguna|ningun|ningún|n\\\.a") ~ "No", #
2678         # Si contiene negaciones comunes → No
2679         TRUE ~ "Sí" # En cualquier otro caso → Sí
2680       ),
2681       Prorroga = factor(Prorroga, levels = c("No", "Sí"))
2682     )
2683
2684   # --- Tratamiento de la variable N_lotes ---
2685
2686   df <- df |>
2687     mutate(
2688       # Sustituimos "Sin lotes" por 1
2689       N_lotes = if_else(N_lotes == "Sin lotes", "1", N_lotes),
2690
2691       # Convertimos a entero
2692       N_lotes = as.integer(N_lotes)
2693     )
2694
2695   # --- Tratamiento de la variable Clasi_max_req ---
2696   df$Clasi_max_req <- ifelse(is.na(df$Clasi_max_req), 0, df$Clasi_max_req)
2697
2698   # --- Selección inicial de variables para modelado predictivo ---
2699
2700   # Lista de variables seleccionadas para el modelo
2701   vars_modelo <- c(
2702     "Identificador",
2703     "Identificador_lote",
2704     "Precio_adjudicacion",
2705     "N_ofertantes",
2706     "N_lotes",
2707     "Presupuesto_licitacion",
2708     "Tipo_de_contrato",
2709     "Tipo_de_Administracion",
2710     "Tipo_de_procedimiento",
2711     "Tramitacion",
2712     "Tipo_ganador_lote",
2713     "Plazo_m",
2714     #"Prorroga",
2715     "C_precio_p",
2716     "C_resto_objetivos_p",
2717     "C_juicio_valor_p",
2718     "Baja_p",
2719     "Modi_eco_e", ## Para poder hacer limpieza de casos
2720     "Modi_eco_p",
2721     "Modi_temp_p", ## Para poder hacer limpieza de casos
2722     "Clasi_max_req",
2723     "Intervalo_lici_d",
2724     "Intervalo_vigor_d",
2725     "Intervalo_urge_d",

```

```

2726     "N_CPV",
2727     "N_clasi_empresa",
2728     "Clasi_empresa_main",
2729     "Codigo_Postal", #-- Lo sacamos
2730     "CPV_main",
2731     "N_requi"
2732 )
2733
2734 # Filtramos el data frame conservando solo esas variables
2735 df <- df |>
2736   dplyr::select(all_of(vars_modelo))
2737
2738 # --- Transformacion de variables para modelado predictivo ---
2739 ## Generamos nuevas variables categoricas a booleanas
2740 df <- df |>
2741   mutate(
2742     Tipo_de_contrato_Obras = if_else(Tipo_de_contrato == "Obras", 1L, 0L),
2743     Tipo_de_Administracion_Autoridad_local = if_else(Tipo_de_Administracion ==
2744       "Autoridad local", 1L, 0L),
2745     Tipo_de_procedimiento_Abierto = if_else(Tipo_de_procedimiento == "Abierto", 1L, 0L
2746     ),
2747     Tipo_de_procedimiento_Simplificado = if_else(Tipo_de_procedimiento == "Abierto
2748     simplificado", 1L, 0L),
2749     Tramitacion_Ordinaria = if_else(Tramitacion == "Ordinaria", 1L, 0L),
2750     Tipo_ganador_lote_NIF = if_else(Tipo_ganador_lote == "NIF", 1L, 0L)
2751   )
2752 ##Variables categóricas: aseguramos que estén como factor
2753 df$CPV_main <- factor(df$CPV_main)
2754 df$Clasi_empresa_main <- factor(df$Clasi_empresa_main)
2755 df$Codigo_Postal <- factor(df$Codigo_Postal)
2756
2757 # Selecccion para transformacion de variables ---
2758 df_model <- df |>
2759   dplyr::select(
2760     "Identificador",
2761     "Identificador_lote",
2762     "Precio_adjudicacion",
2763     "N_ofertantes",
2764     "N_lotes",
2765     #"Presupuesto_licitacion",
2766     #"Tipo_de_contrato",
2767     #"Tipo_de_Administracion",
2768     #"Tipo_de_procedimiento",
2769     #"Tramitacion",
2770     #"Tipo_ganador_lote",
2771     "Tipo_de_contrato_Obras",
2772     "Tipo_de_Administracion_Autoridad_local",
2773     "Tipo_de_procedimiento_Abierto",
2774     "Tipo_de_procedimiento_Simplificado",
2775     "Tramitacion_Ordinaria",
2776     "Tipo_ganador_lote_NIF",
2777     "Plazo_m",
2778     #"Prorroga",
2779     "C_precio_p",
2780     #"C_resto_objetivos_p",
2781     #"C_juicio_valor_p",
2782     "Baja_p",
2783     "Modi_eco_e", ## Para poder hacer limpieza de casos
2784     "Modi_eco_p",
2785     "Modi_temp_p", ## Para poder hacer limpieza de casos
2786     "Clasi_max_req",
2787     #"Intervalo_lici_d",
2788     #"Intervalo_vigor_d",
2789     "Intervalo_urge_d",
2790     "N_CPV",
2791     "N_clasi_empresa",
2792     "Clasi_empresa_main",
2793     #"Codigo_Postal", #-- Lo sacamos
2794     #"CPV_main",
2795     "N_requi"
2796   )

```



```

2796
2797 ##### HISTOGRAMA PREVIO ##### -----
2798
2799 # Crear cortes de 1% entre -5 y 55, incluyendo extremos
2800 intervalos <- c(-Inf, seq(-5, 55, by = 1), Inf)
2801
2802 # Crear factor por rangos tipo (a, b]
2803 df_model <- df_model |>
2804   mutate(
2805     modi_interval = cut(
2806       Modi_eco_p,
2807       breaks = intervalos,
2808       include.lowest = FALSE,
2809       right = TRUE,
2810       dig.lab = 5
2811     )
2812   )
2813
2814 # Guardar histograma en PDF
2815 pdf("Histograma_previo_v3_g.pdf", width = 10, height = 6)
2816 ggplot(df_model, aes(x = modi_interval)) +
2817   geom_bar(fill = "steelblue", color = "black") +
2818   labs(
2819     title = "Distribución de Modi_eco_p por Intervalos de 1%",
2820     x = "Intervalo Modi_eco_p (%)",
2821     y = "Frecuencia"
2822   ) +
2823   theme_minimal(base_size = 11) +
2824   theme(
2825     axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
2826     plot.title = element_text(hjust = 0.5)
2827   )
2828 dev.off()
2829
2830 ##### NUEVO FILTRO DE MODIS ECO 0 #####
2831
2832 # Contar casos que cumplen la condición de eliminación (Modi eco y temp 0)
2833 casos_eliminar <- df_model |>
2834   filter(
2835     Modi_eco_p == 0 &
2836     (is.na(Modi_temp_p) | Modi_temp_p == 0)
2837   )
2838
2839 n_eliminados <- nrow(casos_eliminar)
2840
2841 # Crear nuevo data frame excluyendo esos casos
2842 df_model_v3 <- df_model |>
2843   filter(
2844     !(Modi_eco_p == 0 &
2845       (is.na(Modi_temp_p) | Modi_temp_p == 0))
2846   ) |>
2847   mutate(
2848     modi_interval = cut(
2849       Modi_eco_p,
2850       breaks = intervalos,
2851       include.lowest = FALSE,
2852       right = TRUE,
2853       dig.lab = 5
2854     )
2855   )
2856
2857 # Mostrar el número de casos eliminados
2858 cat("Número de casos eliminados con Modi_eco=0 y Modi_temp=NA/0:", n_eliminados, "\n")
2859
2860 pdf("Histograma_previo_sin_ceros_v3_g.pdf", width = 10, height = 6)
2861 ggplot(df_model_v3, aes(x = modi_interval)) +
2862   geom_bar(fill = "steelblue", color = "black") +
2863   labs(
2864     title = "Distribución de Modi_eco_p por Intervalos de 1%_Filtrados 0s",
2865     x = "Intervalo Modi_eco_p (%)",
2866     y = "Frecuencia"
2867   ) +
2868   theme_minimal(base_size = 11) +

```

```

2869     theme(
2870       axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
2871       plot.title = element_text(hjust = 0.5)
2872     )
2873
2874 dev.off()
2875
2876 ## Forzamos a todos los valores del intervalo "(-1,0]" a ser 0
2877 df_model_v3 <- df_model_v3 |>
2878   mutate(
2879     Modi_eco_p = if_else(
2880       modi_interval == "(-1,0]",
2881       0,
2882       Modi_eco_p
2883     )
2884   )
2885
2886
2887 # === Guardamos el número de casos actuales antes del filtrado ===
2888 n_antes <- nrow(df_model_v3)
2889
2890 # === Eliminamos los casos donde Modi_eco_p sea menor o igual a 0 o mayor a 54.99 ===
2891 df_model_v3 <- df_model_v3 |>
2892   filter(Modi_eco_p > 0, Modi_eco_p <= 54.99)
2893
2894 # === Guardamos cuántos casos se han eliminado ===
2895 n_despues <- nrow(df_model_v3)
2896 n_casos_colas <- n_antes - n_despues
2897
2898 # === Mostramos resultados por consola ===
2899 cat("☑ Número de casos eliminados (Modi_eco_p fuera de (0, 54.99]):", n_casos_colas,
2900     "\n")
2901 cat("☑ Número de casos restantes en df_model_v3:", n_despues, "\n")
2902
2903 pdf("Histograma_post_v3_sin_colas_v3_g.pdf", width = 10, height = 6)
2904 ggplot(df_model_v3, aes(x = modi_interval)) +
2905   geom_bar(fill = "coral", color = "black") +
2906   labs(
2907     title = "Distribución de Modi_eco_p por Intervalos sin colas",
2908     x = "Intervalo Modi_eco_p (%)",
2909     y = "Frecuencia"
2910   ) +
2911   theme_minimal(base_size = 11) +
2912   theme(
2913     axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
2914     plot.title = element_text(hjust = 0.5)
2915   )
2916
2917 dev.off()
2918
2919 ## Eliminamos variable categorica de intervalos
2920 df_model_v3 <- df_model_v3 |>
2921   dplyr::select(- modi_interval)
2922
2923
2924 ### Seleccion final de variables para entrenamiento ### ----
2925 df_model_v3 <- df_model_v3 |>
2926   dplyr::select(
2927     "Identificador",
2928     "Identificador_lote",
2929     "Precio_adjudicacion",
2930     "N_ofertantes",
2931     "N_lotes",
2932     #"Presupuesto_licitacion",
2933     #"Tipo_de_contrato",
2934     #"Tipo_de_Administracion",
2935     #"Tipo_de_procedimiento",
2936     #"Tramitacion",
2937     #"Tipo_ganador_lote",
2938     "Tipo_de_contrato_Obras",
2939     "Tipo_de_Administracion_Autoridad_local",
2940     "Tipo_de_procedimiento_Abierto",

```

```

2941     #"Tipo_de_procedimiento_Simplificado",
2942     "Tramitacion_Ordinaria",
2943     #"Tipo_ganador_lote_NIF",
2944     "Plazo_m",
2945     #"Prorroga",
2946     "C_precio_p",
2947     #"C_resto_objetivos_p",
2948     #"C_juicio_valor_p",
2949     "Baja_p",
2950     #"Modi_eco_e", ## Para poder hacer limpieza de casos
2951     "Modi_eco_p",
2952     #"Modi_temp_p", ## Para poder hacer limpieza de casos
2953     #"Clasi_max_req",
2954     #"Intervalo_licid",
2955     #"Intervalo_vigor_d",
2956     "Intervalo_urge_d",
2957     "N_CPV",
2958     "N_clasi_empresa",
2959     #"Clasi_empresa_main",
2960     #"Codigo_Postal", #-- Lo sacamos
2961     #"CPV_main",
2962     #"N_requi",
2963
2964   )
2965
2966
2967
2968   # === División reproducible de los datos en entrenamiento y test === -----
2969
2970   set.seed(321)
2971
2972   n <- nrow(df_model_v3)
2973   train_idx <- sample.int(n, size = floor(0.8 * n)) # Selección aleatoria reproducible
2974
2975   # Crear conjuntos de entrenamiento y test
2976   df_train_v3 <- df_model_v3[train_idx, ]
2977   df_test_v3 <- df_model_v3[-train_idx, ]
2978
2979   # Comprobamos que la variable objetivo está presente
2980   stopifnot("Modi_eco_p" %in% names(df_model_v3))
2981
2982   # Guardamos los conjuntos y nombres de variables predictoras
2983   saveRDS(list(
2984     df_train = df_train_v3,
2985     df_test = df_test_v3,
2986     vars_predictoras = setdiff(colnames(df_model_v3), "Modi_eco_p")
2987   ), "datos_base_SOM_v3_g.rds")
2988
2989
2990   save(df_train_v3, file="df_train_SOM_v3_g.RData")
2991   write.xlsx(df_train_v3, "df_train_SOM_v3_g.xlsx", colNames = TRUE, rowNames = FALSE)
2992   save(df_test_v3, file="df_test_SOM_v3_g.RData")
2993   write.xlsx(df_test_v3, "df_test_SOM_v3_g.xlsx", colNames = TRUE, rowNames = FALSE)
2994
2995
2996   ## Limpiamos
2997   rm(list=ls())
2998
2999
3000   # -----
3001   # SCRIPT 6 y 7 SOM_v4 y Script_Distribucion_v3 -----
3002   # -----
3003
3004
3005
3006   ## Directorio LEER v4 ----
3007
3008   setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
3009
3010   ## Cargamos datos -----
3011
3012   load("bd_civil_sin_NA_v2.RData")

```

```

3013
3014 ## Directorio GUARDAR ----
3015
3016 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4")
3017
3018
3019
3020 ## PREPROCESAMIENTO de VARIABLES ##### -----
3021
3022 # Aseguramos que estamos trabajando sobre la base correcta
3023 df <- bd_civil_sin_NA
3024
3025 # Convertir factores de interés
3026 df <- df |>
3027   mutate(
3028     Tipo_de_Administracion = as.factor(Tipo_de_Administracion),
3029     Tipo_de_procedimiento = as.factor(Tipo_de_procedimiento),
3030     Tipo_de_contrato = as.factor(Tipo_de_contrato),
3031     Tramitacion = as.factor(Tramitacion),
3032     Tipo_ganador_lote = as.factor(Tipo_ganador_lote),
3033     Modis_s_n = as.factor(Modis_s_n)
3034   )
3035
3036
3037 # --- Tratamiento de la variable CPV en df ---
3038 df <- df |>
3039   mutate(
3040     CPV_truncs = purrr::map(CPV, function(x) {
3041       if (is.na(x) || stringr::str_trim(x) == "") return(character(0))
3042
3043       x_split <- stringr::str_split(x, ":", simplify = FALSE)[[1]]
3044       x_trimmed <- stringr::str_trim(x_split)
3045       x_valid <- x_trimmed[x_trimmed != ""]
3046       stringr::str_sub(x_valid, 1, 5)
3047     }),
3048
3049     N_CPV = purrr::map_int(CPV_truncs, length),
3050
3051     CPV_main = purrr::map_chr(CPV_truncs, function(cpvs) {
3052       if (length(cpvs) == 0) return(NA_character_)
3053
3054       cpvs_45 <- cpvs[stringr::str_starts(cpvs, "45")]
3055       if (length(cpvs_45) == 0) return(NA_character_)
3056
3057       ceros_derecha <- stringr::str_length(cpvs_45) -
3058         stringr::str_length(stringr::str_replace(cpvs_45, "0*$", ""))
3059
3060       min_ceros <- min(ceros_derecha)
3061       seleccionados <- cpvs_45[ceros_derecha == min_ceros]
3062
3063       return(seleccionados[1])
3064     })
3065   )
3066
3067
3068 # --- Tratamiento de la variable Clasi_empresa sobre el data frame df ---
3069
3070 # 1. Calcular N_clasi_empresa: número de clasificaciones registradas (separadas por
":")
3071 df <- df |>
3072   mutate(
3073     N_clasi_empresa = if_else(
3074       is.na(Clasi_empresa) | str_trim(Clasi_empresa) == "",
3075       0L,
3076       str_count(Clasi_empresa, ":") + 1L
3077     )
3078   )
3079
3080 # 2. Función para extraer la letra con el número más alto (o la primera si hay empate)
3081 extraer_letra_dominante <- function(clasi_string) {
3082   if (is.na(clasi_string) || str_trim(clasi_string) == "") {
3083     return("0") # Regla: si no hay clasificaciones, devolver "0"

```

```

3084 }
3085
3086 # Separar clasificaciones por ":"
3087 clasificaciones <- unlist(str_split(clasi_string, ":", simplify = FALSE))
3088
3089 # Extraer letra y número si está, o solo letra si número no es válido
3090 letras_valores <- str_match(clasificaciones, "^[A-Z])[^0-9]*([0-9]*)")
3091
3092 # Filtrar filas válidas
3093 letras <- letras_valores[, 2]
3094 numeros <- suppressWarnings(as.numeric(letras_valores[, 3]))
3095
3096 # Si no hay letras válidas, devolver "0"
3097 if (all(is.na(letras))) {
3098   return("0")
3099 }
3100
3101 # En caso de números faltantes (NA), asumir valor 0
3102 numeros[is.na(numeros)] <- 0
3103
3104 # Agrupar por letra y conservar el máximo valor
3105 resumen <- aggregate(numeros, by = list(letras), FUN = max)
3106 colnames(resumen) <- c("letra", "max_valor")
3107
3108 # Escoger la letra con mayor valor, priorizando la primera en caso de empate
3109 maximo <- max(resumen$max_valor, na.rm = TRUE)
3110 letras_max <- resumen$letra[resumen$max_valor == maximo]
3111
3112 return(letras_max[1])
3113 }
3114
3115 # 3. Aplicar función y ajustar según la regla de N_clasi_empresa
3116 df <- df |>
3117   mutate(
3118     Clasi_empresa_main = vapply(Clasi_empresa, extraer_letra_dominante, character(1)),
3119     Clasi_empresa_main = if_else(N_clasi_empresa == 0, "0", Clasi_empresa_main)
3120   )
3121
3122 # --- Tratamiento de la variable Requi_adm: contar requisitos administrativos ---
3123
3124 df <- df |>
3125   mutate(
3126     # Si Requi_adm es NA o está vacía, asignar 0; si no, contar los elementos
3127     # separados por ":"
3128     N_requi = if_else(
3129       is.na(Requi_adm) | str_trim(Requi_adm) == "",
3130       0L,
3131       str_count(Requi_adm, ":") + 1L
3132     )
3133   )
3134
3135 # --- Tratamiento de la variable Prorroga ---
3136
3137 df <- df |>
3138   mutate(
3139     Prorroga = case_when(
3140       is.na(Prorroga) ~ "No", # Si está vacía → No
3141       str_detect(tolower(Prorroga), "no|sin|ninguna|ningun|ningún|n\\.a") ~ "No", #
3142       Si contiene negaciones comunes → No
3143       TRUE ~ "Sí" # En cualquier otro caso → Sí
3144     ),
3145     Prorroga = factor(Prorroga, levels = c("No", "Sí"))
3146   )
3147
3148 # --- Tratamiento de la variable N_lotes ---
3149
3150 df <- df |>
3151   mutate(
3152     # Sustituimos "Sin lotes" por 1
3153     N_lotes = if_else(N_lotes == "Sin lotes", "1", N_lotes),
3154     # Convertimos a entero

```

```

3155     N_lotes = as.integer(N_lotes)
3156 )
3157
3158 # --- Tratamiento de la variable Clasi_max_req ---
3159 df$Clasi_max_req <- ifelse(is.na(df$Clasi_max_req), 0, df$Clasi_max_req)
3160
3161
3162 # --- Selección inicial de variables para modelado predictivo ---
3163
3164 # Lista de variables seleccionadas para el modelo
3165 vars_modelo <- c(
3166   "Identificador",
3167   "Identificador_lote",
3168   "Precio_adjudicacion",
3169   "N_ofertantes",
3170   "N_lotes",
3171   "Presupuesto_licitacion",
3172   "Tipo_de_contrato",
3173   "Tipo_de_Administracion",
3174   "Tipo_de_procedimiento",
3175   "Tramitacion",
3176   "Tipo_ganador_lote",
3177   "Plazo_m",
3178   #"Prorroga",
3179   "C_precio_p",
3180   "C_resto_objetivos_p",
3181   "C_juicio_valor_p",
3182   "Baja_p",
3183   "Modi_eco_e", ## Para poder hacer limpieza de casos
3184   "Modi_eco_p",
3185   "Modi_temp_p", ## Para poder hacer limpieza de casos
3186   "Clasi_max_req",
3187   "Intervalo_lici_d",
3188   "Intervalo_vigor_d",
3189   "Intervalo_urge_d",
3190   "N_CPV",
3191   "N_clasi_empresa",
3192   "Clasi_empresa_main",
3193   "Codigo_Postal", #-- Lo sacamos
3194   "CPV_main",
3195   "N_requi"
3196 )
3197
3198 # Filtramos el data frame conservando solo esas variables
3199 df <- df |>
3200   dplyr::select(all_of(vars_modelo))
3201
3202 # --- Transformacion de variables para modelado predictivo ---
3203 ## Generamos nuevas variables categoricas a booleanas
3204 df <- df |>
3205   mutate(
3206     Tipo_de_contrato_Obras = if_else(Tipo_de_contrato == "Obras", 1L, 0L),
3207     Tipo_de_Administracion_Autoridad_local = if_else(Tipo_de_Administracion ==
3208       "Autoridad local", 1L, 0L),
3209     Tipo_de_procedimiento_Abierto = if_else(Tipo_de_procedimiento == "Abierto", 1L, 0L
3210     ),
3211     Tipo_de_procedimiento_Simplificado = if_else(Tipo_de_procedimiento == "Abierto
3212     simplificado", 1L, 0L),
3213     Tramitacion_Ordinaria = if_else(Tramitacion == "Ordinaria", 1L, 0L),
3214     Tipo_ganador_lote_NIF = if_else(Tipo_ganador_lote == "NIF", 1L, 0L)
3215   )
3216 ##Variables categóricas: aseguramos que estén como factor
3217 df$CPV_main <- factor(df$CPV_main)
3218 df$Clasi_empresa_main <- factor(df$Clasi_empresa_main)
3219 df$Codigo_Postal <- factor(df$Codigo_Postal)
3220
3221 # Selección para transformacion de variables ---
3222 df_model <- df |>
3223   dplyr::select(
3224     "Identificador",
3225     "Identificador_lote",
3226     "Precio_adjudicacion",

```

```

3225     "N_ofertantes",
3226     "N_lotes",
3227     #"Presupuesto_licitacion",
3228     #"Tipo_de_contrato",
3229     #"Tipo_de_Administracion",
3230     #"Tipo_de_procedimiento",
3231     #"Tramitacion",
3232     #"Tipo_ganador_lote",
3233     "Tipo_de_contrato_Obras",
3234     "Tipo_de_Administracion_Autoridad_local",
3235     "Tipo_de_procedimiento_Abierto",
3236     "Tipo_de_procedimiento_Simplificado",
3237     "Tramitacion_Ordinaria",
3238     "Tipo_ganador_lote_NIF",
3239     "Plazo_m",
3240     #"Prorroga",
3241     "C_precio_p",
3242     #"C_resto_objetivos_p",
3243     #"C_juicio_valor_p",
3244     "Baja_p",
3245     "Modi_eco_e", ## Para poder hacer limpieza de casos
3246     "Modi_eco_p",
3247     "Modi_temp_p", ## Para poder hacer limpieza de casos
3248     "Clasi_max_req",
3249     #"Intervalo_lici_d",
3250     #"Intervalo_vigor_d",
3251     "Intervalo_urge_d",
3252     "N_CPV",
3253     "N_clasi_empresa",
3254     "Clasi_empresa_main",
3255     #"Codigo_Postal", #-- Lo sacamos
3256     #"CPV_main",
3257     "N_requi"
3258
3259 )
3260
3261 ##### HISTOGRAMA PREVIO ##### -----
3262
3263 # Crear cortes de 1% entre -5 y 55, incluyendo extremos
3264 intervalos <- c(-Inf, seq(-5, 55, by = 1), Inf)
3265
3266 # Crear factor por rangos tipo (a, b]
3267 df_model <- df_model |>
3268   mutate(
3269     modi_interval = cut(
3270       Modi_eco_p,
3271       breaks = intervalos,
3272       include.lowest = FALSE,
3273       right = TRUE,
3274       dig.lab = 5
3275     )
3276   )
3277
3278 # Guardar histograma en PDF
3279 pdf("Histograma_previo_v4_g.pdf", width = 10, height = 6)
3280 ggplot(df_model, aes(x = modi_interval)) +
3281   geom_bar(fill = "steelblue", color = "black") +
3282   labs(
3283     title = "Distribución de Modi_eco_p por Intervalos de 1%",
3284     x = "Intervalo Modi_eco_p (%)",
3285     y = "Frecuencia"
3286   ) +
3287   theme_minimal(base_size = 11) +
3288   theme(
3289     axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
3290     plot.title = element_text(hjust = 0.5)
3291   )
3292 dev.off()
3293
3294 ##### NUEVO FILTRO DE MODIS ECO 0 #####
3295
3296 # Contar casos que cumplen la condición de eliminación (Modi eco y temp 0)
3297 casos_eliminar <- df_model |>

```

```

3298     filter(
3299       Modi_eco_p == 0 &
3300       (is.na(Modi_temp_p) | Modi_temp_p == 0)
3301     )
3302
3303 n_eliminados <- nrow(casos_eliminar)
3304
3305 # Crear nuevo data frame excluyendo esos casos
3306 df_model_v3 <- df_model |>
3307   filter(
3308     !(Modi_eco_p == 0 &
3309       (is.na(Modi_temp_p) | Modi_temp_p == 0))
3310   ) |>
3311   mutate(
3312     modi_interval = cut(
3313       Modi_eco_p,
3314       breaks = intervalos,
3315       include.lowest = FALSE,
3316       right = TRUE,
3317       dig.lab = 5
3318     )
3319   )
3320
3321 # Mostrar el número de casos eliminados
3322 cat("Número de casos eliminados con Modi_eco=0 y Modi_temp=NA/0:", n_eliminados, "\n")
3323
3324 pdf("Histograma_previo_sin_ceros_v4_g.pdf", width = 10, height = 6)
3325 ggplot(df_model_v3, aes(x = modi_interval)) +
3326   geom_bar(fill = "steelblue", color = "black") +
3327   labs(
3328     title = "Distribución de Modi_eco_p por Intervalos de 1%_Filtrados 0s",
3329     x = "Intervalo Modi_eco_p (%)",
3330     y = "Frecuencia"
3331   ) +
3332   theme_minimal(base_size = 11) +
3333   theme(
3334     axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
3335     plot.title = element_text(hjust = 0.5)
3336   )
3337
3338 dev.off()
3339
3340 ## Forzamos a todos los valores del intervalo "(-1,0]" a ser 0
3341 df_model_v3 <- df_model_v3 |>
3342   mutate(
3343     Modi_eco_p = if_else(
3344       modi_interval == "(-1,0]",
3345       0,
3346       Modi_eco_p
3347     )
3348   )
3349
3350
3351 # === Guardamos el número de casos actuales antes del filtrado ===
3352 n_antes <- nrow(df_model_v3)
3353
3354 # === Eliminamos los casos donde Modi_eco_p sea menor o igual a 0 o mayor a 54.99 ===
3355 df_model_v3 <- df_model_v3 |>
3356   filter(Modi_eco_p > 0, Modi_eco_p <= 54.99)
3357
3358 # === Guardamos cuántos casos se han eliminado ===
3359 n_despues <- nrow(df_model_v3)
3360 n_casos_colas <- n_antes - n_despues
3361
3362 # === Mostramos resultados por consola ===
3363 cat("☑ Número de casos eliminados (Modi_eco_p fuera de (0, 54.99]):", n_casos_colas,
3364     "\n")
3365 cat("☑ Número de casos restantes en df_model_v3:", n_despues, "\n")
3366
3367 pdf("Histograma_post_v3_sin_colas.pdf", width = 10, height = 6)
3368 ggplot(df_model_v3, aes(x = modi_interval)) +
3369   geom_bar(fill = "coral", color = "black") +

```



```

3370 labs(
3371   title = "Distribución de Modi_eco_p por Intervalos sin colas",
3372   x = "Intervalo Modi_eco_p (%)",
3373   y = "Frecuencia"
3374 ) +
3375 theme_minimal(base_size = 11) +
3376 theme(
3377   axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
3378   plot.title = element_text(hjust = 0.5)
3379 )
3380
3381 dev.off()
3382
3383 ## Eliminamos variable categorica de intervalos
3384 df_model_v3 <-df_model_v3 |>
3385   dplyr::select( - modi_interval)
3386
3387
3388 ### Seleccion final de variables para entrenamiento ### ----
3389 df_model_v3 <- df_model_v3 |>
3390   dplyr::select(
3391     "Identificador",
3392     "Identificador_lote",
3393     "Precio_adjudicacion",
3394     "N_ofertantes",
3395     "N_lotes",
3396     #"Presupuesto_licitacion",
3397     #"Tipo_de_contrato",
3398     #"Tipo_de_Administracion",
3399     #"Tipo_de_procedimiento",
3400     #"Tramitacion",
3401     #"Tipo_ganador_lote",
3402     "Tipo_de_contrato_Obras",
3403     "Tipo_de_Administracion_Autoridad_local",
3404     "Tipo_de_procedimiento_Abierto",
3405     #"Tipo_de_procedimiento_Simplificado",
3406     "Tramitacion Ordinaria",
3407     #"Tipo_ganador_lote_NIF",
3408     "Plazo_m",
3409     #"Prorroga",
3410     "C_precio_p",
3411     #"C_resto_objetivos_p",
3412     #"C_juicio_valor_p",
3413     "Baja_p",
3414     #"Modi_eco_e", ## Para poder hacer limpieza de casos
3415     "Modi_eco_p",
3416     #"Modi_temp_p",## Para poder hacer limpieza de casos
3417     #"Clasi_max_req",
3418     #"Intervalo_lici_d",
3419     #"Intervalo_vigor_d",
3420     "Intervalo_urge_d",
3421     "N_CPV",
3422     "N_clasi_empresa",
3423     #"Clasi_empresa_main",
3424     #"Codigo Postal", #-- Lo sacamos
3425     #"CPV_main",
3426     #"N_requi",
3427
3428   )
3429
3430
3431 ### 📦 Cargar librerías necesarias ----
3432 library(dplyr)
3433 library(ggplot2)
3434 library(patchwork)
3435
3436 ### 📂 Suponemos que ya existe df_model_v3 con los datos filtrados ----
3437
3438 ### ☑ 1. Nube de puntos completa (orden menor a mayor) ----
3439
3440 df_model_v3_ordenado <- df_model_v3 |>
3441   arrange(Mod_i_eco_p) |>
3442   mutate(caso_ordenado = row_number())

```

```

3443 pdf("v3_Casos_ordinal_vs_Mod_i_p_g.pdf", width = 10, height = 6)
3444 ggplot(df_model_v3_ordenado, aes(x = caso_ordenado, y = Modi_eco_p)) +
3445   geom_point(color = "steelblue", size = 0.3, alpha = 0.6) +
3446   scale_x_continuous(breaks = seq(0, max(df_model_v3_ordenado$caso_ordenado), by = 250)
3447 ) +
3448   scale_y_continuous(breaks = seq(0, ceiling(max(df_model_v3_ordenado$Modi_eco_p)), by
3449 = 5)) +
3450   labs(
3451     title = "Distribución ordinal de los casos según Modi_eco_p",
3452     x = "Casos ordenados (menor a mayor Modi_eco_p)",
3453     y = "Modi_eco_p (%)"
3454   ) +
3455   theme_minimal(base_size = 12) +
3456   theme(
3457     plot.title = element_text(hjust = 0.5),
3458     axis.text.x = element_text(angle = 45, size = 8)
3459   )
3460 dev.off()
3461
3462 ### 📊 2. Gráfico de densidad con líneas de referencia ----
3463
3464 pdf("v3_Densidad_Mod_iEco_con_lineas_g.pdf", width = 10, height = 6)
3465 ggplot(df_model_v3, aes(x = Modi_eco_p)) +
3466   geom_density(fill = "skyblue", alpha = 0.5) +
3467   geom_vline(xintercept = c(10, 15, 20, 50),
3468             color = "red", linetype = "dashed") +
3469   labs(
3470     title = "Densidad de Modi_eco_p con líneas guía",
3471     x = "Modi_eco_p (%)",
3472     y = "Densidad"
3473   ) +
3474   theme_minimal(base_size = 12)
3475 dev.off()
3476
3477
3478 ### 🔍 3. Zooms localizados en torno a 10, 15, 20 y 50 ----
3479
3480 # Función para graficar un zoom individual
3481 graficar_zoom <- function(df, valor_central) {
3482   rango <- c(valor_central - 1.5, valor_central + 1)
3483
3484   df_zoom <- df |>
3485     filter(Modi_eco_p >= rango[1], Modi_eco_p <= rango[2])
3486
3487   ggplot(df_zoom, aes(x = caso_ordenado, y = Modi_eco_p)) +
3488     geom_point(size = 0.2, alpha = 0.6, color = "steelblue") +
3489     scale_y_continuous(breaks = seq(rango[1], rango[2], by = 0.1),
3490                       limits = rango) +
3491     scale_x_continuous(
3492       breaks = scales::pretty_breaks(n = 6)
3493     ) +
3494     labs(
3495       title = paste("Zoom entorno al", valor_central, "%"),
3496       x = "Casos ordenados",
3497       y = "Modi_eco_p"
3498     ) +
3499     theme_minimal(base_size = 10) +
3500     theme(plot.title = element_text(hjust = 0.5))
3501 }
3502
3503 # Crear PDF, una página por gráfico
3504 pdf("v3_Zooms_Mod_iEco_detalle_g.pdf", width = 9, height = 6)
3505 print(graficar_zoom(df_model_v3_ordenado, 10))
3506 print(graficar_zoom(df_model_v3_ordenado, 15))
3507 print(graficar_zoom(df_model_v3_ordenado, 20))
3508 print(graficar_zoom(df_model_v3_ordenado, 50))
3509 dev.off()
3510
3511
3512
3513 ### 📈 4. Derivada discreta ( $\Delta$ Modi_eco_p) ----

```

```

3514 df_model_v3_ordenado <- df_model_v3_ordenado |>
3515   mutate(delta_modi = c(NA, diff(Modi_eco_p)))
3516
3517 pdf("v3_Delta_ModiEco_ordenado_g.pdf", width = 10, height = 6)
3518 ggplot(df_model_v3_ordenado, aes(x = caso_ordenado, y = delta_modi)) +
3519   geom_line(color = "darkorange", linewidth = 0.4) +
3520   geom_hline(yintercept = 0, linetype = "dotted") +
3521   labs(
3522     title = "Variación local ( $\Delta$  Modi_eco_p)",
3523     y = "Cambio respecto al anterior",
3524     x = "Casos ordenados"
3525   ) +
3526   theme_minimal(base_size = 12)
3527 dev.off()
3528
3529
3530
3531 ## Directorio guardar ----
3532
3533 ### CREAMOS DATA V4 ELIMINANDO PICOS -----
3534 # Crear los tramos a eliminar (intervalos cerrados)
3535 tramos_eliminar <- list(
3536   c(9.8, 10.1),
3537   c(14.8, 15.1),
3538   c(19.8, 20.1),
3539   c(49.8, 50.1)
3540 )
3541
3542 # Función para verificar si está dentro de alguno de los intervalos
3543 esta_en_tramos <- function(valor) {
3544   any(sapply(tramos_eliminar, function(rango) valor >= rango[1] && valor <= rango[2]))
3545 }
3546
3547 # Aplicar filtro al dataset original ordenado
3548 df_model_v4 <- df_model_v3_ordenado |>
3549   filter(!sapply(Modi_eco_p, esta_en_tramos))
3550
3551 # Confirmación por consola
3552 cat("Número de casos eliminados:", nrow(df_model_v3_ordenado) - nrow(df_model_v4),
3553     "\n")
3554 cat("Número de casos restantes en df_model_v4:", nrow(df_model_v4), "\n")
3555
3556 ## VISUALIZACION DF_MODEL_V4
3557
3558 ### 📦 Suponemos que ya existe df_model_v3 con los datos filtrados ----
3559
3560 ### ☑ 1. Nube de puntos completa (orden menor a mayor) ----
3561
3562 pdf("v4_Casos_ordinal_vs_Modi_p_g.pdf", width = 10, height = 6)
3563 ggplot(df_model_v4, aes(x = caso_ordenado, y = Modi_eco_p)) +
3564   geom_point(color = "steelblue", size = 0.3, alpha = 0.6) +
3565   scale_x_continuous(breaks = seq(0, max(df_model_v4$caso_ordenado), by = 250)) +
3566   scale_y_continuous(breaks = seq(0, ceiling(max(df_model_v4$Modi_eco_p)), by = 5)) +
3567   labs(
3568     title = "v4_Distribución ordinal de los casos según Modi_eco_p",
3569     x = "Casos ordenados (menor a mayor Modi_eco_p)",
3570     y = "Modi_eco_p (%)"
3571   ) +
3572   theme_minimal(base_size = 12) +
3573   theme(
3574     plot.title = element_text(hjust = 0.5),
3575     axis.text.x = element_text(angle = 45, size = 8)
3576   )
3577 dev.off()
3578
3579
3580 ### 📊 2. Gráfico de densidad con líneas de referencia ----
3581
3582 pdf("v4_Densidad_ModiEco_con_lineas_g.pdf", width = 10, height = 6)
3583 ggplot(df_model_v4, aes(x = Modi_eco_p)) +
3584   geom_density(fill = "skyblue", alpha = 0.5) +
3585   geom_vline(xintercept = c(10, 15, 20, 50),

```

```

3586         color = "red", linetype = "dashed") +
3587     labs(
3588         title = "v4_Densidad de Modi_eco_p con líneas guía",
3589         x = "Modi_eco_p (%)",
3590         y = "Densidad"
3591     ) +
3592     theme_minimal(base_size = 12)
3593 dev.off()
3594
3595
3596 ### Q 3. Zooms localizados en torno a 10, 15, 20 y 50 ----
3597
3598 # Función para graficar un zoom individual
3599 graficar_zoom <- function(df, valor_central) {
3600     rango <- c(valor_central - 1.5, valor_central + 1)
3601
3602     df_zoom <- df |>
3603         filter(Modi_eco_p >= rango[1], Modi_eco_p <= rango[2])
3604
3605     ggplot(df_zoom, aes(x = caso_ordenado, y = Modi_eco_p)) +
3606         geom_point(size = 0.2, alpha = 0.6, color = "steelblue") +
3607         scale_y_continuous(breaks = seq(rango[1], rango[2], by = 0.1),
3608             limits = rango) +
3609         scale_x_continuous(
3610             breaks = scales::pretty_breaks(n = 6)
3611         ) +
3612         labs(
3613             title = paste("v4_Zoom entorno al", valor_central, "%"),
3614             x = "Casos ordenados",
3615             y = "Modi_eco_p"
3616         ) +
3617         theme_minimal(base_size = 10) +
3618         theme(plot.title = element_text(hjust = 0.5))
3619     }
3620
3621 # Crear PDF, una página por gráfico
3622 pdf("v4_Zooms ModiEco_detalle_g.pdf", width = 9, height = 6)
3623 print(graficar_zoom(df_model_v4, 10))
3624 print(graficar_zoom(df_model_v4, 15))
3625 print(graficar_zoom(df_model_v4, 20))
3626 print(graficar_zoom(df_model_v4, 50))
3627 dev.off()
3628
3629
3630
3631 ### @ 4. Derivada discreta ( $\Delta$ Modi_eco_p) ----
3632
3633
3634 pdf("v4_Delta ModiEco_ordenado_g.pdf", width = 10, height = 6)
3635 ggplot(df_model_v4, aes(x = caso_ordenado, y = delta_modi)) +
3636     geom_line(color = "darkorange", linewidth = 0.4) +
3637     geom_hline(yintercept = 0, linetype = "dotted") +
3638     labs(
3639         title = "v4_Variación local ( $\Delta$  Modi_eco_p)",
3640         y = "Cambio respecto al anterior",
3641         x = "Casos ordenados"
3642     ) +
3643     theme_minimal(base_size = 12)
3644 dev.off()
3645
3646 ## Eliminamos variables auxiliares
3647 df_model_v4 <- df_model_v4 |>
3648     dplyr::select(-c(caso_ordenado, delta_modi))
3649
3650
3651
3652 # === División reproducible de los datos en entrenamiento y test === -----
3653
3654 set.seed(321)
3655
3656 n <- nrow(df_model_v4)
3657 train_idx <- sample.int(n, size = floor(0.8 * n)) # Selección aleatoria reproducible
3658

```

```

3659 # Crear conjuntos de entrenamiento y test
3660 df_train_v4 <- df_model_v4[train_idx, ]
3661 df_test_v4 <- df_model_v4[-train_idx, ]
3662
3663 # Comprobamos que la variable objetivo está presente
3664 stopifnot("Modi_eco_p" %in% names(df_model_v4))
3665
3666 # Guardamos los conjuntos y nombres de variables predictoras
3667 saveRDS(list(
3668   df_train = df_train_v4,
3669   df_test = df_test_v4,
3670   vars_predictoras = setdiff(colnames(df_model_v4), "Modi_eco_p")
3671 ), "datos_base_SOM_v4_g.rds")
3672
3673
3674
3675 save(df_train_v4, file="df_train_SOM_v4_g.RData")
3676 write.xlsx(df_train_v4, "df_train_SOM_v4_g.xlsx", colNames = TRUE, rowNames = FALSE)
3677 save(df_test_v4, file="df_test_SOM_v4_g.RData")
3678 write.xlsx(df_test_v4, "df_test_SOM_v4_g.xlsx", colNames = TRUE, rowNames = FALSE)
3679
3680
3681
3682
3683 ##### TRABAJAMOS SOBRE LA CURVA DE DENSIDAD -----
3684 # Crear subconjunto del conjunto de entrenamiento con Modi_eco_p ≤ 40
3685 distrib_train_40 <- df_train_v4 |>
3686   filter(Modi_eco_p <= 40)
3687
3688 # Crear subconjunto del conjunto de test con Modi_eco_p ≤ 40
3689 distrib_test_40 <- df_test_v4 |>
3690   filter(Modi_eco_p <= 40)
3691
3692
3693 # Confirmar cuántos casos hay
3694 cat("Número de casos en distrib_train_40:", nrow(distrib_train_40), "\n")
3695
3696 # Creamos data frame de casos eliminados en los picos, en el train, menores de 40
3697
3698 # Recuperar los casos que fueron eliminados al pasar de v3 a v4
3699 distrib_elimi <- df_model_v3_ordenado |>
3700   filter(sapply(Modi_eco_p, esta_en_tramos)) |>
3701   dplyr::select(-caso_ordenado, -delta_modi)
3702
3703 # Confirmación
3704 cat("Casos eliminados (picos 9.8-10.1, 14.8-15.1, etc.):", nrow(distrib_elimi), "\n")
3705
3706 # Crear subconjunto del conjunto de eliminados con Modi_eco_p ≤ 40
3707 distrib_elimi_40 <- distrib_elimi |>
3708   filter(Modi_eco_p <= 40)
3709
3710
3711 save(distrib_train_40, file="distrib_train_40_v4_g.RData")
3712 write.xlsx(distrib_train_40, "distrib_train_40_v4_g.xlsx", colNames = TRUE, rowNames =
  FALSE)
3713 save(distrib_test_40, file="distrib_test_40_v4_g.RData")
3714 write.xlsx(distrib_test_40, "distrib_test_40_v4_g.xlsx", colNames = TRUE, rowNames =
  FALSE)
3715 save(distrib_elimi, file="distrib_elimi40_v4_g.RData")
3716 write.xlsx(distrib_elimi, "distrib_elimi40_v4_g.xlsx", colNames = TRUE, rowNames =
  FALSE)
3717
3718
3719
3720 ##### FIT de la DISTRIBUCION A OTRAS ##### -----
3721 # Librerías necesarias
3722 library(fitdistrplus)
3723 library(ggplot2)
3724 library(gridExtra)
3725
3726 # === 1. Datos a ajustar ===
3727 x <- distrib_test_40$Modi_eco_p
3728

```

```

3729 # === 2. Ajuste de distribuciones ===
3730 fit_weibull <- fitdist(x, "weibull")
3731 fit_gamma <- fitdist(x, "gamma")
3732 fit_lnorm <- fitdist(x, "lnorm")
3733 fit_norm <- fitdist(x, "norm")
3734
3735 # Rayleigh como caso particular de Weibull con shape = 2
3736 fit_rayleigh <- fitdist(
3737   data = x,
3738   distr = "weibull",
3739   fix.arg = list(shape = 2),
3740   start = list(scale = 10)
3741 )
3742
3743 # Agrupamos todos
3744 ajustes <- list(
3745   Rayleigh = fit_rayleigh,
3746   Weibull = fit_weibull,
3747   Gamma = fit_gamma,
3748   Lognorm = fit_lnorm,
3749   Normal = fit_norm
3750 )
3751
3752 # === 3. Comparativa de ajustes ===
3753 gof <- gofstat(ajustes)
3754 print(gof)
3755
3756 # === 4. Comparativa de densidad con curva suavizada ===
3757
3758 # Secuencia de valores para evaluar densidades teóricas
3759 x_vals <- seq(min(x), max(x), length.out = 500)
3760
3761 # Crear data frame con densidades teóricas
3762 densidades <- data.frame(
3763   x = x_vals,
3764   Rayleigh = VGAM::drayleigh(x_vals, scale = ajustes$Rayleigh$estimate["scale"]),
3765   Weibull = dweibull(x_vals, shape = ajustes$Weibull$estimate["shape"], scale =
ajustes$Weibull$estimate["scale"]),
3766   Gamma = dgamma(x_vals, shape = ajustes$Gamma$estimate["shape"], rate = ajustes$
Gamma$estimate["rate"]),
3767   Lognorm = dlnorm(x_vals, meanlog = ajustes$Lognorm$estimate["meanlog"], sdlog =
ajustes$Lognorm$estimate["sdlog"]),
3768   Normal = dnorm(x_vals, mean = ajustes$Normal$estimate["mean"], sd = ajustes$Normal
$estimate["sd"])
3769 )
3770
3771 # Convertimos a formato largo
3772 library(tidyr)
3773 densidades_long <- densidades |>
3774   pivot_longer(~x, names_to = "Distribucion", values_to = "Densidad")
3775
3776 # Gráfico final
3777 pdf("v4_Distribuciones_ajustadas_con_curva_g.pdf", width = 10, height = 6)
3778
3779 ggplot() +
3780   geom_density(data = distrib_test_40, aes(x = Modi_eco_p),
3781     fill = "gray80", alpha = 0.5, linewidth = 0.3) +
3782   geom_line(data = densidades_long,
3783     aes(x = x, y = Densidad, color = Distribucion),
3784     linewidth = 1.1) +
3785   scale_color_brewer(palette = "Dark2") +
3786   labs(
3787     title = "Comparación: curva de datos vs ajustes teóricos",
3788     x = "Modi_eco_p (%)",
3789     y = "Densidad"
3790   ) +
3791   theme_minimal(base_size = 12) +
3792   theme(
3793     legend.title = element_blank(),
3794     plot.title = element_text(hjust = 0.5)
3795   )
3796
3797 dev.off()

```

```

3798
3799
3800 # === 5. Q-Q plots seguros ===
3801 pdf("v4_QQplots_Distribuciones_ajustadas_g.pdf", width = 10, height = 8)
3802
3803 par(mfrow = c(2, 3)) # 2 filas, 3 columnas
3804
3805 for (i in seq_along(ajustes)) {
3806   qqcomp(
3807     list(ajustes[[i]]), # lista con un solo ajuste
3808     main = paste("Q-Q plot:", names(ajustes)[i]),
3809     line01 = TRUE
3810   )
3811 }
3812
3813 par(mfrow = c(1, 1)) # restaurar parámetros gráficos
3814 dev.off()
3815
3816
3817
3818
3819 par(mfrow = c(1, 1)) # restaurar
3820 dev.off()
3821
3822 # === 1. Extraer métricas del ajuste ===
3823 aic_vals <- gof$aic
3824 bic_vals <- gof$bic
3825 logliks <- sapply(ajustes, function(f) f$loglik)
3826 ks_vals <- gof$ks
3827 cvm_vals <- gof$cvm
3828 ad_vals <- gof$ad
3829
3830 # === 2. Construir tabla resumen ===
3831 tabla_resultados <- data.frame(
3832   Distribucion = names(ajustes),
3833   AIC          = round(aic_vals, 2),
3834   BIC          = round(bic_vals, 2),
3835   LogLik       = round(logliks, 2),
3836   KS           = round(ks_vals, 4),
3837   CvM          = round(cvm_vals, 4),
3838   AD           = round(ad_vals, 4)
3839 )
3840
3841 # === 3. Ordenadas por AIC y BIC ===
3842 tabla_por_aic <- tabla_resultados[order(tabla_resultados$AIC), ]
3843 tabla_por_bic <- tabla_resultados[order(tabla_resultados$BIC), ]
3844
3845 # === 4. Guardar a archivo TXT ===
3846 sink("v4_Resumen_Ajustes_Distribuciones_g.txt")
3847
3848 cat("=== MÉTRICAS DE AJUSTE DE DISTRIBUCIONES ===\n\n")
3849
3850 cat(">> Tabla ordenada por AIC (menor es mejor):\n")
3851 print(tabla_por_aic, row.names = FALSE)
3852
3853 cat("\n>> Tabla ordenada por BIC (menor es mejor):\n")
3854 print(tabla_por_bic, row.names = FALSE)
3855
3856 cat("\n>> Tabla completa sin ordenar:\n")
3857 print(tabla_resultados, row.names = FALSE)
3858
3859 sink()
3860
3861
3862
3863 # === Crear archivo de salida ===
3864 sink("v4_Formulas_Distribuciones_ajustadas_g.txt")
3865
3866 cat("=== FORMULAS DE DISTRIBUCIONES AJUSTADAS ===\n\n")
3867
3868 # === 1. Weibull ===
3869 cat("♦ Weibull Distribution\n")
3870 cat("f(x) = (shape/scale) * (x/scale)^(shape - 1) * exp(-(x/scale)^shape)\n")

```

```

3871 params <- ajustes$Weibull$estimate
3872 cat(sprintf("→ shape = %.4f\n→ scale = %.4f\n\n", params["shape"], params["scale"]))
3873
3874 # === 2. Gamma ===
3875 cat("◇ Gamma Distribution\n")
3876 cat("f(x) = (1 / (Γ(shape) * scale^shape)) * x^(shape - 1) * exp(-x / scale)\n")
3877 params <- ajustes$Gamma$estimate
3878 cat(sprintf("→ shape = %.4f\n→ scale = %.4f\n\n", params["shape"], params["scale"]))
3879
3880 # === 3. Lognormal ===
3881 cat("◇ Lognormal Distribution\n")
3882 cat("f(x) = (1 / (x * sdlog * sqrt(2π))) * exp(-(log(x) - meanlog)^2 / (2 *
sdlog^2))\n")
3883 params <- ajustes$Lognorm$estimate
3884 cat(sprintf("→ meanlog = %.4f\n→ sdlog = %.4f\n\n", params["meanlog"], params[
"sdlog"]))
3885
3886 # === 4. Normal ===
3887 cat("◇ Normal Distribution\n")
3888 cat("f(x) = (1 / (sd * sqrt(2π))) * exp(-(x - mean)^2 / (2 * sd^2))\n")
3889 params <- ajustes$Normal$estimate
3890 cat(sprintf("→ mean = %.4f\n→ sd = %.4f\n\n", params["mean"], params["sd"]))
3891
3892 # === 5. Rayleigh ===
3893 cat("◇ Rayleigh Distribution (Weibull con shape = 2)\n")
3894 cat("f(x) = (x / scale^2) * exp(-(x^2) / (2 * scale^2))\n")
3895 params <- ajustes$Rayleigh$estimate
3896 cat(sprintf("→ shape (fijo) = 2.0000\n→ scale = %.4f\n\n", params["scale"]))
3897
3898 sink()
3899
3900 ### AJUSTE FINO WEIBULL ### -----
3901
3902 # a) Ajuste vía optimización directa personalizada (con optim())
3903 # 1. Función de log-verosimilitud negativa para Weibull
3904 loglik_weibull <- function(params, data) {
3905   shape <- params[1]
3906   scale <- params[2]
3907
3908   if (shape <= 0 || scale <= 0) return(Inf)
3909
3910   -sum(dweibull(data, shape = shape, scale = scale, log = TRUE))
3911 }
3912
3913 # 2. Ejecutar optimización
3914 resultado_opt <- optim(
3915   par = c(shape = 1, scale = 10),
3916   fn = loglik_weibull,
3917   data = x,
3918   method = "L-BFGS-B",
3919   lower = c(0.01, 0.01)
3920 )
3921
3922 # 3. Extraer parámetros óptimos
3923 parametros_optimos <- resultado_opt$par
3924 parametros_optimos
3925
3926 # b) Ajuste por minimización de la distancia de Kolmogorov-Smirnov (KS)
3927
3928 ks_distancia <- function(params, data) {
3929   shape <- params[1]
3930   scale <- params[2]
3931
3932   if (shape <= 0 || scale <= 0) return(Inf)
3933
3934   ks <- suppressWarnings(ks.test(data, "pweibull", shape = shape, scale = scale))
3935   ks$statistic
3936 }
3937
3938 res_ks <- optim(
3939   par = c(1, 10),
3940   fn = ks_distancia,
3941   data = x,

```



```

3942     method = "L-BFGS-B",
3943     lower = c(0.01, 0.01)
3944 )
3945
3946 res_ks$par
3947
3948 # c) Ajuste bayesiano de la Weibull con rstan o brms
3949
3950 # Usando brms (más sencillo que rstan)
3951 library(brms)
3952
3953 # Modelar Weibull bayesiano
3954 modelo_bayes <- brm(
3955   bf(Modi_eco_p ~ 1),
3956   data = distrib_test_40,
3957   family = weibull(),
3958   chains = 4, iter = 2000, cores = 4
3959 )
3960
3961 summary(modelo_bayes)
3962
3963
3964 ### 📊 COMPARACIÓN FINAL DE LOS AJUSTES FINOS DE WEIBULL ###
3965
3966 # 1. Recolectar los ajustes
3967 weibull_mle <- ajustes$Weibull$estimate
3968 weibull_optim <- parametros_optimos
3969 weibull_ks <- res_ks$par
3970 names(weibull_ks) <- c("shape", "scale")
3971
3972 # 2. Obtener shape y scale del ajuste bayesiano
3973 shape_bayes <- posterior_summary(modelo_bayes, variable = "shape")["shape", "Estimate"]
3974
3975 scale_bayes <- exp(fixef(modelo_bayes)["Intercept", "Estimate"])
3976 weibull_bayes <- c(shape = shape_bayes, scale = scale_bayes)
3977
3978 # 3. Tabla resumen con verificación
3979 df_compara <- data.frame(
3980   Metodo = c("Inicial (MLE)", "Optimización LogLik", "Minimización KS", "Bayesiano"),
3981   Shape = c(
3982     weibull_mle["shape"],
3983     weibull_optim["shape"],
3984     weibull_ks["shape"],
3985     weibull_bayes["shape"]
3986   ),
3987   Scale = c(
3988     weibull_mle["scale"],
3989     weibull_optim["scale"],
3990     weibull_ks["scale"],
3991     weibull_bayes["scale"]
3992   )
3993 )
3994
3995 # 4. Guardar la tabla como .TXT
3996 write.table(
3997   df_compara,
3998   "v4_Ajustes_Weibull_Finos_g.txt",
3999   sep = "\t", row.names = FALSE, quote = FALSE
4000 )
4001
4002 # 5. Filtrar métodos con parámetros válidos
4003 metodos_validos <- complete.cases(df_compara[, c("Shape", "Scale")])
4004 df_compara_valid <- df_compara[metodos_validos, ]
4005
4006 # 6. Calcular las curvas solo para métodos válidos
4007 x_vals <- seq(min(x), max(x), length.out = 500)
4008
4009 densidades_df <- data.frame(x = x_vals)
4010
4011 for (i in seq_len(nrow(df_compara_valid))) {
4012   metodo <- df_compara_valid$Metodo[i]
4013   shape <- df_compara_valid$Shape[i]

```

```

4014     scale <- df_compara_valid$Scale[i]
4015
4016     densidades_df[[metodo]] <- dweibull(x_vals, shape = shape, scale = scale)
4017 }
4018
4019 # 7. Convertir a formato largo
4020 densidades_long <- densidades_df |>
4021   pivot_longer(~x, names_to = "Metodo", values_to = "Densidad")
4022
4023 # 8. Crear gráfico PDF
4024 pdf("v4_Comparacion_Ajustes_Finos_Weibull_g.pdf", width = 10, height = 6)
4025 ggplot() +
4026   geom_density(
4027     data = distrib_test_40,
4028     aes(x = Modi_eco_p),
4029     fill = "gray80", alpha = 0.5, linewidth = 0.3
4030   ) +
4031   geom_line(
4032     data = densidades_long,
4033     aes(x = x, y = Densidad, color = Metodo),
4034     linewidth = 1.1
4035   ) +
4036   scale_color_brewer(palette = "Set1") +
4037   labs(
4038     title = "Comparación de ajustes finos de Weibull",
4039     x = "Modi_eco_p (%)",
4040     y = "Densidad"
4041   ) +
4042   theme_minimal(base_size = 12) +
4043   theme(
4044     legend.title = element_blank(),
4045     plot.title = element_text(hjust = 0.5)
4046   )
4047 dev.off()
4048
4049
4050 ### 📊 TABLA FINAL DE COMPARACIÓN DE AJUSTES FINOS WEIBULL ###
4051
4052 # --- 1. Cálculo de métricas para cada ajuste ---
4053
4054 # Datos de entrada
4055 x_vals_eval <- distrib_test_40$Modi_eco_p
4056
4057 # MLE (ya calculado)
4058 loglik_mle <- ajustes$Weibull$loglik
4059 ks_mle <- ks.test(x_vals_eval, "pweibull",
4060                  shape = ajustes$Weibull$estimate["shape"],
4061                  scale = ajustes$Weibull$estimate["scale"])$statistic
4062 aic_mle <- AIC(ajustes$Weibull)
4063 bic_mle <- BIC(ajustes$Weibull)
4064
4065 # Optimización loglik
4066 loglik_optim <- -resultado_opt$value
4067 ks_optim <- ks.test(x_vals_eval, "pweibull",
4068                   shape = resultado_opt$par["shape"],
4069                   scale = resultado_opt$par["scale"])$statistic
4070 aic_optim <- -2 * loglik_optim + 2 * 2
4071 bic_optim <- -2 * loglik_optim + log(length(x_vals_eval)) * 2
4072
4073 # Minimización KS
4074 ks_ks <- ks.test(x_vals_eval, "pweibull",
4075                shape = weibull_ks["shape"],
4076                scale = weibull_ks["scale"])$statistic
4077 loglik_ks <- sum(dweibull(x_vals_eval, shape = weibull_ks["shape"], scale = weibull_ks
4078 ["scale"], log = TRUE))
4079 aic_ks <- -2 * loglik_ks + 2 * 2
4080 bic_ks <- -2 * loglik_ks + log(length(x_vals_eval)) * 2
4081
4082 # Bayesiano
4083 loglik_bayes <- log_lik(modelo_bayes)
4084 mean_loglik_bayes <- sum(colMeans(loglik_bayes))
4085 ks_bayes <- ks.test(
4086   x_vals_eval,

```

```

4086     "pweibull",
4087     shape = posterior_summary(modelo_bayes, variable = "shape")[1],
4088     scale = exp(fixef(modelo_bayes)["Intercept", "Estimate"])
4089 )$statistic
4090 # AIC/BIC bayesianos no aplicables directamente
4091
4092 # --- 2. Construcción de tabla resumen ---
4093
4094 tabla_final <- data.frame(
4095   Método = c("Inicial (MLE)", "Optimización LogLik", "Minimización KS", "Bayesiano"),
4096   LogLik = round(c(loglik_mle, loglik_optim, loglik_ks, mean_loglik_bayes), 2),
4097   KS      = round(c(ks_mle, ks_optim, ks_ks, ks_bayes), 4),
4098   AIC     = round(c(aic_mle, aic_optim, aic_ks, NA), 2),
4099   BIC     = round(c(bic_mle, bic_optim, bic_ks, NA), 2)
4100 )
4101
4102 # Nota: El warning de 'ties should not be present' en ks.test()
4103 # es esperado ya que los datos tienen valores repetidos.
4104 # Como sólo usamos el estadístico KS (D) para comparar ajustes sobre los mismos datos,
4105 # la comparación sigue siendo válida.
4106 # pero la p-valor (si la calcularas) no sería confiable
4107 # porque el procedimiento asume una distribución continua estricta.
4108
4109 # --- 3. Guardar como .txt ---
4110 sink("v4_Comparativa_Metricas_Ajustes_Weibull_g.txt")
4111 cat("=== COMPARACIÓN DE MÉTRICAS ENTRE AJUSTES FINOS DE WEIBULL ===\n\n")
4112 print(tabla_final, row.names = FALSE)
4113 sink()
4114
4115 ## Limpiamos
4116 rm(list=ls())
4117
4118
4119 # -----
4120 # SCRIPT 8 SOM_v5 -----
4121 # -----
4122
4123
4124 ## Directorio GUARDAR ----
4125
4126 setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
4127 atom/PLACSP/_final_conjunto_v4")
4128
4129
4130
4131 # === 1.1 Cargar datos base ===
4132 datos_base <- readRDS("datos_base_SOM_v4_g.rds")
4133
4134 df_train <- datos_base$df_train
4135 df_test  <- datos_base$df_test
4136 vars     <- datos_base$vars_predictoras
4137
4138
4139 # === 1.2. Filtro extra datos base modelos ===
4140
4141 ### CREAMOS DATA V5 ELIMINANDO PICOS (Aumentamos el filtro de 50)
4142 # Crear los tramos a eliminar (intervalos cerrados)
4143 tramos_eliminar <- list(
4144   c(9.8, 10.1),
4145   c(14.8, 15.1),
4146   c(19.8, 20.1),
4147   c(47.5, 50.5)
4148 )
4149
4150
4151 # Función para verificar si está dentro de alguno de los intervalos
4152 esta_en_tramos <- function(valor) {
4153   any(sapply(tramos_eliminar, function(rango) valor >= rango[1] && valor <= rango[2]))
4154 }
4155
4156 # Aplicar filtro al dataset original train
4157 df_train_v5 <- df_train |>

```

```

4158     filter(!sapply(Modi_eco_p, esta_en_tramos))
4159
4160 # Aplicar filtro al dataset original test
4161 df_test_v5 <- df_test |>
4162     filter(!sapply(Modi_eco_p, esta_en_tramos))
4163
4164 #Seleccionamos casos eliminados
4165
4166 # Aplicar filtro al dataset original train
4167 df_elim_train_v5 <- df_train |>
4168     filter(sapply(Modi_eco_p, esta_en_tramos))
4169 df_elim_test_v5 <- df_test |>
4170     filter(sapply(Modi_eco_p, esta_en_tramos))
4171 # Unimos
4172 df_elim_v5<-rbind(df_elim_train_v5,df_elim_test_v5)
4173
4174
4175 # Confirmación por consola
4176 cat("Número de casos eliminados entrenamiento (extra pico 50):", nrow(df_train) - nrow
(df_train_v5), "\n")
4177 cat("Número de casos restantes en df_train_v5:", nrow(df_train_v5), "\n")
4178 cat("Número de casos restantes en df_test_v5:", nrow(df_test_v5), "\n")
4179
4180 # === 1.3. Guardamos datos base de entrenamiento y test v5 === #
4181
4182 saveRDS(list(
4183     df_train = df_train_v5,
4184     df_test = df_test_v5,
4185     vars_predictoras = setdiff(colnames(df_train_v5), "Modi_eco_p")
4186 ), "datos_base_SOM_v5_g.rds")
4187
4188
4189 save(df_train_v5, file="df_train_SOM_v5_g.RData")
4190 write.xlsx(df_train_v5, "df_train_SOM_v5_g.xlsx", colNames = TRUE, rowNames = FALSE)
4191 save(df_test_v5, file="df_test_SOM_v5_g.RData")
4192 write.xlsx(df_test_v5, "df_test_SOM_v5_g.xlsx", colNames = TRUE, rowNames = FALSE)
4193 save(df_elim_v5, file="df_elim_v5_g.RData")
4194 write.xlsx(df_elim_v5, "df_elim_v5_g.xlsx", colNames = TRUE, rowNames = FALSE)
4195
4196
4197
4198 # ==== Unir archivos de eliminados por filas y guardar ====
4199
4200 suppressPackageStartupMessages({ library(readxl); library(dplyr); library(stringr) })
4201
4202
4203 # Rutas reales (busca primero en _final_conjunto_v2)
4204 f1 <- ("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4/distrib_elimi40_v4_g.xlsx")
4205 f2 <- ("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos
atom/PLACSP/_final_conjunto_v4/df_elim_v5_g.xlsx")
4206
4207 # Lectura (primera hoja por defecto). Añadimos columna de procedencia para
trazabilidad
4208 x1 <- readxl::read_excel(f1, guess_max = 100000) |> mutate(Fuente = basename(f1))
4209 x2 <- readxl::read_excel(f2, guess_max = 100000) |> mutate(Fuente = basename(f2))
4210
4211 # Unir por filas; dplyr alineará columnas distintas rellenando con NA donde falten
casos_eliminados_picos <- dplyr::bind_rows(x1, x2)
4213 casos_eliminados_picos <- casos_eliminados_picos [,-17] #Quitamos el df de fuente
4214
4215 # Recolocar IDs si existen (Identificador / Identificador_lote) al principio
4216 if (exists("relocate_id_cols_first")) {
4217     casos_eliminados_picos <- relocate_id_cols_first(casos_eliminados_picos)
4218 }
4219
4220 #Ordenamos por Modi_eco_p
4221 # Ascendente (NAs al final)
4222 casos_eliminados_picos <- casos_eliminados_picos %>%
4223     dplyr::arrange(is.na(Mod_i_eco_p), Modi_eco_p)
4224
4225 # Guardado casos eliminados simple:
4226 write.xlsx(casos_eliminados_picos, "casos_eliminados_picos_g.xlsx", colNames = TRUE,

```

```

rowNames = FALSE)
4227 save(casos_eliminados_picos, file="casos_eliminados_picos_g.RData")
4228
4229
4230
4231 # ==== Cruce: casos_eliminado_picos □ bd_civil_sin_NA (por Identificador ) ====
4232
4233 suppressPackageStartupMessages({ library(dplyr); library(stringr) })
4234
4235 ## Cargamos datos -----
4236
4237 load("bd_civil_sin_NA_v2.RData")
4238
4239 # ==== Cruce: casos_eliminado(s)_picos □ bd_civil_sin_NA (por Identificador +
Identificador_lote) ====
4240
4241 suppressPackageStartupMessages({ library(dplyr); library(stringr) })
4242
4243
4244 # 1) Definir claves y validar presencia
4245 req_keys <- c("Identificador")
4246
4247 if (!all(req_keys %in% names(casos_eliminados_picos))) {
4248   stop("Faltan claves en el df de picos: ", paste(setdiff(req_keys, names(
casos_eliminados_picos)), collapse = ", "))
4249 }
4250 if (!exists("bd_civil_sin_NA")) stop("'bd_civil_sin_NA' no está cargado en el
entorno.")
4251
4252 if (!all(req_keys %in% names(bd_civil_sin_NA))) {
4253   stop("Faltan claves en 'bd_civil_sin_NA': ", paste(setdiff(req_keys, names(
bd_civil_sin_NA)), collapse = ", "))
4254 }
4255
4256
4257 # 2) Renombrar TODAS las columnas de bd_civil_sin_NA salvo las claves con sufijo _c
4258 to_suffix <- setdiff(names(bd_civil_sin_NA), req_keys)
4259 bd_civil_sin_NA_c <- bd_civil_sin_NA %>%
4260   dplyr::rename_with(~ paste0(., "_c"), .cols = dplyr::all_of(to_suffix))
4261
4262 # 3) Forzar tipo character en claves para un matching robusto
4263 casos_df2 <- casos_eliminados_picos %>%
4264   mutate(
4265     Identificador = as.character(Identificador),
4266   )
4267 bd_civil_sin_NA_c2 <- bd_civil_sin_NA_c %>%
4268   mutate(
4269     Identificador = as.character(Identificador),
4270   )
4271
4272 # 4) LEFT JOIN: primero columnas de casos_df, después las *_c
4273 casos_eliminado_picos_completo <- casos_df2 %>%
4274   dplyr::left_join(bd_civil_sin_NA_c2, by = req_keys)
4275
4276 # 5) Recolocar IDs al principio si tienes el helper
4277 if (exists("relocate_id_cols_first")) {
4278   casos_eliminado_picos_completo <- relocate_id_cols_first(
casos_eliminado_picos_completo)
4279 }
4280
4281 # 6) Guardar (usa tu register_output si está disponible)
4282 if (exists("register_output")) {
4283   register_output("casos_eliminado_picos_completo", casos_eliminado_picos_completo,
4284     save = TRUE, formats = c("rdata", "xlsx", "csv"))
4285 } else {
4286   suppressPackageStartupMessages(library(openxlsx))
4287   out_dir <- if (exists("paths")) paths$processed_dir else getwd()
4288   openxlsx::write.xlsx(casos_eliminado_picos_completo,
4289     file.path(out_dir, "casos_eliminado_picos_completo.xlsx"),
4290     colNames = TRUE, rowNames = FALSE)
4291   message("[saved] ", file.path(out_dir, "casos_eliminado_picos_completo.xlsx"))
4292 }
4293

```

```

4294
4295 # ==== Cruce: df_train_v5 / df_test_v5 □ bd_civil_sin_NA_v2(por Identificador) ====
4296
4297 suppressPackageStartupMessages({ library(dplyr); library(stringr) })
4298
4299 # 0) Cargar base de referencia si no estuviera en memoria
4300 if (!exists("bd_civil_sin_NA")) {
4301   load("bd_civil_sin_NA_v2.RData") # debe contener un objeto llamado bd_civil_sin_NA
4302 }
4303 if (!exists("bd_civil_sin_NA")) stop("'bd_civil_sin_NA' no está disponible tras el
load().")
4304
4305 # 1) Definir clave y validar
4306 req_keys <- c("Identificador")
4307
4308 # 2) Renombrar TODAS las columnas de bd_civil_sin_NA salvo la clave con sufijo _c
(para evitar colisiones)
4309 to_suffix <- setdiff(names(bd_civil_sin_NA), req_keys)
4310 bd_civil_sin_NA_c <- bd_civil_sin_NA %>%
4311   dplyr::rename_with(~ paste0(., "_c"), .cols = dplyr::all_of(to_suffix))
4312
4313 # 3) Helper para cruzar de forma idéntica train/test
4314 .cruzar_por_identificador <- function(df_in, name_out) {
4315   if (!all(req_keys %in% names(df_in))) {
4316     stop("Faltan claves en '", deparse(substitute(df_in)), "': ",
4317         paste(setdiff(req_keys, names(df_in)), collapse = ", "))
4318   }
4319
4320   # Forzar tipo character en la clave para evitar mismatches
4321   df_left <- df_in %>%
4322     dplyr::mutate(Identificador = as.character(Identificador))
4323   df_right <- bd_civil_sin_NA_c %>%
4324     dplyr::mutate(Identificador = as.character(Identificador))
4325
4326   res <- df_left %>%
4327     dplyr::left_join(df_right, by = req_keys)
4328
4329   # Recolocar IDs al principio si tienes el helper
4330   if (exists("relocate_id_cols_first")) {
4331     res <- relocate_id_cols_first(res)
4332   }
4333
4334   # Guardar
4335   if (exists("register_output")) {
4336     register_output(name_out, res, save = TRUE, formats = c("rdata", "xlsx", "csv"))
4337   } else {
4338     # Fallback simple
4339     suppressPackageStartupMessages(library(openxlsx))
4340     out_dir <- if (exists("paths")) paths$processed_dir else getwd()
4341     openxlsx::write.xlsx(res, file.path(out_dir, paste0(name_out, ".xlsx")),
4342         colNames = TRUE, rowNames = FALSE)
4343     message("[saved] ", file.path(out_dir, paste0(name_out, ".xlsx")))
4344   }
4345
4346   invisible(res)
4347 }
4348
4349 # 4) Ejecutar para TRAIN y TEST (exige que existan df_train_v5 y df_test_v5)
4350 if (!exists("df_train_v5")) stop("'df_train_v5' no existe en el entorno.")
4351 if (!exists("df_test_v5")) stop("'df_test_v5' no existe en el entorno.")
4352
4353 df_train_v5_completo <- .cruzar_por_identificador(df_train_v5, "df_train_v5_completo")
4354 df_test_v5_completo <- .cruzar_por_identificador(df_test_v5, "df_test_v5_completo")
4355
4356
4357
4358
4359
4360
4361

```