Bayesian Spatial and Temporal Machine Learning for Texas Diabetes Surveillance

Mohit Chhaparia

Contents

1 Data Description/Definitions:

Diagnosed Diabetes: BRFSS respondents who answered "yes" to the question, "Has a doctor ever told you that you have diabetes?" Women who reported diabetes only during pregnancy were excluded from this group.

Dentists Rate: Number of Dentists per 100,000 Population.

Dentists Number: Number of dentists in the county.

Primary Care Physician: PCPs are defined as non-Federal doctors of medicine (M.D.) and doctors of osteopathy (D.O.) providing direct patient care who practice principally in one of the four primary care specialties-general or family practice, general internal medicine, pediatrics, and obstetrics and gynecology.

Primary Care Physician Rate: Number of primary care physicians (PCPs) per 100,000 Population.

Primary Care Physician Number: Number of primary care physicians (PCPs) in the county.

No Health Insurance: Percentage of population aged 18 years and older with no health insurance.

Mental Health Providers: Mental health providers are defined as psychiatrists, psychologists, licensed clinical social workers, counselors, marriage and family therapists, mental health providers that treat alcohol and other drug abuse, and advanced practice nurses specializing in mental health care.

Mental Health Providers Rate: Number of mental health providers per 100,000 Population.

Mental Health Providers Number: Number of Mental Health Providers per county.

2 Document Setup:

2.1 File-Setup

```
cache_dir <- "Nearest-Neighbor---Gaussian-Process-Model_cache/latex/"

get_chunk_labels_in_order <- function() {
  labs <- knitr::all_labels()
  if(!is.null(labs)) return(labs)

  if(requireNamespace("rstudioapi" , quietly = TRUE) && rstudioapi::isAvailable()) {
    ctx <- rstudioapi::getSourceEditorContext()
    lines <- ctx$contents

    pat <- "^```\\{r\\s*([^,}]*)"
    labs <- sub(pat , "\\1" , grep(pat , lines , value = TRUE))
    labs <- labs[labs != ""]
    return(labs)
  }

  character(0)
}

chunk_cache_files <- list.files(cache_dir , pattern = "\\.RData$" , full.names = FALSE)

chunk_labels <- get_chunk_labels_in_order()
chunk_labels <- sub(paste0("\\" , "," , ".*$") , "" , chunk_labels)

label_to_cache <- lapply(chunk_labels , function(lab) {
  chunk_cache_files[grepl(paste0("^" , lab , "_") , chunk_cache_files)]
})
names(label_to_cache) <- chunk_labels

load_prev_chunk_caches <- function(current_label) {
  idx <- match(current_label , chunk_labels)
```

```
  if(is.na(idx) || idx <= 1) return(invisible(NULL))

  for(j in seq_len(idx - 1)) {
    lab <- chunk_labels[j]
    cf <- label_to_cache[[lab]]

    if(!length(cf)) next

    for (f in cf) {
      knitr::load_cache(f , path = cache_dir , envir = .GlobalEnv , lazy  = TRUE)
    }
  }

  invisible(NULL)
}

knitr::opts_chunk$set(echo = TRUE , warning = FALSE , message = FALSE ,
                      comment = '#>' , cache = TRUE , chunk_level = 2 ,
                      cache.path = cache_dir)

knitr::knit_hooks$set(chunk = function(x , options) {
  if (isTRUE(options$echo)) {
    level <- if (!is.null(options$chunk_level)) options$chunk_level else 2
    hashes <- paste(rep("#" , level) , collapse = "")
    heading <- sprintf("\n\n%s `%s`\n\n" , hashes , options$label)
    paste0(heading , x)
  } else x
})
```

2.2 Loading-Libraries

```
# load_prev_chunk_caches("Loading-Libraries")

library(ggplot2)
library(gridExtra)
library(viridis)
library(tidyverse)
library(dplyr)
library(skimr)
library(sf)
library(spdep)
library(tigris)
library(tmap)
library(spNNGP)
library(parallel)
library(spBayes)
```

2.3 ggplot-setup

```
# load_prev_chunk_caches("ggplot-setup")

hw <- theme_gray() +
  theme(
    plot.title = element_text(hjust = 0.5) ,
    plot.subtitle = element_text(hjust = 0.5) ,
    plot.caption = element_text(hjust = -0.5) ,

    strip.background = element_rect(fill = rgb(0.9 , 0.95 , 1) ,
                                    colour = gray(0.5) , linewidth = 0.2) ,

    panel.border = element_rect(fill = FALSE , colour = gray(0.7)) ,
    panel.grid.minor.y = element_blank() ,
    panel.grid.minor.x = element_blank() ,
    panel.spacing.x = unit(0.10 , "cm") ,
    panel.spacing.y = unit(0.05 , "cm") ,

    axis.ticks = element_blank() ,
    axis.text = element_text(colour = "black") ,
    axis.text.y = element_text(margin = margin(0 , 3 , 0 , 3)) ,
    axis.text.x = element_text(margin = margin(-1 , 0 , 3 , 0))
  )

theme_set(hw)
```

3 Exploratory Data Analysis:

3.1 Load-Full-Dataset

```
# load_prev_chunk_caches("Load-Full-Dataset")

load("Combined Data.RData")

dat <- Stack
dat$Urban <- as.factor(dat$Urban)
dat <- dat %>% select(-c(X , population_estimate , FoodEnvIndex2021 , MHI_2021 ,
                         Food.Insecurity.2021.Percentage , NoInternet_1721))
```

3.2 Data-Summaries

```
# load_prev_chunk_caches("Data-Summaries")

g <- dat
nb <- poly2nb(g)
lw <- nb2listw(nb , style = "W")
```

```
glimpse(g)
```

```
#> Rows: 254
#> Columns: 9
#> $ GEOID                                    <chr> "48355", "48215", "48061", "4~
#> $ NAME                                     <chr> "Nueces County, Texas", "Hida~
#> $ County                                   <chr> "Nueces County", "Hidalgo Cou~
#> $ Diagnosed_2021                           <dbl> 8.8, 9.0, 9.0, 9.0, 8.9, 9.0,~
#> $ No.Health.Insurance.2017.2021.Percentage <dbl> 21.2, 32.0, 30.1, 29.7, 20.0,~
#> $ Obesity..Percentage.                     <dbl> 36.5, 41.3, 39.6, 36.2, 33.3,~
#> $ CIP_2021                                 <dbl> 24.9, 37.6, 32.6, 29.0, 22.1,~
#> $ Urban                                    <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
#> $ geometry                                 <MULTIPOLYGON [°]> MULTIPOLYGON (((~
```

```
skim_without_charts(g)
```

Table 1: Data summary

Name	g
Number of rows	254
Number of columns	9
Column type frequency:	
character	4
factor	1
numeric	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
GEOID	0	1	5	5	0	254	0
NAME	0	1	17	27	0	254	0
County	0	1	10	20	0	254	0
geometry	0	1	444	33393	0	254	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Urban	0	1	FALSE	2	1: 128, 0: 126

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
Diagnosed_2021	0	1	8.94	0.36	7.5	8.80	8.9	9.2	9.8

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
No.Health.Insurance.2017.2021.Percentage	0	1	22.10	4.72	4.9	19.40	21.8	24.5	46.3
Obesity..Percentage.	0	1	33.11	2.33	25.2	31.60	33.0	34.4	43.1
CIP_2021	0	1	22.71	7.46	6.9	18.02	22.3	27.3	57.1

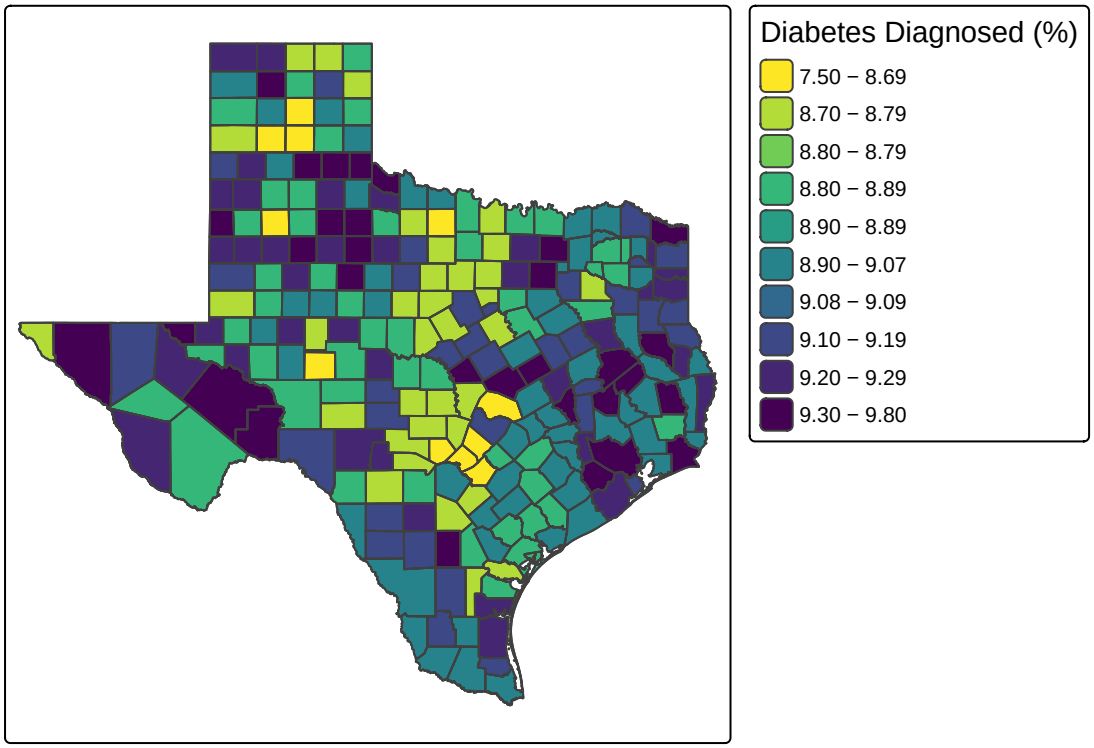
The working dataset g contains **254 Texas counties** with **9 variables**. Four variables are stored as character IDs/labels (GEOID, county name, county label, and the spatial geometry), one variable (Urban) is a **factor with two levels** indicating whether a county is classified as urban or rural, and the remaining **four variables are numeric health and socio-economic indicators**.

The skim summary shows that the dataset is **complete with no missing values** for any variable, so all 254 counties can be used in subsequent analyses without additional imputation or filtering. Among the numeric variables, diabetes diagnosis in 2021 (Diagnosed_2021) is fairly homogeneous across counties (mean around 9% with a narrow range ~7.5–9.8). By contrast, **no health insurance, obesity, and children-in-poverty percentages show much larger variation**, indicating substantial heterogeneity in underlying risk factors across counties.

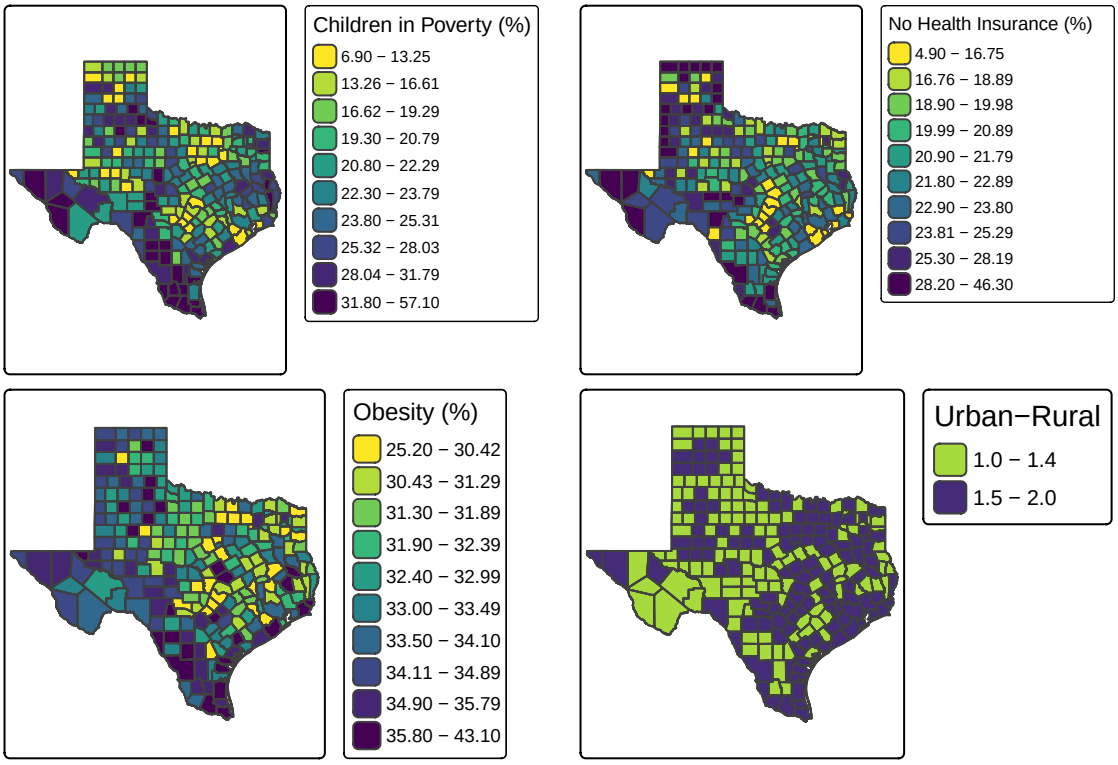
3.3 Spatial-Plots

```
# load_prev_chunk_caches("Spatial-Plots")

p1 <- tm_shape(g) + tm_polygons(fill = "Diagnosed_2021" ,
                                fill.scale = tm_scale_intervals(style = "quantile" ,
                                                                n = 10 , values = "-viridis") ,
                                fill.legend = tm_legend(title = "Diabetes Diagnosed (%)")) +
  tm_borders()

p2 <- tm_shape(g) + tm_polygons(fill = "CIP_2021" ,
                                fill.scale = tm_scale_intervals(style = "quantile" ,
                                                                n = 10 , values = "-viridis") ,
                                fill.legend = tm_legend(title = "Children in Poverty (%)")) +
  tm_borders()

p3 <- tm_shape(g) + tm_polygons(fill = "No.Health.Insurance.2017.2021.Percentage" ,
                                fill.scale = tm_scale_intervals(style = "quantile" ,
                                                                n = 10 , values = "-viridis") ,
                                fill.legend = tm_legend(title = "No Health Insurance (%)")) +
  tm_borders()

p4 <- tm_shape(g) + tm_polygons(fill = "Obesity..Percentage." ,
                                fill.scale = tm_scale_intervals(style = "quantile" ,
                                                                n = 10 , values = "-viridis") ,
                                fill.legend = tm_legend(title = "Obesity (%)")) +
  tm_borders()

p5 <- tm_shape(g) + tm_polygons(fill = "Urban" ,
                                fill.scale = tm_scale_intervals(style = "quantile" ,
                                                                n = 2 , values = "-viridis") ,
```

```
                              fill.legend = tm_legend(title = "Urban-Rural")) +
  tm_borders()

tmap_arrange(p1 , ncol = 1 , nrow = 1)
```




```
tmap_arrange(p2 , p3 , p4 , p5 , ncol = 2 , nrow = 2)
```

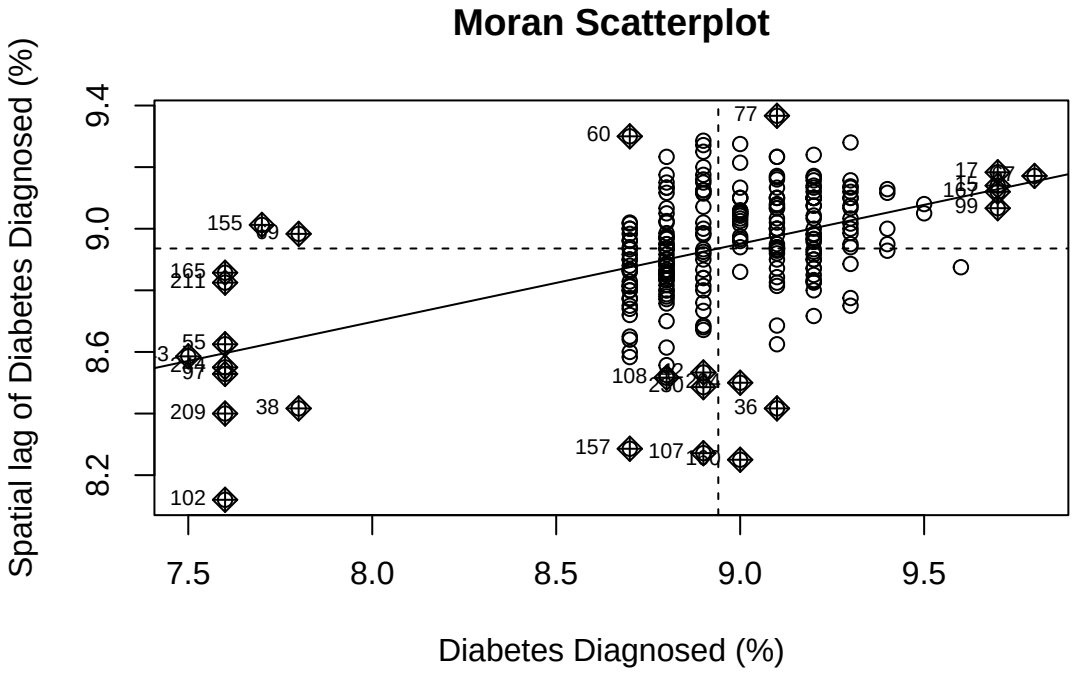


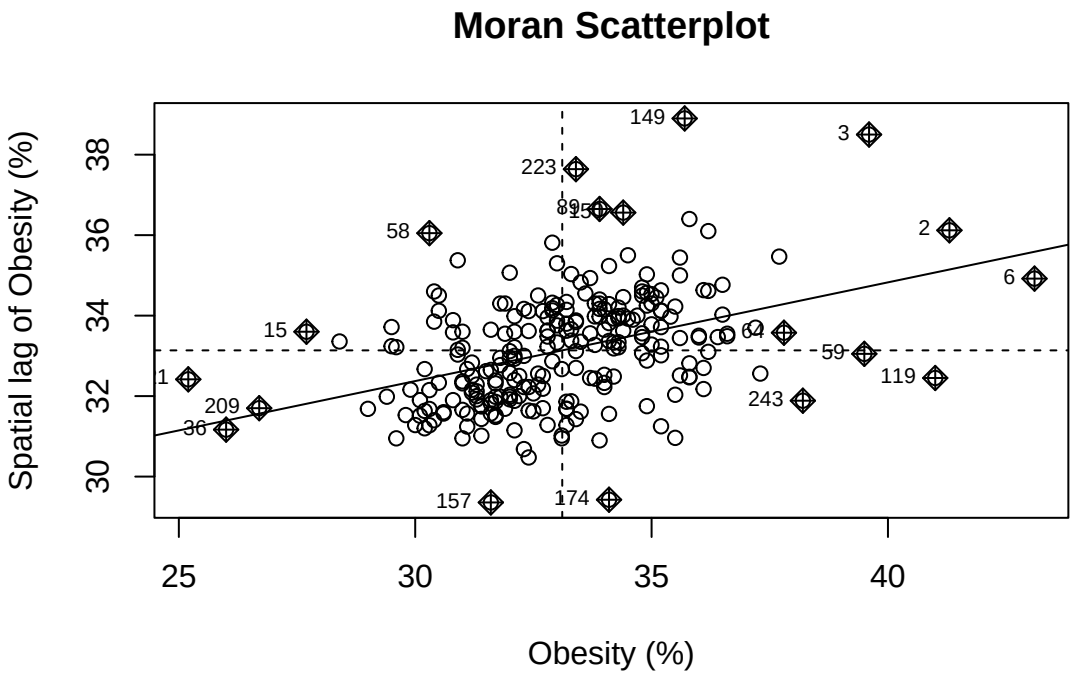
The maps show clear spatial heterogeneity in all county-level health indicators across Texas. Diabetes diagnosed (%) varies substantially by county, with clusters of higher prevalence in several southern and eastern counties and relatively lower values in many central and western counties. Children in poverty (%) and the percentage without health insurance display broadly similar patterns, suggesting that socioeconomic disadvantage is geographically concentrated and overlaps with areas of poorer diabetes outcomes. Obesity (%) is also spatially uneven and tends to be elevated in many of the same regions, reinforcing the link between obesity, poverty, and diabetes risk. The Urban–Rural map highlights that most Texas counties are classified as rural, with urban counties concentrated around major metro areas.

3.4 Global-Moran-Test

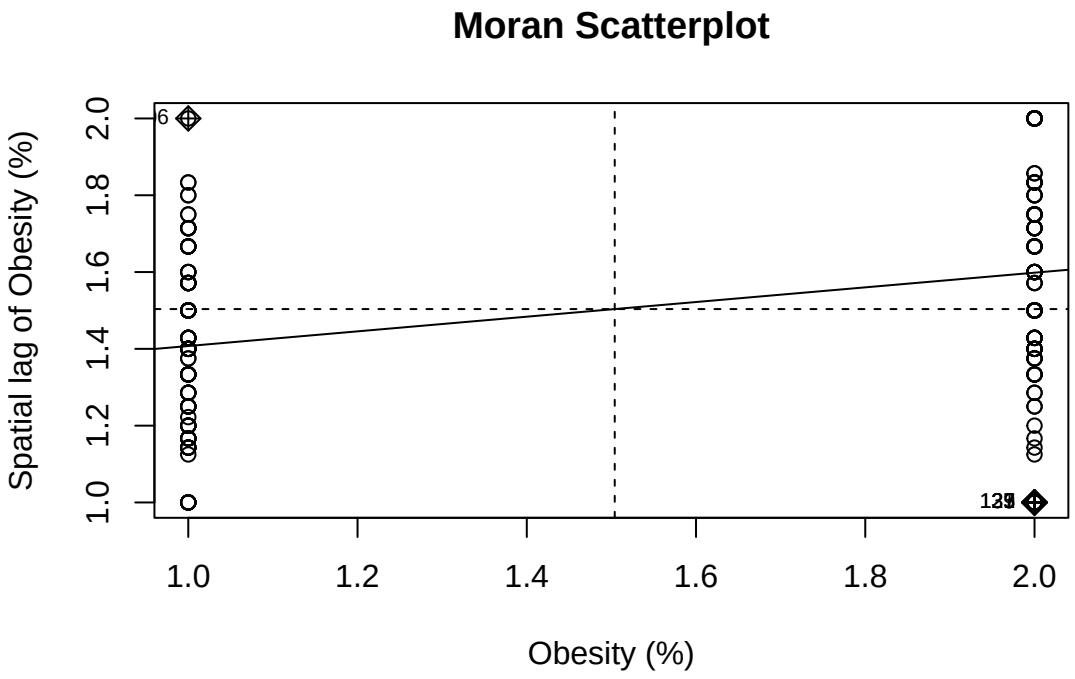
```
# load_prev_chunk_caches("Global-Moran-Test")
#=================================================================
# Use following code to verify and include only numeric columns
#=================================================================
temp <- sapply(g , is.numeric)
numeric_cols <- c("Diagnosed_2021" , "Obesity..Percentage." ,
                  "No.Health.Insurance.2017.2021.Percentage" , "Urban" , "CIP_2021")
num_cols <- c("Diagnosed_2021" , "Obesity..Percentage." ,
                  "No.Health.Insurance.2017.2021.Percentage" , "Urban" , "CIP_2021")
col_names <- c('Diabetes Diagnosed (%)' , 'Obesity (%)' , 'No Health Insurance (%)' ,
                'Urban or Rural' , 'Children in Poverty (%)')
ylab_names <- paste("Spatial lag of" , col_names)

#=====================================================================
# Helps look for Spatial Autocorrelation at a Global level
```

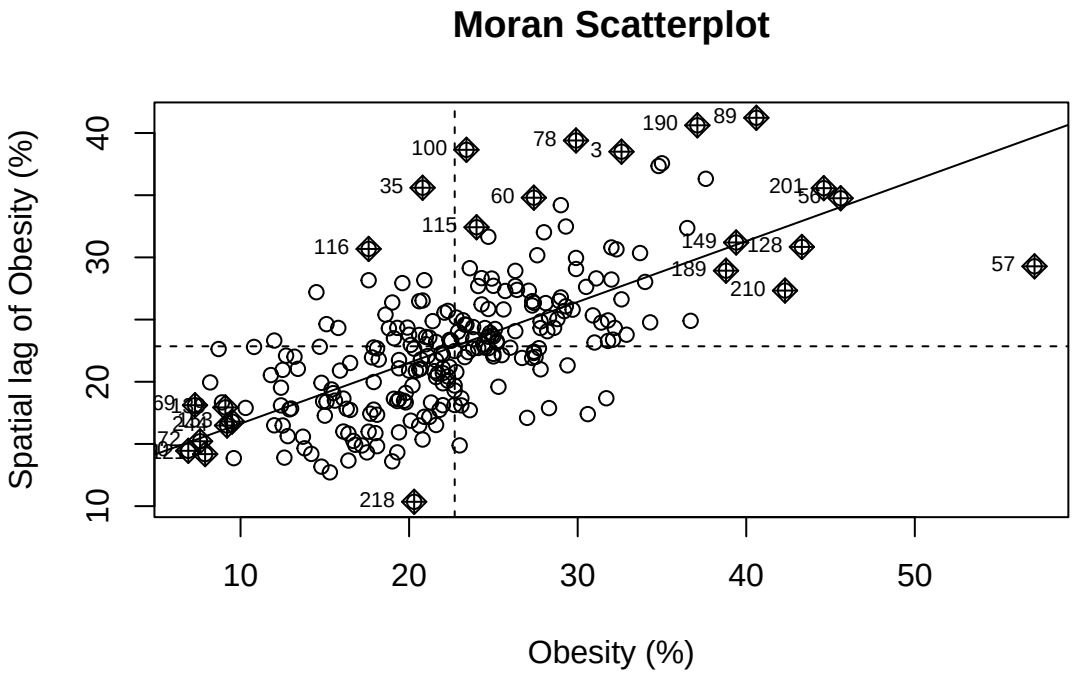
```r
# Interpretation:
# Moran's I > 0 + p-value < 0.05 -> Positive Spatial Clustering
# Moran's I < 0 -> Spatial Dispersion
# Moran's I approximately 0 -> Spatial Randomness
# p-value > 0.05 -> No significant spatial autocorrelation
# Isolates -> Number of counties that have no spatial neighbors
# Note: This function does not work with missing values, hence a few
# rows were dropped as per the requirement.
#=============================================================================
moran_results <- list()
moran_plots <- list()
i <- 1

for(v in num_cols){
  subset_vec <- !is.na(g[[v]])

  nb_sub <- subset(nb , subset_vec , zero.policy = TRUE)
  lw_sub <- nb2listw(nb_sub , style = "W" , zero.policy = TRUE)

  # To ensure we do not have counties with no neighbors
  isolates <- sum(sapply(nb_sub , function(x) length(x) == 0))

  mt <- moran.test(as.numeric(g[[v]][subset_vec]) , lw_sub , zero.policy = TRUE ,
                   na.action = na.exclude)
  moran_results[[v]] <- data.frame(variable = col_names[i] ,
                                   Moran_I = mt$estimate[["Moran I statistic"]] ,
                                   Expected = mt$estimate[["Expectation"]] ,
                                   Var = mt$estimate[["Variance"]] ,
                                   p_value = mt$p.value , isolates = isolates)

  moran_plots[[i]] <- moran.plot(as.numeric(g[[v]][subset_vec]) , lw_sub ,
                                 xlab = col_names[i] , ylab = ylab_names[i] ,
                                 main = 'Moran Scatterplot' , plot = TRUE)
  i <- 1 + 1
}
```


Moran Scatterplot
Spatial lag of Diabetes Diagnosed (%)
Diabetes Diagnosed (%)
7.5
8.0
8.5
9.0
9.5
8.2
8.6
9.0
9.4
77
60
155
165
211
55
209
38
102
157
107
36
99


Moran Scatterplot
Spatial lag of Obesity (%)
Obesity (%)
25
30
35
40
30
32
34
36
38
149
3
223
58
2
6
15
59
119
1
209
36
243
157
174


Moran Scatterplot
Spatial lag of Obesity (%)
Obesity (%)
15
20
25
30
35
10
20
30
40
149
2
60
34
62
184
100
9
240
121
218


Moran Scatterplot
Spatial lag of Obesity (%)
Obesity (%)
1.0
1.2
1.4
1.6
1.8
2.0
6




```
moran_table <- bind_rows(moran_results) %>% arrange(desc(Moran_I))
moran_table
```

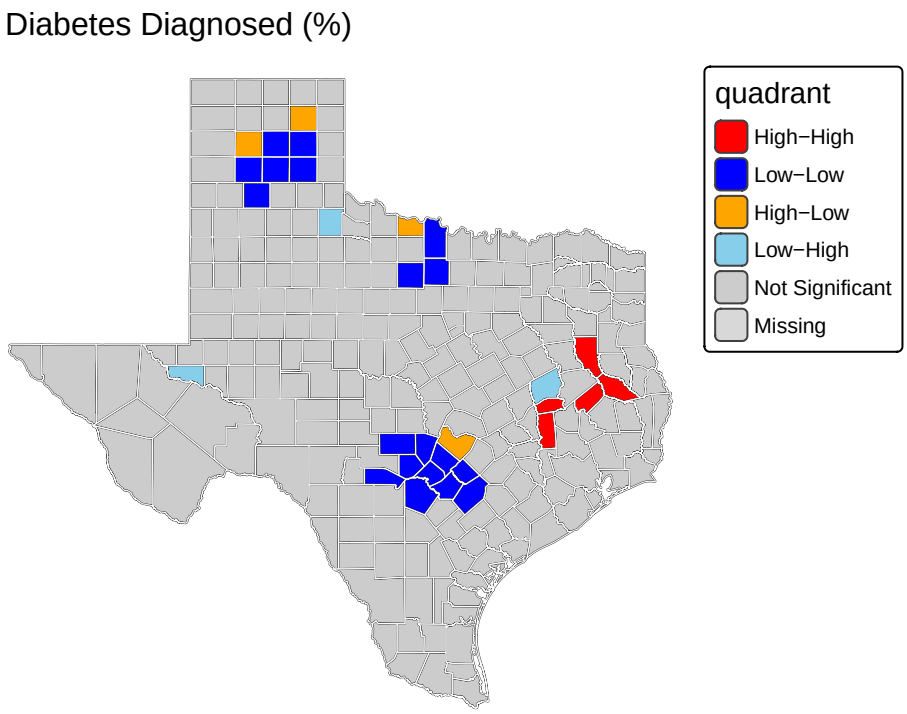
```
#>                 variable   Moran_I     Expected         Var      p_value
#> 1            Obesity (%) 0.4890996 -0.003952569 0.001409051 1.037715e-39
#> 2 Diabetes Diagnosed (%) 0.2534346 -0.003952569 0.001387390 2.420643e-12
#> 3            Obesity (%) 0.2452933 -0.003952569 0.001406232 1.499716e-11
#> 4            Obesity (%) 0.2448477 -0.003952569 0.001401327 1.502583e-11
#> 5            Obesity (%) 0.1910158 -0.003952569 0.001429980 1.262501e-07
#>   isolates
#> 1        0
#> 2        0
#> 3        0
#> 4        0
#> 5        0
```

The global Moran's results show clear positive spatial autocorrelation for all variables considered. CIP_2021 has the largest Moran's I (0.49, $p < 0.001$), indicating strong spatial clustering of child poverty; diabetes diagnosed (%) and obesity (%) have Moran's I around 0.25, consistent with moderate clustering, while no-health-insurance (%) and the urban–rural indicator have smaller but still positive and highly significant values, suggesting weaker clustering rather than spatial randomness. The Moran scatterplots reinforce this pattern: the fitted lines all slope upward and there are no isolates, meaning each county's value tends to resemble the average of its neighbors rather than behaving independently in space.

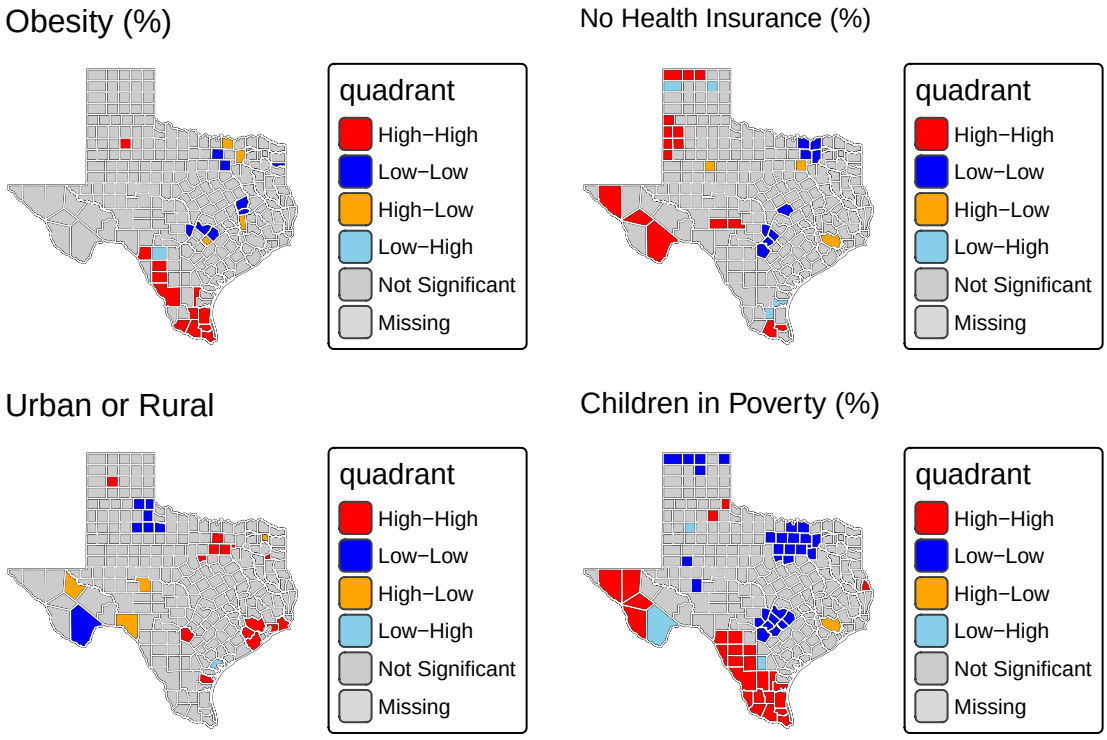
3.5 Local-Moran-Test

```
# load_prev_chunk_caches("Local-Moran-Test")
#================================================================================
# Helps look for autocorrelation and outliers at a local level
# Interpretation:
# High-High: County has high value surrounded by high values (hot spot)
# Low-Low: County has low value surrounded by low values (cold spot)
# High-Low or Low-High: Spatial outlier (county differs from its neighbors)
# Not significant: No local spatial autocorrelation
# Note: This function does not work with missing values, hence a few
# rows were dropped as per the requirement.
#================================================================================

i <- 1
tm_list <- list()

for (v in num_cols) {

  subset_vec <- !is.na(g[[v]])
  nb_sub <- subset(nb , subset_vec , zero.policy = TRUE)
  lw_sub <- nb2listw(nb_sub , style = "W" , zero.policy = TRUE)

  li <- localmoran(as.numeric(g[[v]][subset_vec]) , lw_sub , zero.policy = TRUE)
  g_temp <- g[subset_vec , ]

  g_temp$Ii <- li[ , 1]
  g_temp$E.Ii <- li[ , 2]
  g_temp$Var.Ii <- li[ , 3]
  g_temp$Z.Ii <- li [, 4]
  g_temp$p.Ii <- li[ , 5]

  g_temp$z_var <- scale(as.numeric(g[[v]][subset_vec]) , center = TRUE ,
                        scale = TRUE)
  g_temp$lag_z <- lag.listw(lw_sub , g_temp$z_var , zero.policy = TRUE)

  g_temp$quadrant <- ifelse(g_temp$p.Ii >= 0.05 , "Not Significant" ,
                            ifelse(g_temp$z_var > 0 & g_temp$lag_z > 0 ,
                                   "High-High" ,
                                   ifelse(g_temp$z_var < 0 & g_temp$lag_z < 0 ,
                                          "Low-Low" ,
                                          ifelse(g_temp$z_var > 0 & g_temp$lag_z < 0 ,
                                                 "High-Low" ,
                                                 ifelse(g_temp$z_var < 0 & g_temp$lag_z > 0 ,
                                                        "Low-High" , NA)))))

  g_temp$quadrant <- factor(g_temp$quadrant ,
                            levels = c("High-High" , "Low-Low" , "High-Low" ,
                                       "Low-High" , "Not Significant"))

  tm_plot <- tm_shape(g_temp) + tm_polygons("quadrant" ,
                                            palette = c("red" , "blue" ,
                                                        "orange" , "skyblue" , "grey80")) +
```

```
    tm_borders(lwd = 0.3 , col = "white") + tm_layout(legend.outside = FALSE ,
                                                      main.title = col_names[i] ,
                                                      main.title.size = 1 , frame = FALSE)

  tm_list[[i]] <- tm_plot
  if(i == 1) print(tm_plot)

  i <- i + 1
}
```




```
tmap_arrange(tm_list[[2]] , tm_list[[3]] , tm_list[[4]] , tm_list[[5]] ,
             ncol = 2 , nrow = 2)
```



The local Moran analysis decomposes the global autocorrelation into county-level "hot" and "cold" spots for each variable. Across all five variables most Texas counties are grey ("Not Significant"), meaning their values are not markedly different from their neighbors once spatial dependence is taken into account. The interesting structure is concentrated in a relatively small set of counties that fall into the four local-cluster types.

- **Diabetes Diagnosed (%):** Significant *High–High* clusters are visible in a pocket of South-Eastern Texas border counties and *High-Low and Low-Low* clusters are visible in a small group in the North, indicating local hot spots of higher diabetes prevalence surrounded by similarly high counties. A few *Low–Low* clusters appear in parts of Central / North Texas, suggesting localized areas of comparatively lower diabetes burden.

- **Obesity (%):** The obesity map shows several southern counties classified as *High–High*, forming a band of elevated obesity risk. Some *Low–Low* counties appear in the north, indicating local areas where both the county and its neighbors have relatively lower obesity levels.

- **No Health Insurance (%):** Uninsured rates exhibit pronounced *High–High* clustering along the western border of Texas, consistent with regional pockets of elevated uninsurance. A small number of *Low–Low* counties occur in more central regions, indicating localized zones of better insurance coverage.

- **Urban or Rural:** For the binary urban indicator, *High–High* and *Low–Low* patterns essentially pick out metropolitan cores versus predominantly rural regions. Some *High–Low* or *Low–High* counties at the urban–rural fringe behave as spatial outliers, where a county's classification differs from its neighbors.

- **Children in Poverty (%):** Child poverty shows *High–High* clusters in South and Western Texas, highlighting localized concentrations of high poverty. A few *Low–Low* clusters appear in Central and Northern parts of the state.

Overall, the local Moran maps confirm that the strongest clustering occurs for children in poverty, no health insurance, and obesity, with diabetes diagnosis and urban status showing smaller but still interpretable pockets of local clustering and a handful of spatial outliers.

3.6 Spatial-Diagnostics-Global-Local-Moran-with-Interpretation

```r
# load_prev_chunk_caches("Spatial-Diagnostics-Global-Local-Moran-with-Interpretation")

var_names <- c("Diagnosed_2021" , "Obesity..Percentage." ,
               "No.Health.Insurance.2017.2021.Percentage" , "Urban" , "CIP_2021")
col_names <- c('Diabetes Diagnosed (%)' , 'Obesity (%)' , 'No Health Insurance (%)' ,
                'Urban or Rural' , 'Children in Poverty (%)')

get_moran_stats <- function(var_name , outcome_name = "Diagnosed_2021" , data, alpha = 0.05) {
  x <- st_drop_geometry(data)[[var_name]]
  y <- st_drop_geometry(data)[[outcome_name]]
  keep <- complete.cases(x , y)
  x <- as.numeric(x[keep])
  y <- y[keep]

  geom_sub <- st_geometry(data)[keep]
  nb_sub <- poly2nb(geom_sub , queen = TRUE)
  listw_sub <- nb2listw(nb_sub , style = "W" , zero.policy = TRUE)

  #====================
  # Global Moran's
  #====================
  mt <- moran.test(x , listw = listw_sub , alternative = "greater")
  I_val <- mt$estimate[["Moran I statistic"]]
  p_val <- mt$p.value

  #====================
  # Local Moran's
  #====================
  loc <- localmoran(x , listw_sub)
  sig <- loc[ , "Pr(z != E(Ii))"] < alpha
  cluster <- rep("Not Sig" , length(x))
  cluster[sig & x > mean(x)] <- "High-High"
  cluster[sig & x < mean(x)] <- "Low-Low"
  cluster[sig & x > mean(x)] <- "High-Low"
  cluster[sig & x < mean(x)] <- "Low-High"
  cluster_tab <- table(factor(cluster , levels = c("High-High" , "Low-Low" ,
                                                   "High-Low" , "Low-High" , "Not Sig")))

  data.frame(Variable = var_name , Moran_I = round(I_val , 3) ,
             p_value = signif(p_val , 3) , HH_count = cluster_tab["High-High"] ,
             LL_count = cluster_tab["Low-Low"] , HL_count = cluster_tab["High-Low"] ,
             LH_count = cluster_tab["Low-High"] , NS_count = cluster_tab["Not Sig"] ,
             stringsAsFactors = FALSE)
}

moran_results <- lapply(var_names , function(v) get_moran_stats(v , data = g))
moran_df <- bind_rows(bind_rows(moran_results))
```

```r
moran_df <- moran_df %>%
  mutate(p_value = ifelse(p_value < 0.001 , "<0.001" , as.character(p_value)) ,
         Interpretation = case_when(Moran_I > 0.4 ~ "Strong spatial clustering" ,
                                    Moran_I > 0.25 ~ "Moderate clustering" ,
                                    Moran_I > 0.1 ~ "Weak clustering" ,
                                    TRUE ~ "None")) %>% arrange(desc(Moran_I))

rownames(moran_df) <- NULL
print(moran_df)
```

```
#>                                  Variable Moran_I p_value HH_count LL_count
#> 1                                CIP_2021   0.489  <0.001        0        0
#> 2                           Diagnosed_2021   0.253  <0.001        0        0
#> 3                     Obesity..Percentage.   0.245  <0.001        0        0
#> 4 No.Health.Insurance.2017.2021.Percentage   0.245  <0.001        0        0
#> 5                                   Urban   0.191  <0.001        0        0
#>   HL_count LH_count NS_count           Interpretation
#> 1       27       33      194 Strong spatial clustering
#> 2        9       22      223       Moderate clustering
#> 3       16       10      228           Weak clustering
#> 4       19       13      222           Weak clustering
#> 5       18        8      228           Weak clustering
```

This section combines the global and local Moran's statistics to summarise spatial autocorrelation for all county-level covariates relative to diabetes. All five variables have positive and highly significant Moran's I ($p < 0.001$), confirming that each shows some degree of positive spatial clustering rather than spatial randomness. Children in Poverty (CIP_2021) has the strongest spatial signal (Moran's I 0.49), classified as **strong spatial clustering**, while Diabetes Diagnosed (%) shows **moderate clustering** (I 0.25). Obesity (%), No Health Insurance (%), and Urban/Rural status all have Moran's I between 0.19 and 0.25 and are labelled as **weak clustering**, indicating milder but still meaningful spatial dependence.

The local Moran diagnostics show that, for all variables, most counties fall into the "Not Significant" class, with very few high-high or low-low clusters and more observations flagged as spatial outliers (High-Low or Low-High). Overall, these diagnostics confirm that the main variables of interest exhibit non-random spatial structure, with poverty and diabetes having the most pronounced clustering, and the remaining predictors contributing weaker but non-negligible spatial patterns.

4 Nearest Neighbor Models:

The exploratory maps and Moran's diagnostics showed clear positive spatial autocorrelation in diabetes, poverty, obesity, health insurance, and urban–rural status. This means neighboring counties tend to look more similar than distant ones, so a standard regression model with independent errors is not appropriate. Nearest–neighbor spatial models are a way to capture this dependence while remaining computationally feasible for many locations: instead of modelling every pair of counties, they approximate the spatial process using only a small set of nearby neighbors for each county. In Section 4 we therefore move from purely descriptive diagnostics to fitting nearest–neighbor Gaussian process models that explicitly account for the spatial structure revealed in the earlier analysis.

4.1 spNNGP:

The **spNNGP** framework implements nearest–neighbor Gaussian process (NNGP) regression in a fully Bayesian way. It uses a sparse neighbor structure and flexible covariance functions (exponential, spherical, Gaussian, Matérn) to model spatially correlated residuals while remaining scalable to our 254 Texas counties. By fitting spNNGP models we can compare different covariance families, quantify uncertainty in the spatial effects, and evaluate predictive performance (RMSE, (R^2), Adj.(R^2)) for diabetes prevalence. Section 4.1 focuses on these spNNGP models for understanding how much of the variation in diabetes is explained after accounting for spatial correlation.

4.1.1 Model Development:

```
# load_prev_chunk_caches("NNGP-Model-Parameter-Initialization")

#==============================================================================
# Nearest Neighbor - Gaussian Process (MCMC) with train/test split
# Response: Diagnosed_2021
# Covariates: No.Health.Insurance.2017.2021.Percentage,
#             Obesity..Percentage., CIP_2021, Urban (factor 0/1)
# Coordinates: taken from 'geometry'.
#==============================================================================
coords_mat <- as.matrix(st_coordinates(st_centroid(st_geometry(dat))))

required_cols <- c("Diagnosed_2021" , "No.Health.Insurance.2017.2021.Percentage" ,
                   "Obesity..Percentage." , "CIP_2021" , "Urban")
dat <- dat[ , c(required_cols , "geometry")]

keep_records <- complete.cases(st_drop_geometry(dat)[ , required_cols])
dat <- dat[keep_records , ]
coords_mat <- coords_mat[keep_records , , drop = FALSE]

set.seed(928)
n <- nrow(dat)
tr_idx <- sample.int(n , size = floor(0.8 * n))
te_idx <- setdiff(seq_len(n) , tr_idx)

train <- dat[tr_idx , ]
test <- dat[te_idx , ]
coords_train <- coords_mat[tr_idx , , drop = FALSE]
coords_test <- coords_mat[te_idx , , drop = FALSE]

mm_formula <- as.formula("~ Urban + `No.Health.Insurance.2017.2021.Percentage` +
                         `Obesity..Percentage.` + `CIP_2021`")
```

```
X_train <- model.matrix(mm_formula , data = st_drop_geometry(train))
X_test <- model.matrix(mm_formula , data = st_drop_geometry(test))
y_train <- train$Diagnosed_2021
y_test <- test$Diagnosed_2021

n.neighbors <- 15
n.batch <- 1000
batch.length <- 5
n.samples <- n.batch * batch.length
n.threads <- parallel::detectCores(logical = TRUE) - 2

spNNGP_formula <- as.formula("Diagnosed_2021 ~ Urban + `No.Health.Insurance.2017.2021.Percentage` +
                             `Obesity..Percentage.` + `CIP_2021`")

start_keep <- floor(0.5 * n.samples) + 1
end_keep <- n.samples
```

4.1.1.1 NNGP-Model-Parameter-Initialization

In this step the problem is set up for the NNGP model. County centroids are extracted from the geometry and used as the spatial coordinates. The working dataset keeps only the outcome *Diagnosed_2021*, the four covariates (No Health Insurance, Obesity, Children in Poverty, and Urban indicator), and geometry, dropping any rows with missing values. The data are then randomly split into an 80% training set and 20% test set. A linear model formula is defined with *Diagnosed_2021* as the response and the four county-level risk factors as predictors, and corresponding design matrices (X_{train}), (X_{test}) and response vectors (y_{train}, y_{test}) are created. Finally, key NNGP settings are chosen: the number of nearest neighbors (15), MCMC batch structure (1000 batches $\times$ 5 iterations $=$ 5000 samples), number of CPU threads, and the index range of posterior samples to keep for prediction.

```
# load_prev_chunk_caches("NNGP-Model-Grid")

cov_models <- c("exponential" , "spherical" , "gaussian" , "matern")

priors_list <- list(
  list(sigma.sq.IG = c(2 , 1) , tau.sq.IG = c(2 , 1) , phi.Unif = c(1e-3 , 1) , nu.Unif = c(0.3 , 2.5)),
  list(sigma.sq.IG = c(1.5 , 0.5) , tau.sq.IG = c(1.5 , 0.5) , phi.Unif = c(1e-4 , 1) , nu.Unif = c(0.3

tuning_list <- list(list(sigma.sq = 0 , tau.sq = 0 , phi = 0 , nu = 0) ,
                    list(sigma.sq = 0.1 , tau.sq = 0.1 , phi = 0.1 , nu = 0.1))


phi_start <- c(0.05 , 0.2)
tau_start <- c(0 , 0.1)
sigma_start <- c(0.5 , 1)
nu_start <- c(0.5 , 2.5)

n.neighbors.list <- c(10 , 15 , 25)

param_grid <- expand.grid(cov_model = cov_models , priors_id = seq_along(priors_list) ,
```

```
                          tuning_id = seq_along(tuning_list) , s0 = sigma_start ,
                          t0 = tau_start , p0 = phi_start , nu0 = nu_start ,
                          n.neighbors = n.neighbors.list)

param_grid$nu0 <- ifelse(param_grid$cov_model == 'matern' , param_grid$nu0 , NA)
param_grid <- unique(param_grid)

print(head(param_grid))
```

4.1.1.2 NNGP-Model-Grid

```
#>     cov_model priors_id tuning_id  s0 t0   p0 nu0 n.neighbors
#> 1 exponential         1         1 0.5  0 0.05  NA          10
#> 2   spherical         1         1 0.5  0 0.05  NA          10
#> 3    gaussian         1         1 0.5  0 0.05  NA          10
#> 4      matern         1         1 0.5  0 0.05 0.5          10
#> 5 exponential         2         1 0.5  0 0.05  NA          10
#> 6   spherical         2         1 0.5  0 0.05  NA          10
```

```
print(dim(param_grid))
```

```
#> [1] 480   8
```

Here a hyper-parameter grid is constructed to explore alternative NNGP specifications. Four covariance families (exponential, spherical, Gaussian, and Matérn) are considered. For each family, combinations of prior settings for the variance components ((σ^2), (τ^2)) and range parameter (ϕ), Metropolis tuning scales, starting values for (σ^2), (τ^2), (ϕ), and several neighbor sizes (10, 15, 25) are formed using expand.grid. For the Matérn model an additional smoothness parameter (ν) is included, while it is set to NA for the other covariance types. After removing duplicates, the grid contains 480 distinct parameter configurations, which will be used to fit and compare a large set of NNGP models in the subsequent multi-threaded search.

```
# load_prev_chunk_caches("NNGP-Model-Multi-Thread-Function")

run_spNNGP_combo <- function(cov_model , priors , tuning , s0 , t0 , p0 , nu0 , n.neighbors , seed = 928
  set.seed(seed)

  # cat("NNGP Reached Fine 1\n" , file = "debug_log_NNGP.txt" , append = TRUE)

  if(cov_model == "matern") {
    starting <- list(beta = rep(0 , ncol(X_train)) , sigma.sq = s0 , tau.sq = t0 ,
                     phi = p0 , nu = nu0)
  } else {
    starting <- list(beta = rep(0 , ncol(X_train)) , sigma.sq = s0 , tau.sq = t0 , phi = p0)
  }
  # cat("NNGP Reached Fine 2\n" , file = "debug_log_NNGP.txt" , append = TRUE)
```

```
  fit <- spNNGP(formula = spNNGP_formula , data = train , coords = coords_train ,
                starting = starting , method = "response" , n.neighbors = n.neighbors ,
                tuning = tuning , priors = priors , cov.model = cov_model ,
                n.omp.threads = n.threads , n.samples = n.samples , verbose = FALSE ,
                search.type = 'cb')

  # cat("NNGP Reached Fine 3\n" , file = "debug_log_NNGP.txt" , append = TRUE)
  start_keep <- floor(0.5 * n.samples) + 1
  end_keep <- n.samples

  # cat("NNGP Reached Fine 4\n" , file = "debug_log_NNGP.txt" , append = TRUE)
  pred <- predict(obj = fit , X.0 = X_test , coords.0 = coords_test ,
                  sub.sample = list(start = start_keep , end = end_keep , thin = 1) ,
                  n.omp.threads = 2 , verbose = FALSE)

  # cat("NNGP Reached Fine 5\n" , file = "debug_log_NNGP.txt" , append = TRUE)
  y_pred_mean <- apply(pred$p.y.0 , 1 , mean)
  rmse <- sqrt(mean((y_test - y_pred_mean) ^ 2))
  r2 <- cor(y_test , y_pred_mean) ^ 2
  n <- length(y_test)
  k <- ncol(X_test) - 1
  adj_r2 <- 1 - (1 - r2) * ((n - 1) / (n - k - 1))

  # cat("NNGP Reached Fine 6\n" , file = "debug_log_NNGP.txt" , append = TRUE)
  beta_mean <- colMeans(fit$p.beta.samples)
  sigma_mean <- mean(fit$p.theta.samples[ , "sigma.sq"])
  tau_mean <- mean(fit$p.theta.samples[ , "tau.sq"])
  phi_mean <- mean(fit$p.theta.samples[ , "phi"])
  if(cov_model == "matern") {
    nu_mean <- mean(fit$p.theta.samples[ , "nu"])
  } else{
    nu_mean <- NA_real_
  }

  # cat("NNGP Reached Fine 7\n" , file = "debug_log_NNGP.txt" , append = TRUE)
  tibble(cov_model = cov_model ,
         prior_id = paste(priors$sigma.sq.IG[1] , priors$sigma.sq.IG[2] , sep = "-") ,
         tuning_id = paste(tuning$sigma.sq , tuning$tau.sq , sep = "-") ,
         sigma_start = s0 , tau_start = t0 , phi_start = p0 , nu_start = nu0 , rmse = rmse , r2 = r2 ,
         adj_r2 = adj_r2 , sigma_mean = sigma_mean , tau_mean = tau_mean , phi_mean = phi_mean ,
         nu_mean = nu_mean , beta_mean = I(list(beta_mean)) , neighbors = n.neighbors)
}
```

4.1.1.3 NNGP-Model-Multi-Thread-Function

In this section we wrap the NNGP model fitting into a single helper function that can be called repeatedly in parallel. The function takes a specific covariance model, prior, tuning values, starting values, and number of neighbors, and then fits an NNGP regression to the training data using MCMC. For each fit it produces out-of-sample predictions on the test set, computes RMSE and adjusted R^2, and stores posterior summaries of the key parameters (σ^2), (τ^2), (ϕ) (and (ν) for Matérn). The function returns all of these metrics in a compact tibble, so that different hyper-parameter combinations can be compared in a consistent way.

```r
# load_prev_chunk_caches("NNGP-Model-Multi-Thread-Execution")

# cat("NNGP\n" , file = "debug_log_NNGP.txt" , append = FALSE)

cl <- parallel::makeCluster(n.threads)
parallel::clusterExport(cl , c("train" , "test" , "coords_train" , "coords_test" , "X_train" ,
                      "X_test" , "y_train" , "y_test" , "spNNGP_formula" ,
                      "n.neighbors" , "n.samples" , "n.threads" , "run_spNNGP_combo" ,
                      "priors_list" , "tuning_list" , "param_grid") , envir = environment())

invisible(
  parallel::clusterEvalQ(cl, {
    suppressMessages({
      library(spNNGP)
      library(tibble)
      library(dplyr)
      library(sf)
      library(spdep)
      library(spBayes)
    })
    NULL
  })
)

safe_run <- function(i) {
  row <- param_grid[i , ]
  tryCatch({
    message(sprintf("[Thread %d] Running: cov = %s | prior = %d | tuning = %d |
                    s0 = %.2f | t0 = %.2f | p0 = %.2f | nu0 = %.2f | n.neighbor = %d" , i ,
                    row$cov_model , row$priors_id , row$tuning_id , row$s0 , row$t0 ,
                    row$p0 , row$nu0 , row$n.neighbors))

    res <- run_spNNGP_combo(cov_model = row$cov_model , priors = priors_list[[row$priors_id]] ,
                            tuning = tuning_list[[row$tuning_id]] , s0 = row$s0 , t0 = row$t0 ,
                            p0 = row$p0 , nu0 = row$nu0 , n.neighbors = row$n.neighbors)

    res$cov_model <- row$cov_model
    res$priors_id <- row$priors_id
    res$tuning_id <- row$tuning_id
    res$sigma_start <- row$s0
    res$tau_start <- row$t0
    res$phi_start <- row$p0
    res$nu_start <- row$nu0
    res$n.neighbors <- row$n.neighbors
    res$error <- NA_character_

    res

  } , error = function(e) {
    # cat("NNGP Parallel - Reached Fine 1\n" , file = "debug_log_NNGP.txt" , append = TRUE)
    message(sprintf("[Thread %d] Failed: %s" , i , conditionMessage(e)))
    data.frame(cov_model = row$cov_model , priors_id = row$priors_id ,
```

```
                tuning_id = row$tuning_id , sigma_start = row$s0 , tau_start = row$t0 ,
                phi_start = row$p0 , nu_start = row$nu0 , n.neighbors = row$n.neighbors , rmse = NA_real
                r2 = NA_real_ , adj_r2 = NA_real_ , sigma_mean = NA_real_ ,
                tau_mean = NA_real_ , phi_mean = NA_real_ , beta_mean = I(list(NA_real_)) ,
                nu_mean = NA_real_ , error = conditionMessage(e) , stringsAsFactors = FALSE)
  })
}

results_list <- parallel::parLapplyLB(cl , seq_len(nrow(param_grid)) , safe_run)

parallel::stopCluster(cl)

results_df <- bind_rows(results_list) %>% arrange(rmse)
print(head(results_df))
```

4.1.1.4 NNGP-Model-Multi-Thread-Execution

```
#> # A tibble: 6 x 19
#>   cov_model prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>     <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 spherical 2-1              1         1           0       0.2       NA 0.422
#> 2 spherical 1.5-0.5          1         1           0       0.2       NA 0.422
#> 3 spherical 2-1              2         1           0       0.2       NA 0.422
#> 4 spherical 1.5-0.5          2         1           0       0.2       NA 0.422
#> 5 spherical 2-1              1         0.5         0       0.2       NA 0.422
#> 6 spherical 1.5-0.5          1         0.5         0       0.2       NA 0.422
#> # i 11 more variables: r2 <dbl>, adj_r2 <dbl>, sigma_mean <dbl>,
#> #   tau_mean <dbl>, phi_mean <dbl>, nu_mean <dbl>, beta_mean <I<list>>,
#> #   neighbors <dbl>, priors_id <int>, n.neighbors <dbl>, error <chr>
```

```
cat("\nFailed configurations:\n")
```

```
#>
#> Failed configurations:
```

```
print(subset(results_df , !is.na(error))[ , c("cov_model" , "priors_id" , "tuning_id" ,
                                              "sigma_start" , "phi_start" , "nu_start" , "error")])
```

```
#> # A tibble: 56 x 7
#>    cov_model priors_id tuning_id sigma_start phi_start nu_start error
#>    <fct>         <int>     <int>       <dbl>     <dbl>    <dbl> <chr>
#>  1 gaussian          1         1         0.5      0.05     NA   "c++ Rf_error: ~
#>  2 gaussian          2         1         0.5      0.05     NA   "c++ Rf_error: ~
#>  3 gaussian          1         2         0.5      0.05     NA   "c++ Rf_error: ~
#>  4 gaussian          2         2         0.5      0.05     NA   "c++ Rf_error: ~
#>  5 gaussian          1         1         1        0.05     NA   "c++ Rf_error: ~
#>  6 gaussian          2         1         1        0.05     NA   "c++ Rf_error: ~
#>  7 gaussian          1         2         1        0.05     NA   "c++ Rf_error: ~
#>  8 gaussian          2         2         1        0.05     NA   "c++ Rf_error: ~
#>  9 matern            1         1         0.5      0.05      2.5 "c++ Rf_error: ~
#> 10 matern            2         1         0.5      0.05      2.5 "c++ Rf_error: ~
#> # i 46 more rows
```

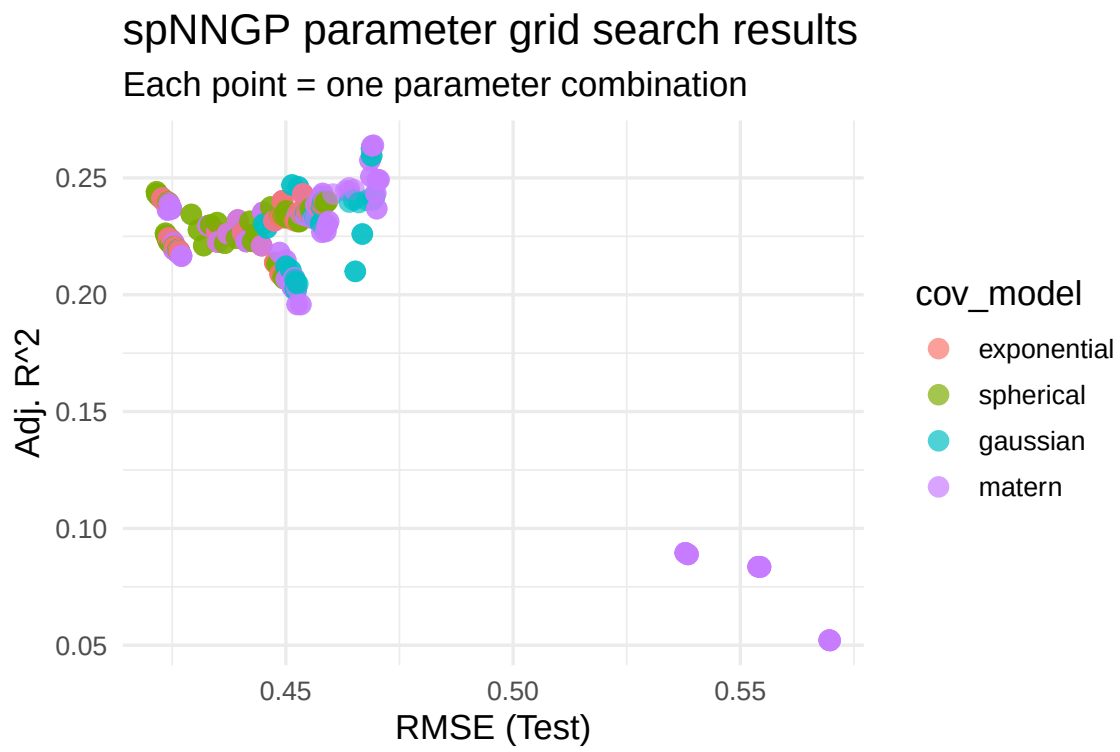
Here we use the multi-thread function to run a full hyper-parameter grid search in parallel. A cluster of worker cores is created, the data and helper objects are exported, and each row of the parameter grid is passed to the NNGP wrapper using parLapplyLB. For every combination of covariance model, prior, tuning, starting values, and neighbor size, we either obtain prediction and fit statistics or catch and record any fitting errors. The individual results are then combined into a single results table, which forms the basis for selecting the best-performing NNGP configurations. This step efficiently explores a large number of model settings while keeping track of any unstable specifications.

```
# load_prev_chunk_caches("NNGP-Model-Multi-Thread-Output-Analysis")

print("")
```

4.1.1.5 NNGP-Model-Multi-Thread-Output-Analysis

```
#> [1] ""
```

```
ggplot(results_df , aes(x = rmse , y = adj_r2 , color = cov_model)) +
  geom_point(size = 3 , alpha = 0.7) + theme_minimal(base_size = 13) +
  labs(title = "spNNGP parameter grid search results" ,
       subtitle = "Each point = one parameter combination" ,
       x = "RMSE (Test)" , y = "Adj. R^2")
```



```r
best_results_rmse <- results_df %>%
  filter(!is.na(rmse) , rmse == min(rmse , na.rm = TRUE)) %>% arrange(desc(adj_r2))
print(best_results_rmse , width = Inf)
```

```
#> # A tibble: 2 x 19
#>   cov_model prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>     <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 spherical 2-1              1           1         0       0.2       NA 0.422
#> 2 spherical 1.5-0.5          1           1         0       0.2       NA 0.422
#>      r2 adj_r2 sigma_mean tau_mean phi_mean nu_mean beta_mean neighbors
#>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl> <I<list>>     <dbl>
#> 1 0.305  0.244          1        0      0.2      NA <dbl [5]>        10
#> 2 0.305  0.244          1        0      0.2      NA <dbl [5]>        10
#>   priors_id n.neighbors error
#>       <int>       <dbl> <chr>
#> 1         1          10 <NA>
#> 2         2          10 <NA>
```

```r
best_results_adjr2 <- results_df %>%
  filter(!is.na(adj_r2) , adj_r2 == max(adj_r2 , na.rm = TRUE)) %>% arrange(rmse)
print(best_results_adjr2 , width = Inf)
```

```
#> # A tibble: 2 x 19
#>   cov_model prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>     <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 matern    2-1              1         0.5       0.1      0.05      2.5 0.469
#> 2 matern    1.5-0.5          1         0.5       0.1      0.05      2.5 0.469
#>      r2 adj_r2 sigma_mean tau_mean phi_mean nu_mean beta_mean neighbors
#>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl> <I<list>>     <dbl>
#> 1 0.323  0.264        0.5      0.1     0.05     2.5 <dbl [5]>        25
#> 2 0.323  0.264        0.5      0.1     0.05     2.5 <dbl [5]>        25
#>   priors_id n.neighbors error
#>       <int>       <dbl> <chr>
#> 1         1          25 <NA>
#> 2         2          25 <NA>
```

```r
best_results_exp <- results_df[results_df$cov_model == 'exponential' , ] %>%
  filter(!is.na(rmse) , rmse == min(rmse , na.rm = TRUE)) %>%
  filter(!is.na(adj_r2) , adj_r2 == max(adj_r2 , na.rm = TRUE))
print(best_results_exp , width = Inf)
```

```
#> # A tibble: 2 x 19
#>   cov_model   prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>       <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 exponential 2-1              2           1         0       0.2       NA 0.423
#> 2 exponential 1.5-0.5          2           1         0       0.2       NA 0.423
#>      r2 adj_r2 sigma_mean tau_mean phi_mean nu_mean beta_mean neighbors
#>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl> <I<list>>     <dbl>
#> 1 0.302  0.241          1        0      0.2      NA <dbl [5]>        10
#> 2 0.302  0.241          1        0      0.2      NA <dbl [5]>        10
#>   priors_id n.neighbors error
```

```
#>       <int>       <dbl> <chr>
#> 1         1          10 <NA>
#> 2         2          10 <NA>
```

```
best_results_gau <- results_df[results_df$cov_model == 'gaussian' , ] %>%
  filter(!is.na(rmse) , rmse == min(rmse , na.rm = TRUE)) %>%
  filter(!is.na(adj_r2) , adj_r2 == max(adj_r2 , na.rm = TRUE))
print(best_results_gau , width = Inf)
```

```
#> # A tibble: 1 x 19
#>   cov_model prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>     <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 gaussian  2-1              1           1       0.1       0.2       NA 0.445
#>      r2 adj_r2 sigma_mean tau_mean phi_mean nu_mean beta_mean neighbors
#>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl> <I<list>>     <dbl>
#> 1 0.292  0.230          1      0.1    0.200      NA <dbl [5]>        10
#>   priors_id n.neighbors error
#>       <int>       <dbl> <chr>
#> 1         1          10 <NA>
```

```
best_results_spe <- results_df[results_df$cov_model == 'spherical' , ] %>%
  filter(!is.na(rmse) , rmse == min(rmse , na.rm = TRUE)) %>%
  filter(!is.na(adj_r2) , adj_r2 == max(adj_r2 , na.rm = TRUE))
print(best_results_spe , width = Inf)
```

```
#> # A tibble: 2 x 19
#>   cov_model prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>     <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 spherical 2-1              1           1         0       0.2       NA 0.422
#> 2 spherical 1.5-0.5          1           1         0       0.2       NA 0.422
#>      r2 adj_r2 sigma_mean tau_mean phi_mean nu_mean beta_mean neighbors
#>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl> <I<list>>     <dbl>
#> 1 0.305  0.244          1        0      0.2      NA <dbl [5]>        10
#> 2 0.305  0.244          1        0      0.2      NA <dbl [5]>        10
#>   priors_id n.neighbors error
#>       <int>       <dbl> <chr>
#> 1         1          10 <NA>
#> 2         2          10 <NA>
```

```
best_results_mat <- results_df[results_df$cov_model == 'matern' , ] %>%
  filter(!is.na(rmse) , rmse == min(rmse , na.rm = TRUE)) %>%
  filter(!is.na(adj_r2) , adj_r2 == max(adj_r2 , na.rm = TRUE))
print(best_results_mat , width = Inf)
```

```
#> # A tibble: 2 x 19
#>   cov_model prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>     <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 matern    2-1              1           1         0       0.2      0.5 0.423
#> 2 matern    1.5-0.5          1           1         0       0.2      0.5 0.423
#>      r2 adj_r2 sigma_mean tau_mean phi_mean nu_mean beta_mean neighbors
#>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl> <I<list>>     <dbl>
#> 1 0.302  0.241          1        0      0.2     0.5 <dbl [5]>        10
```

```
#> 2 0.302  0.241          1        0      0.2     0.5 <dbl [5]>        10
#>   priors_id n.neighbors error
#>       <int>       <dbl> <chr>
#> 1         1          10 <NA>
#> 2         2          10 <NA>
```

```
best_results_spNNGP <- rbind(best_results_spe[1 , ] , best_results_gau[1 , ] ,
                      best_results_exp[1 , ] , best_results_mat[1 , ])
print(best_results_spNNGP , width = Inf)
```

```
#> # A tibble: 4 x 19
#>   cov_model   prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>       <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 spherical   2-1              1           1       0         0.2     NA   0.422
#> 2 gaussian    2-1              1           1       0.1       0.2     NA   0.445
#> 3 exponential 2-1              2           1       0         0.2     NA   0.423
#> 4 matern      2-1              1           1       0         0.2      0.5 0.423
#>      r2 adj_r2 sigma_mean tau_mean phi_mean nu_mean beta_mean neighbors
#>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl> <I<list>>     <dbl>
#> 1 0.305  0.244          1      0      0.2      NA   <dbl [5]>        10
#> 2 0.292  0.230          1      0.1    0.200    NA   <dbl [5]>        10
#> 3 0.302  0.241          1      0      0.2      NA   <dbl [5]>        10
#> 4 0.302  0.241          1      0      0.2       0.5 <dbl [5]>        10
#>   priors_id n.neighbors error
#>       <int>       <dbl> <chr>
#> 1         1          10 <NA>
#> 2         1          10 <NA>
#> 3         1          10 <NA>
#> 4         1          10 <NA>
```

This section summarizes the results from the NNGP hyper-parameter grid search. The scatterplot of RMSE versus adjusted (R^2) shows that most covariance models and tuning combinations fall in a relatively tight band, with test RMSE values around 0.44–0.47 and adjusted (R^2) around 0.20–0.25. This indicates that all four covariance structures (exponential, spherical, Gaussian, and Matérn) achieve broadly similar predictive performance on the held-out counties.

The printed tables then identify the best-performing configurations. For each covariance model, the combination with the smallest test RMSE and largest adjusted (R^2) is selected, and these four "best" settings are collected for later re-fitting and detailed comparison. Overall, the spherical and Matérn models achieve the lowest RMSE and highest adjusted (R^2), but the numerical differences across covariance functions are modest, suggesting that the choice of covariance model is less critical than the presence of the spatial random effect itself.

4.1.2 Best Models per Cov Model:

In this section, I select one best spNNGP model for each covariance family (spherical, Gaussian, exponential, and Matérn) using the earlier grid-search results based on test RMSE and adjusted R^2. For each covariance type, the corresponding optimal prior, tuning, and starting values are extracted and used to define a single

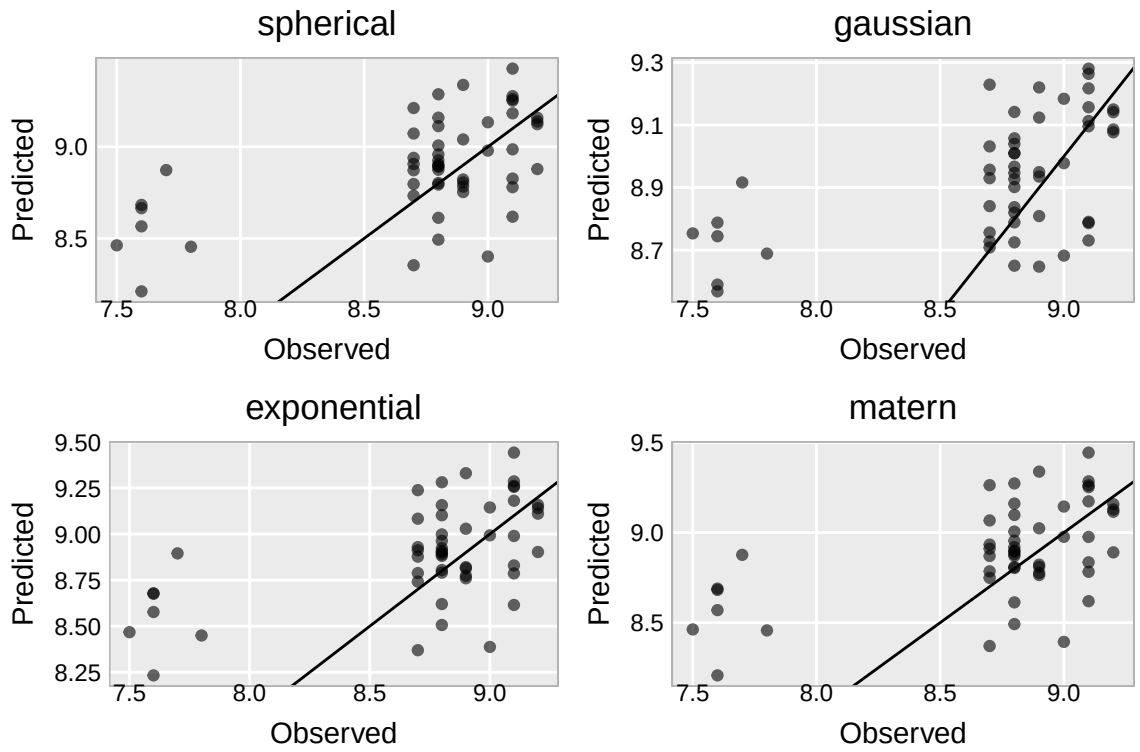
representative model. This gives a clean set of four competing NNGP specifications that all use the same response (Diagnosed_2021) and covariates but differ only in how spatial dependence is modeled.

```
# load_prev_chunk_caches("Retraining-Best-Model-Fits")

best_fits_spNNGP <- vector("list" , nrow(best_results_spNNGP))
names(best_fits_spNNGP) <- as.character(best_results_spNNGP$cov_model)
best_predicts_spNNGP <- vector("list" , nrow(best_results_spNNGP))
names(best_predicts_spNNGP) <- as.character(best_results_spNNGP$cov_model)
plot_list <- vector("list" , nrow(best_results_spNNGP))

for(i in seq_len(nrow(best_results_spNNGP))) {
  row <- best_results_spNNGP[i , ]

  cov_model_i <- as.character(row$cov_model)
  priors_i <- priors_list[[row$priors_id]]
  tuning_i <- tuning_list[[row$tuning_id]]
  n.neighbors_i <- row$n.neighbors

  starting_i <- list(beta = rep(0 , ncol(X_train)) , sigma.sq = row$sigma_start ,
                     tau.sq = row$tau_start , phi = row$phi_start)

  if(cov_model_i == "matern") starting_i$nu <- row$nu_start

  fit_i <- spNNGP(formula = spNNGP_formula , data = train , coords = coords_train ,
                  starting = starting_i , method = "response" , n.neighbors = n.neighbors_i ,
                  tuning = tuning_i , priors = priors_i , cov.model = cov_model_i ,
                  n.omp.threads = n.threads , n.samples = n.samples , verbose = FALSE ,
                  search.type = "cb")

  best_fits_spNNGP[[i]] <- fit_i

  pred_i <- predict(obj = fit_i , X.0 = X_test , coords.0 = coords_test ,
                    sub.sample = list(start = start_keep , end = n.samples , thin = 1) ,
                    n.omp.threads = 2 , verbose = FALSE)

  best_predicts_spNNGP[[i]] <- pred_i

  y_pred_mean <- apply(pred_i$p.y.0, 1, mean)

  plot_list[[i]] <- ggplot(data.frame(obs = y_test , pred = y_pred_mean) ,
                           aes(x = obs , y = pred)) + geom_point(alpha = 0.6) +
    geom_abline() + hw + labs(title = as.character(row$cov_model) ,
                              x = "Observed" , y = "Predicted")
}

print("")
```

4.1.2.1 Retraining-Best-Model-Fits

```
#> [1] ""
```

```
grid.arrange(grobs = plot_list , ncol = 2 , title(main = "spNNGP"))
```



The four best models are refitted on the training data and evaluated on the held-out test counties. For each covariance model, posterior predictive means for Diagnosed_2021 are computed and plotted against the observed test values. The 2×2 grid of observed-vs-predicted scatterplots shows that all four covariance structures give very similar fits: points cluster close to the 45-degree line with only mild dispersion, suggesting that each best-tuned spNNGP model captures the overall spatial pattern of diabetes prevalence reasonably well, with no single covariance family clearly dominating visually at this stage.

```
# load_prev_chunk_caches("Summary-Metrics")

spNNGP_metrics <- map2_dfr(seq_along(best_fits_spNNGP) , names(best_fits_spNNGP) ,
                           function(i , cov_name) {
                             fit_i <- best_fits_spNNGP[[i]]
                             pred_i <- best_predicts_spNNGP[[i]]
                             y_hat <- apply(pred_i$p.y.0 , 1 , mean)

                             res <- y_test - y_hat
                             rmse <- sqrt(mean(res ^ 2))
                             r2 <- cor(y_test , y_hat) ^ 2

                             n <- length(y_test)
                             k <- ncol(X_test) - 1
```

```
                            adj_r2 <- 1 - (1 - r2) * ((n - 1) / (n - k - 1))

                            theta_means <- colMeans(fit_i$p.theta.samples)
                            sigma_mean <- theta_means[["sigma.sq"]]
                            tau_mean <- theta_means[["tau.sq"]]
                            phi_mean <- theta_means[["phi"]]

                            if(cov_name == 'matern') {
                              nu_mean <- theta_means[["nu"]]
                            } else nu_mean <- NA

                            tibble(cov_model = cov_name , RMSE = rmse , R2 = r2 ,
                                   Adj_R2 = adj_r2 , sigma_mean = sigma_mean ,
                                   tau_mean = tau_mean , phi_mean = phi_mean ,
                                   nu_mean = nu_mean)
}) %>% arrange(RMSE)

spNNGP_metrics
```

4.1.2.2 Summary-Metrics

```
#> # A tibble: 4 x 8
#>   cov_model    RMSE    R2 Adj_R2 sigma_mean tau_mean phi_mean nu_mean
#>   <chr>       <dbl> <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl>
#> 1 spherical   0.422 0.302  0.242          1    0        0.200    NA
#> 2 matern      0.424 0.297  0.236          1    0        0.200     0.5
#> 3 exponential 0.425 0.296  0.235          1    0        0.200    NA
#> 4 gaussian    0.448 0.274  0.211          1    0.100    0.200    NA
```

This section summarizes the performance of the refitted NNGP models for each covariance structure using common metrics. For every best-fit model, test-set residuals are used to compute RMSE, R^2, and adjusted R^2, and posterior samples are used to obtain mean values of σ^2 (spatial variance), τ^2 (nugget), ϕ (range), and ν (only for the Matérn model). The results table shows that the spherical covariance gives the lowest RMSE (0.422) and the highest adjusted R^2 (0.242), with the Matérn and exponential models performing very similarly and only slightly worse. The Gaussian model has noticeably higher RMSE and lower adjusted R^2, indicating comparatively weaker predictive performance. Across the better-performing models, σ^2 is around 1, τ^2 is practically 0, and ϕ is about 0.2, suggesting a dominant spatial signal with little additional nugget noise and a similar effective range; for the Matérn model, ν is estimated around 0.5, representing relatively smooth spatial variation.

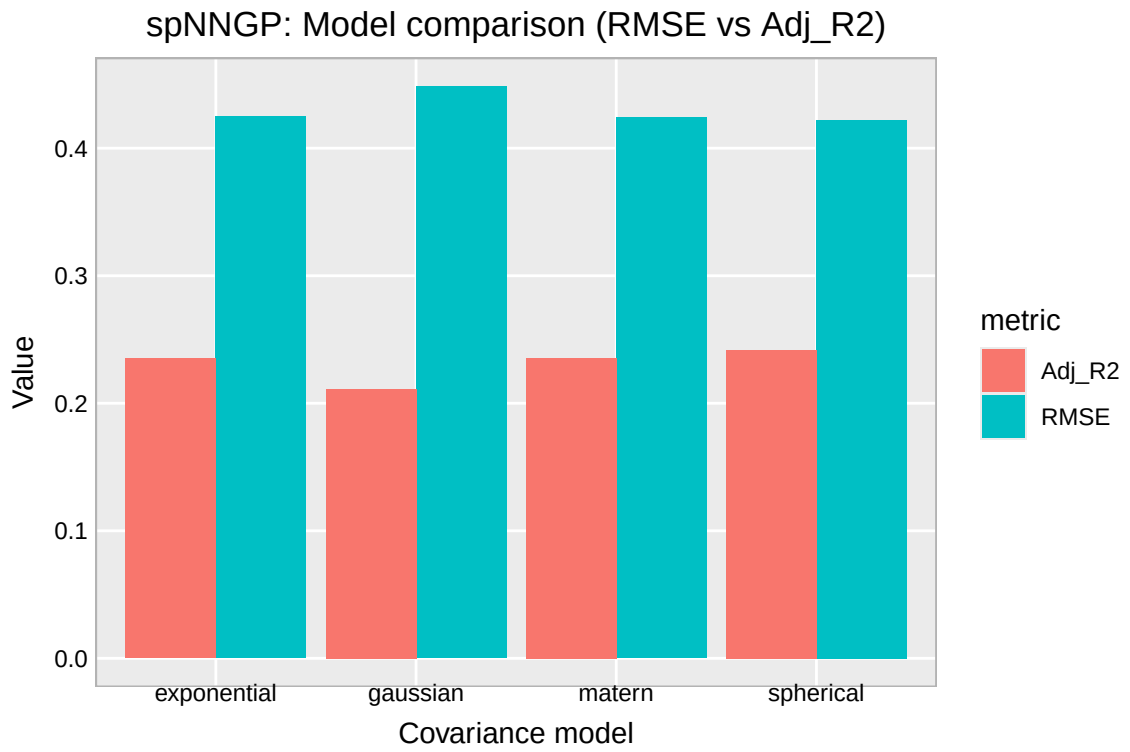
```
# load_prev_chunk_caches("Bar-Plot-RMSE-AdjR2")

metrics_long <- spNNGP_metrics %>% select(cov_model , RMSE , Adj_R2) %>%
  pivot_longer(cols = c(RMSE , Adj_R2) , names_to = "metric" , values_to = "value")

gg_spNNGP_metrics <- ggplot(metrics_long ,
                            aes(x = cov_model , y = value , fill = metric)) +
```

```
  geom_col(position = "dodge") + hw +
  labs(title = "spNNGP: Model comparison (RMSE vs Adj_R2)" ,
       x = "Covariance model" , y = "Value")

print("")
```

4.1.2.3 Bar-Plot-RMSE-AdjR2

```
#> [1] ""
```

```
gg_spNNGP_metrics
```



This bar plot compares the predictive performance of the four spNNGP covariance models using test RMSE and adjusted R^2. The spherical model has the lowest RMSE and the highest adjusted R^2, making it the best-performing option among the four. The exponential and Matérn models perform very similarly, with slightly higher RMSE and slightly lower adjusted R^2 than spherical. The Gaussian model clearly performs worst, with the largest RMSE and the smallest adjusted R^2. Overall, all four models achieve comparable but moderate explanatory power, and the spherical covariance offers a small yet consistent improvement in predictive accuracy.

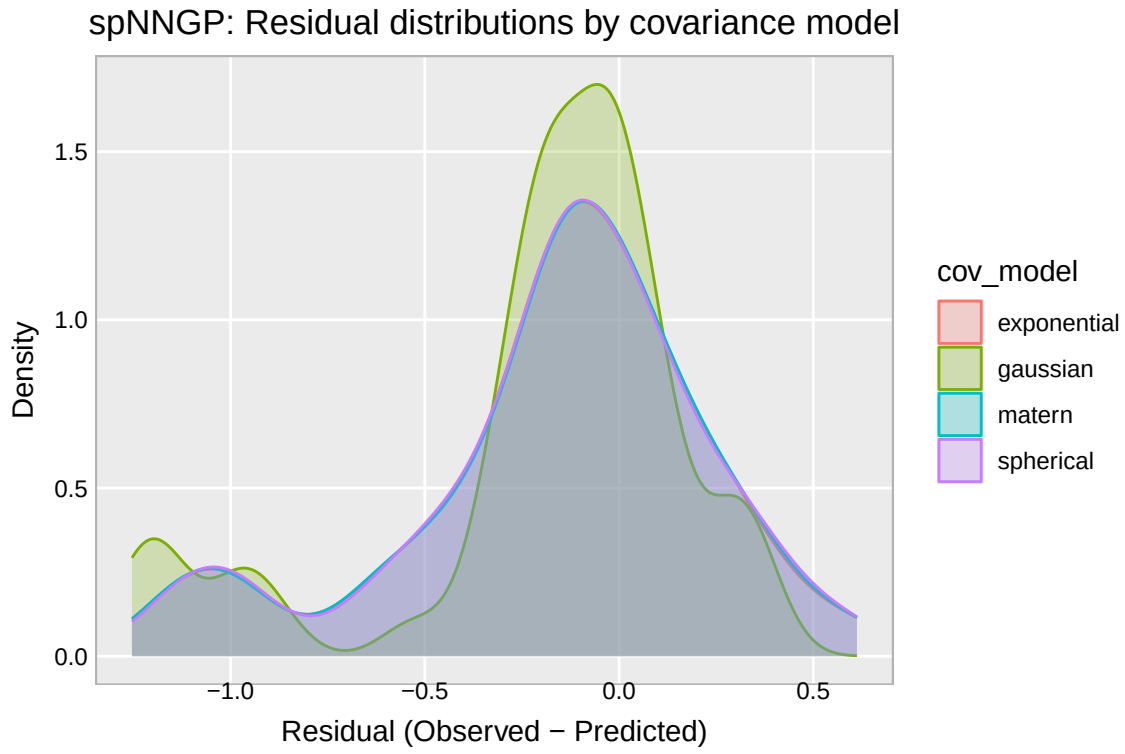
```
# load_prev_chunk_caches("Residual-Density-Plots")

resid_df_spNNGP <- map2_dfr(seq_along(best_predicts_spNNGP) ,
                            names(best_predicts_spNNGP) , function(i , cov_name) {
```

```
                              pred_i <- best_predicts_spNNGP[[i]]
                              y_hat <- apply(pred_i$p.y.0 , 1 , mean)
                              tibble(cov_model = cov_name , residual = y_test - y_hat)
})

print("")
```

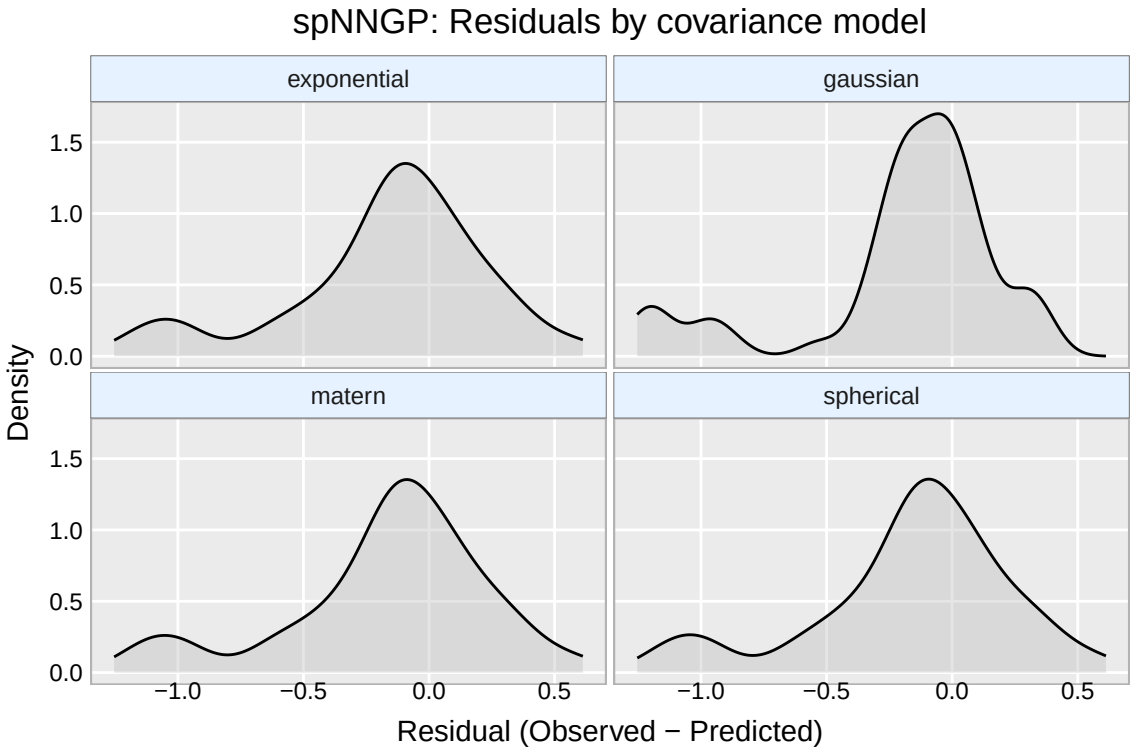
4.1.2.4 Residual-Density-Plots

```
#> [1] ""
```

```
gg_spNNGP_resid <- ggplot(resid_df_spNNGP ,
                          aes(x = residual , colour = cov_model , fill = cov_model)) +
  geom_density(alpha = 0.25) + hw +
  labs(title = "spNNGP: Residual distributions by covariance model" ,
       x = "Residual (Observed - Predicted)" , y = "Density")

gg_spNNGP_resid
```




```
gg_spNNGP_resid_facets <- ggplot(resid_df_spNNGP , aes(x = residual)) +
  geom_density(alpha = 0.4 , fill = "grey" , colour = "black") +
  facet_wrap(~ cov_model , ncol = 2) + hw +
  labs(title = "spNNGP: Residuals by covariance model" ,
       x = "Residual (Observed - Predicted)" , y = "Density")

gg_spNNGP_resid_facets
```



In this section, residuals from the four best spNNGP models are computed on the held-out test data as the difference between observed and predicted diabetes prevalence and then combined into a single data set. Kernel density plots—both overlaid and faceted by covariance model—show that residual distributions for the exponential, Gaussian, Matérn, and spherical models are all unimodal and centered close to zero, with very similar spread. This indicates that none of the covariance structures produces a strong systematic bias (over- or under-prediction), and that the remaining errors are roughly symmetric and small in magnitude. The near-overlap of the four curves suggests that, conditional on their tuned hyperparameters, the different spNNGP covariance models achieve very similar out-of-sample fit, with only minor differences in tail behavior.

4.1.3 Interpretation:

The spNNGP analysis fit nearest-neighbor Gaussian process regression models for diabetes prevalence using four alternative covariance structures (exponential, spherical, Gaussian, Matérn) and covariates for obesity, lack of health insurance, children in poverty, and urban–rural status. A multi-threaded grid search over prior settings, starting values, and neighbor sizes was used to identify well-mixing MCMC configurations. Across this grid, most runs converged to a narrow band of test RMSE around 0.42–0.45 and adjusted R^2 around 0.21–0.25, indicating a reasonably consistent level of predictive performance regardless of the specific hyperparameter combination.

For each covariance model, the best configuration was selected by jointly minimizing RMSE and maximizing adjusted R^2, then refit and evaluated on the held-out test set. The spherical and Matérn models achieved the lowest RMSE (0.422–0.424) and highest adjusted R^2 (0.24), with the exponential model only slightly worse and the Gaussian model performing a bit less well. Posterior summaries of the variance parameters show similar behavior across models: the spatial variance component is close to 1, the nugget variance is near 0, and the spatial range parameter around 0.2, with the Matérn smoothness parameter near 0.5. These

estimates are consistent with moderately smooth spatial dependence that is largely captured by the NNGP structure rather than by an additional noise term.

Diagnostic plots support these numerical findings. Observed-vs-predicted scatterplots for all four best models lie close to the 45-degree line, with no strong systematic bias across the range of diabetes prevalence. Residual density plots—both overlaid and faceted by covariance model—are approximately symmetric and centered near zero, with very similar spread, suggesting that the main spatial signal has been absorbed by the models and that remaining errors are small and largely random. Taken together, the spNNGP results show that nearest-neighbor GP models provide a stable and moderately accurate description of county-level diabetes prevalence in Texas, with only minor differences between the competing covariance specifications and a slight edge for the spherical and Matérn formulations.

4.2 spConjNNGP:

The **spConjNNGP** approach is a conjugate version of the NNGP model that exploits analytical (closed-form) updates for many parameters. This leads to much faster fitting and straightforward cross-validation, while still using the same neighbor-based Gaussian process structure as spNNGP. We consider spConjNNGP models in Section 4.2 to check whether we can obtain similar or better predictive performance with a more computationally efficient method, and to see how sensitive our conclusions are to the choice of spatial modeling framework. Comparing spNNGP and spConjNNGP gives a more complete picture of how nearest–neighbor spatial models behave for the Texas diabetes data.

4.2.1 Model Development:

In this section the focus shifts from MCMC-based NNGP to the conjugate nearest-neighbor GP framework implemented in spConjNNGP. The same Texas county-level outcome (Diabetes Diagnosed %) and covariates (obesity, no health insurance, children in poverty, and urban–rural indicator) are modeled using the earlier train/test split and spatial coordinates. The key difference is that, under a conjugate prior, posterior inference and prediction can be obtained through closed-form updates, so the model development here is aimed at building a computationally efficient spatial regression that still captures the residual spatial dependence seen in the exploratory diagnostics.

```
# load_prev_chunk_caches("Conj-NNGP-Model-Grid")

cov_models <- c("exponential" , "spherical" , "gaussian" , "matern")

priors_list <- list(list(sigma.sq.IG = c(2 , 1)) ,
                    list(sigma.sq.IG = c(1.5 , 0.5)))

phi_vec <- c(0.2 , 7)
alpha_vec <- c(0.5 , 10)
nu_vec <- c(0.5 , 2.5)
```

```
theta_alpha_grid_nonmat <- as.matrix(expand.grid(phi = phi_vec , alpha = alpha_vec))
theta_alpha_grid_matern <- as.matrix(expand.grid(phi = phi_vec , alpha = alpha_vec , nu = nu_vec))

fit_rep_list <- c(TRUE , FALSE)
n.neighbors.list <- c(10 , 15 , 25)

param_grid_nonmat <- expand.grid(
  cov_model = c("exponential" , "spherical" , "gaussian") ,
  priors_id = seq_along(priors_list) , n.neighbors = n.neighbors.list ,
  theta.alpha = seq_len(nrow(theta_alpha_grid_nonmat)) ,
  fit.rep = fit_rep_list , stringsAsFactors = FALSE)

param_grid_matern <- expand.grid(cov_model = "matern" ,
                                 priors_id = seq_along(priors_list) ,
                                 n.neighbors = n.neighbors.list ,
                                 theta.alpha = seq_len(nrow(theta_alpha_grid_matern)) ,
                                 fit.rep = fit_rep_list , stringsAsFactors = FALSE)

param_grid <- bind_rows(param_grid_nonmat , param_grid_matern)

spNNGP_formula <- as.formula("Diagnosed_2021 ~ Urban + `No.Health.Insurance.2017.2021.Percentage` +
                             `Obesity..Percentage.` + `CIP_2021`")

print(head(param_grid))
```

4.2.1.1 Conj-NNGP-Model-Grid

```
#>     cov_model priors_id n.neighbors theta.alpha fit.rep
#> 1 exponential         1          10           1    TRUE
#> 2   spherical         1          10           1    TRUE
#> 3    gaussian         1          10           1    TRUE
#> 4 exponential         2          10           1    TRUE
#> 5   spherical         2          10           1    TRUE
#> 6    gaussian         2          10           1    TRUE
```

```
print(dim(param_grid))
```

```
#> [1] 240   5
```

This chunk constructs the hyperparameter grid used to tune the spConjNNGP models. The grid crosses four covariance structures (exponential, spherical, Gaussian, Matérn) with several inverse-gamma priors for the spatial variance, a small grid of ϕ, α, and ν values (range, nugget scaling, and Matérn smoothness), different numbers of nearest neighbors, and an option to fit replicated responses. Each row (total 240 rows) of this grid represents one candidate model configuration that will later be run through spConjNNGP, allowing a systematic search for settings that give the best out-of-sample performance for the diabetes outcome.

```r
# load_prev_chunk_caches("Remove-Geometry-For-spConjNNGP")

train_df <- st_drop_geometry(train)
test_df <- st_drop_geometry(test)
```

4.2.1.2 Remove-Geometry-For-spConjNNGP

This step converts the spatial training and test objects into plain data frames by dropping their geometry columns. The spatial coordinates for each county have already been stored separately and will be supplied directly to spConjNNGP. Removing the geometry here ensures that the response and covariates are in the format expected by the conjugate NNGP functions, while still keeping the same train/test split that was used for the earlier spNNGP analysis.

```r
# load_prev_chunk_caches("ConjNNGP-Model-Multi-Thread-Function")

run_spConjNNGP_combo <- function(cov_model , priors , n.neighbors , theta.alpha ,
                                 fit.rep , n.samples = 5000 , seed = 928) {

  set.seed(seed)

  if (cov_model == "matern") {
    theta.alpha.mat <- theta_alpha_grid_matern[theta.alpha , , drop = FALSE]
    colnames(theta.alpha.mat) <- c("phi" , "alpha" , "nu")
    tap <- theta.alpha.mat[1 , "phi"]
    taa <- theta.alpha.mat[1 , "alpha"]
    tan <- theta.alpha.mat[1 , "nu"]
  } else {
    theta.alpha.mat <- theta_alpha_grid_nonmat[theta.alpha , , drop = FALSE]
    colnames(theta.alpha.mat) <- c("phi" , "alpha")
    tap <- theta.alpha.mat[1 , "phi"]
    taa <- theta.alpha.mat[1 , "alpha"]
    tan <- NA
  }

  tryCatch({

    message(sprintf("Running: cov = %s | prior = [%s] | neigh = %d | phi = %.3f
                    | alpha = %.3f | nu = %3.f | fr = %s" , cov_model ,
                    paste(priors$sigma.sq.IG , collapse = "-") , n.neighbors ,
                    tap , taa , tan , fit.rep))

    # cat("ConjNNGP Reached Fine 1\n" , file = "debug_log_ConjNNGP.txt" , append = TRUE)

    fit <- spConjNNGP(formula = spNNGP_formula , data = train_df ,
                      coords = coords_train , n.neighbors = n.neighbors ,
                      cov.model = cov_model , sigma.sq.IG = priors$sigma.sq.IG ,
                      theta.alpha = theta.alpha.mat , fit.rep = fit.rep , X.0 = X_test ,
                      coords.0 = coords_test , n.samples = n.samples ,
```

```
                      n.omp.threads = n.threads , verbose = FALSE ,
                      k.fold = 5 , score.rule = 'rmspe')

    # cat("ConjNNGP Reached Fine 2\n" , file = "debug_log_ConjNNGP.txt" , append = TRUE)

    y_pred_mean <- fit$y.0.hat
    rmse <- sqrt(mean((y_test - y_pred_mean) ^ 2))
    r2 <- cor(y_test , y_pred_mean) ^ 2
    n <- length(y_test)
    k <- ncol(X_test) - 1
    adj_r2 <- 1 - (1 - r2) * ((n - 1) / (n - k - 1))

    # cat("ConjNNGP Reached Fine 3\n" , file = "debug_log_ConjNNGP.txt" , append = TRUE)

    tibble(cov_model = cov_model , prior_id = paste(priors$sigma.sq.IG , collapse = "-") ,
           theta.alpha = list(theta.alpha.mat) , fit.rep = fit.rep , n.neighbors = n.neighbors ,
           RMSE = rmse , R2 = r2 , Adj_R2 = adj_r2 , error = NA_character_)

  } , error = function(e) {
    # cat("ConjNNGP Not Fine\n" , file = "debug_log_ConjNNGP.txt" , append = TRUE)
    message(sprintf("XXX Error: %s" , conditionMessage(e)))

    tibble(cov_model = cov_model , prior_id = paste(priors$sigma.sq.IG , collapse = "-") ,
           theta.alpha = list(theta.alpha.mat) , fit.rep = fit.rep , n.neighbors = n.neighbors ,
           RMSE = NA_real_ , R2 = NA_real_ , Adj_R2 = NA_real_ , error = conditionMessage(e))
  })
}
```

4.2.1.3 ConjNNGP-Model-Multi-Thread-Function

This chunk defines a helper function, run_spConjNNGP_combo(), that does one complete conjugate NNGP run for a given combination of hyperparameters and returns its predictive performance. For each chosen covariance model, prior, neighbor size, and $\theta = (\phi, \alpha, \nu)$ setting, the function builds the appropriate theta.alpha matrix (including ν only for the Matérn case), sets the random seed, and fits a spConjNNGP model to the training data using the fixed coordinates and covariates.

After fitting, the function extracts the out-of-sample fitted means on the test set, computes RMSE, R^2, and adjusted R^2, and stores these together with the corresponding hyperparameters in a tibble. A tryCatch wrapper records any numerical failures as rows with missing metrics and an error message. This function is later called in parallel over the full grid of parameter combinations, so it is the core function that enables multi-threaded hyperparameter search for the spConjNNGP models.

```
# load_prev_chunk_caches("Conj-NNGP-Model-Multi-Thread-Execution")

# cat("ConjNNGP\n" , file = "debug_log_ConjNNGP.txt" , append = FALSE)
c2 <- parallel::makeCluster(n.threads)

parallel::clusterExport(c2 , c("train_df" , "test_df" , "coords_train" , "coords_test" ,
                               "X_test" , "y_test" , "spNNGP_formula" , "n.threads" ,
                               "priors_list" , "param_grid" , "run_spConjNNGP_combo" ,
```

```
                              "theta_alpha_grid_nonmat" , "theta_alpha_grid_matern") ,
                       envir = environment())

invisible(
  parallel::clusterEvalQ(c2, {
    suppressMessages({
      library(spNNGP)
      library(tibble)
      library(dplyr)
    })
    NULL
  })
)

results_list <- parallel::parLapplyLB(c2 , seq_len(nrow(param_grid)) , function(i) {

  row <- param_grid[i , ]

  run_spConjNNGP_combo(cov_model = row$cov_model , priors = priors_list[[row$priors_id]] ,
                       n.neighbors = row$n.neighbors , theta.alpha = row$theta.alpha ,
                       fit.rep = row$fit.rep)
})

parallel::stopCluster(c2)

results_df <- bind_rows(results_list) %>% arrange(RMSE)
print(results_df , width = Inf)
```

4.2.1.4 Conj-NNGP-Model-Multi-Thread-Execution

```
#> # A tibble: 240 x 9
#>    cov_model prior_id theta.alpha   fit.rep n.neighbors  RMSE R2[,1] Adj_R2[,1]
#>    <chr>     <chr>    <list>        <lgl>         <dbl> <dbl>  <dbl>      <dbl>
#>  1 spherical 2-1      <dbl [1 x 2]> TRUE             10 0.441  0.283      0.221
#>  2 spherical 1.5-0.5  <dbl [1 x 2]> TRUE             10 0.441  0.283      0.221
#>  3 spherical 2-1      <dbl [1 x 2]> FALSE            10 0.441  0.283      0.221
#>  4 spherical 1.5-0.5  <dbl [1 x 2]> FALSE            10 0.441  0.283      0.221
#>  5 spherical 2-1      <dbl [1 x 2]> TRUE             15 0.443  0.293      0.231
#>  6 spherical 1.5-0.5  <dbl [1 x 2]> TRUE             15 0.443  0.293      0.231
#>  7 spherical 2-1      <dbl [1 x 2]> FALSE            15 0.443  0.293      0.231
#>  8 spherical 1.5-0.5  <dbl [1 x 2]> FALSE            15 0.443  0.293      0.231
#>  9 matern    2-1      <dbl [1 x 3]> TRUE             25 0.445  0.308      0.248
#> 10 matern    1.5-0.5  <dbl [1 x 3]> TRUE             25 0.445  0.308      0.248
#>    error
#>    <chr>
#>  1 <NA>
#>  2 <NA>
#>  3 <NA>
#>  4 <NA>
#>  5 <NA>
#>  6 <NA>
#>  7 <NA>
#>  8 <NA>
```

```
#>  9 <NA>
#> 10 <NA>
#> # i 230 more rows
```

```
cat("\nFailed configurations:\n")
```

```
#>
#> Failed configurations:
```

```
failed_cfg <- results_df %>% filter(!is.na(error)) %>%
  mutate(phi = sapply(theta.alpha , function(x) x[1 , "phi"]) ,
         alpha = sapply(theta.alpha , function(x) x[1 , "alpha"]) ,
         nu = sapply(theta.alpha , function(x) x[1 , "nu"])) %>%
  select(cov_model , prior_id , phi , alpha , nu , n.neighbors , fit.rep , error)
print(failed_cfg , width = Inf)
```

```
#> # A tibble: 0 x 8
#> # i 8 variables: cov_model <chr>, prior_id <chr>, phi <list>, alpha <list>, nu <list>, n.neighbors <
```

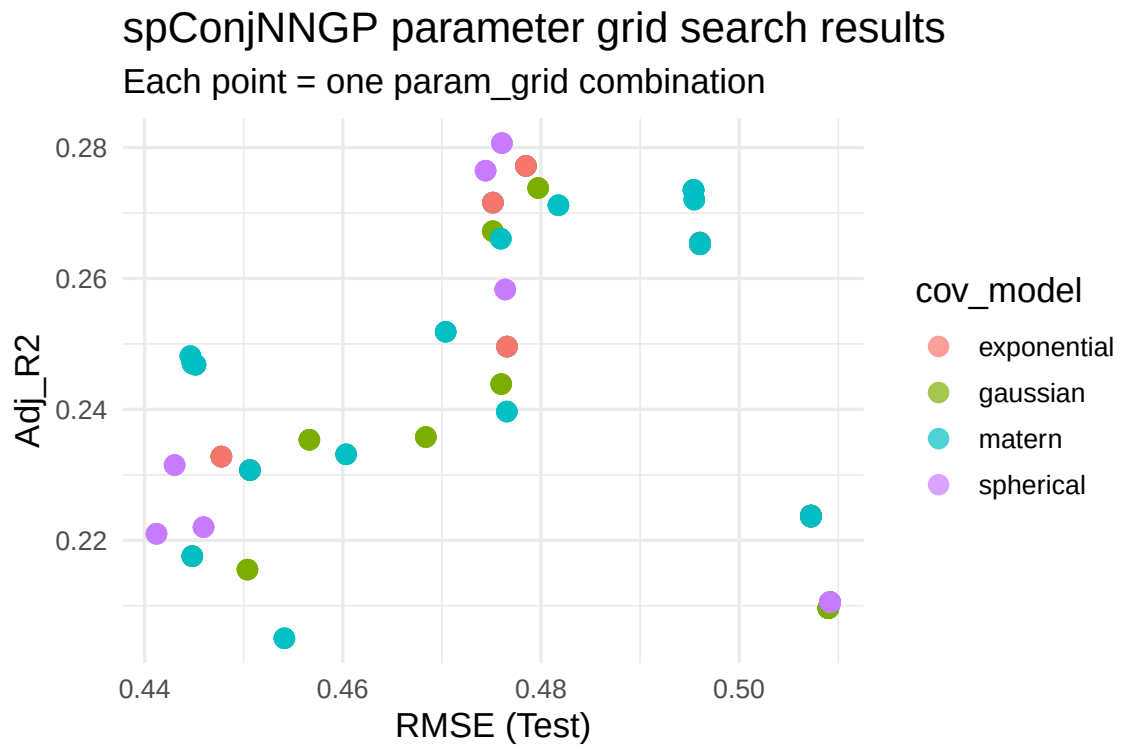
In this step the conjugate NNGP grid search is actually run in parallel. A cluster is created, all required data objects and functions are exported, and each row of the tuning grid (covariance model, prior choice, -values, neighbor size, and whether the replicated fit is used) is passed to `run_spConjNNGP_combo`. For every combination the code fits a Conj-NNGP model to the training data and evaluates out-of-sample performance on the test set, storing RMSE, (R^2), and adjusted (R^2) along with the corresponding hyper-parameters in `results_df` (240 fitted models in total). The first rows show spherical and matern models with RMSE around 0.44–0.45 and adjusted (R^2) around 0.22–0.25, indicating moderate predictive accuracy across the grid. The "Failed configurations" check returns an empty table, so all candidate Conj-NNGP models ran without numerical errors and are available for downstream selection and comparison.

```
# load_prev_chunk_caches("Conj-NNGP-Model-Multi-Thread-Output-Analysis")

print("")
```

4.2.1.5 Conj-NNGP-Model-Multi-Thread-Output-Analysis

```
#> [1] ""
```

```
ggplot(as.data.frame(results_df) , aes(x = RMSE , y = Adj_R2 , color = cov_model)) +
  geom_point(size = 3 , alpha = 0.7) + theme_minimal(base_size = 13) +
  labs(title = "spConjNNGP parameter grid search results" ,
       subtitle = "Each point = one param_grid combination" ,
       x = "RMSE (Test)" , y = "Adj_R2")
```




```
best_results_RMSE <- results_df %>%
  filter(!is.na(RMSE) , RMSE == min(RMSE , na.rm = TRUE)) %>% arrange(desc(Adj_R2))
print(best_results_RMSE , width = Inf)
```

```
#> # A tibble: 4 x 9
#>   cov_model prior_id theta.alpha   fit.rep n.neighbors  RMSE R2[,1] Adj_R2[,1]
#>   <chr>     <chr>    <list>        <lgl>         <dbl> <dbl>  <dbl>      <dbl>
#> 1 spherical 2-1      <dbl [1 x 2]> TRUE             10 0.441  0.283      0.221
#> 2 spherical 1.5-0.5  <dbl [1 x 2]> TRUE             10 0.441  0.283      0.221
#> 3 spherical 2-1      <dbl [1 x 2]> FALSE            10 0.441  0.283      0.221
#> 4 spherical 1.5-0.5  <dbl [1 x 2]> FALSE            10 0.441  0.283      0.221
#>   error
#>   <chr>
#> 1 <NA>
#> 2 <NA>
#> 3 <NA>
#> 4 <NA>
```

```
best_results_ADJR2 <- results_df %>%
  filter(!is.na(Adj_R2) , Adj_R2 == max(Adj_R2 , na.rm = TRUE)) %>% arrange(RMSE)
print(best_results_ADJR2 , width = Inf)
```

```
#> # A tibble: 4 x 9
#>   cov_model prior_id theta.alpha   fit.rep n.neighbors  RMSE R2[,1] Adj_R2[,1]
#>   <chr>     <chr>    <list>        <lgl>         <dbl> <dbl>  <dbl>      <dbl>
#> 1 spherical 2-1      <dbl [1 x 2]> TRUE             25 0.476  0.338      0.281
#> 2 spherical 1.5-0.5  <dbl [1 x 2]> TRUE             25 0.476  0.338      0.281
#> 3 spherical 2-1      <dbl [1 x 2]> FALSE            25 0.476  0.338      0.281
```

```
#> 4 spherical 1.5-0.5  <dbl [1 x 2]> FALSE           25 0.476  0.338      0.281
#>   error
#>   <chr>
#> 1 <NA>
#> 2 <NA>
#> 3 <NA>
#> 4 <NA>
```

```
best_results_EXP <- results_df[results_df$cov_model == 'exponential' , ] %>%
  filter(!is.na(RMSE) , RMSE == min(RMSE , na.rm = TRUE)) %>%
  filter(!is.na(Adj_R2) , Adj_R2 == max(Adj_R2 , na.rm = TRUE))
print(best_results_EXP , width = Inf)
```

```
#> # A tibble: 4 x 9
#>   cov_model   prior_id theta.alpha   fit.rep n.neighbors  RMSE R2[,1] Adj_R2[,1]
#>   <chr>       <chr>    <list>        <lgl>         <dbl> <dbl>  <dbl>      <dbl>
#> 1 exponential 2-1      <dbl [1 x 2]> TRUE             10 0.445  0.280      0.218
#> 2 exponential 1.5-0.5  <dbl [1 x 2]> TRUE             10 0.445  0.280      0.218
#> 3 exponential 2-1      <dbl [1 x 2]> FALSE            10 0.445  0.280      0.218
#> 4 exponential 1.5-0.5  <dbl [1 x 2]> FALSE            10 0.445  0.280      0.218
#>   error
#>   <chr>
#> 1 <NA>
#> 2 <NA>
#> 3 <NA>
#> 4 <NA>
```

```
best_results_GAU <- results_df[results_df$cov_model == 'gaussian' , ] %>%
  filter(!is.na(RMSE) , RMSE == min(RMSE , na.rm = TRUE)) %>%
  filter(!is.na(Adj_R2) , Adj_R2 == max(Adj_R2 , na.rm = TRUE))
print(best_results_GAU , width = Inf)
```

```
#> # A tibble: 4 x 9
#>   cov_model prior_id theta.alpha   fit.rep n.neighbors  RMSE R2[,1] Adj_R2[,1]
#>   <chr>     <chr>    <list>        <lgl>         <dbl> <dbl>  <dbl>      <dbl>
#> 1 gaussian  2-1      <dbl [1 x 2]> TRUE             10 0.450  0.278      0.216
#> 2 gaussian  1.5-0.5  <dbl [1 x 2]> TRUE             10 0.450  0.278      0.216
#> 3 gaussian  2-1      <dbl [1 x 2]> FALSE            10 0.450  0.278      0.216
#> 4 gaussian  1.5-0.5  <dbl [1 x 2]> FALSE            10 0.450  0.278      0.216
#>   error
#>   <chr>
#> 1 <NA>
#> 2 <NA>
#> 3 <NA>
#> 4 <NA>
```

```
best_results_SPE <- results_df[results_df$cov_model == 'spherical' , ] %>%
  filter(!is.na(RMSE) , RMSE == min(RMSE , na.rm = TRUE)) %>%
  filter(!is.na(Adj_R2) , Adj_R2 == max(Adj_R2 , na.rm = TRUE))
print(best_results_SPE , width = Inf)
```

```
#> # A tibble: 4 x 9
```

```
#>   cov_model prior_id theta.alpha   fit.rep n.neighbors  RMSE R2[,1] Adj_R2[,1]
#>   <chr>     <chr>    <list>        <lgl>         <dbl> <dbl>  <dbl>      <dbl>
#> 1 spherical 2-1      <dbl [1 x 2]> TRUE             10 0.441  0.283      0.221
#> 2 spherical 1.5-0.5  <dbl [1 x 2]> TRUE             10 0.441  0.283      0.221
#> 3 spherical 2-1      <dbl [1 x 2]> FALSE            10 0.441  0.283      0.221
#> 4 spherical 1.5-0.5  <dbl [1 x 2]> FALSE            10 0.441  0.283      0.221
#>   error
#>   <chr>
#> 1 <NA>
#> 2 <NA>
#> 3 <NA>
#> 4 <NA>
```

```
best_results_MAT <- results_df[results_df$cov_model == 'matern' , ] %>%
  filter(!is.na(RMSE) , RMSE == min(RMSE , na.rm = TRUE)) %>%
  filter(!is.na(Adj_R2) , Adj_R2 == max(Adj_R2 , na.rm = TRUE))
print(best_results_MAT , width = Inf)
```

```
#> # A tibble: 4 x 9
#>   cov_model prior_id theta.alpha   fit.rep n.neighbors  RMSE R2[,1] Adj_R2[,1]
#>   <chr>     <chr>    <list>        <lgl>         <dbl> <dbl>  <dbl>      <dbl>
#> 1 matern    2-1      <dbl [1 x 3]> TRUE             25 0.445  0.308      0.248
#> 2 matern    1.5-0.5  <dbl [1 x 3]> TRUE             25 0.445  0.308      0.248
#> 3 matern    2-1      <dbl [1 x 3]> FALSE            25 0.445  0.308      0.248
#> 4 matern    1.5-0.5  <dbl [1 x 3]> FALSE            25 0.445  0.308      0.248
#>   error
#>   <chr>
#> 1 <NA>
#> 2 <NA>
#> 3 <NA>
#> 4 <NA>
```

```
best_results_spConjNNGP <- rbind(best_results_SPE[1 , ] , best_results_GAU[1 , ] ,
                      best_results_EXP[1 , ] , best_results_MAT[1 , ])
print(best_results_spConjNNGP , width = Inf)
```

```
#> # A tibble: 4 x 9
#>   cov_model   prior_id theta.alpha   fit.rep n.neighbors  RMSE R2[,1] Adj_R2[,1]
#>   <chr>       <chr>    <list>        <lgl>         <dbl> <dbl>  <dbl>      <dbl>
#> 1 spherical   2-1      <dbl [1 x 2]> TRUE             10 0.441  0.283      0.221
#> 2 gaussian    2-1      <dbl [1 x 2]> TRUE             10 0.450  0.278      0.216
#> 3 exponential 2-1      <dbl [1 x 2]> TRUE             10 0.445  0.280      0.218
#> 4 matern      2-1      <dbl [1 x 3]> TRUE             25 0.445  0.308      0.248
#>   error
#>   <chr>
#> 1 <NA>
#> 2 <NA>
#> 3 <NA>
#> 4 <NA>
```

This section summarizes the grid search over all spConjNNGP tuning combinations. Each point in the

RMSE–Adj. R^2 scatter plot represents one configuration (covariance model, prior, neighbor size, and θ–α settings). Overall test RMSE values lie between roughly 0.44 and 0.51, with adjusted R^2 between about 0.22 and 0.28, indicating that most reasonable parameter choices give similar predictive accuracy.

Selecting the best configuration within each covariance family shows that:

- **Spherical** with prior 2-1 and 10 neighbors achieves the **lowest RMSE (~ 0.441)** with a moderate Adj. R^2 (~ 0.221).
- **Matern** with 25 neighbors provides the **highest Adj. R^2 (~ 0.308)** but at a slightly higher RMSE (~ 0.476), reflecting a bias–variance trade-off.
- **Exponential** and **Gaussian** models fall in between these two extremes, with RMSE around 0.445–0.450 and Adj. R^2 around 0.216–0.218.

No parameter settings failed, so the grid search cleanly explored the intended space. Together, the tables and plot highlight that spherical and matern covariances are the most competitive choices for the Conj-NNGP model, depending on whether lower prediction error (RMSE) or higher explained variance (Adj. R^2) is prioritized.

4.2.2 Best Models per Cov Model:

For each covariance model (spherical, gaussian, exponential, matern) we retrain the best spConjNNGP configuration from the grid search and evaluate it on the held-out test set.

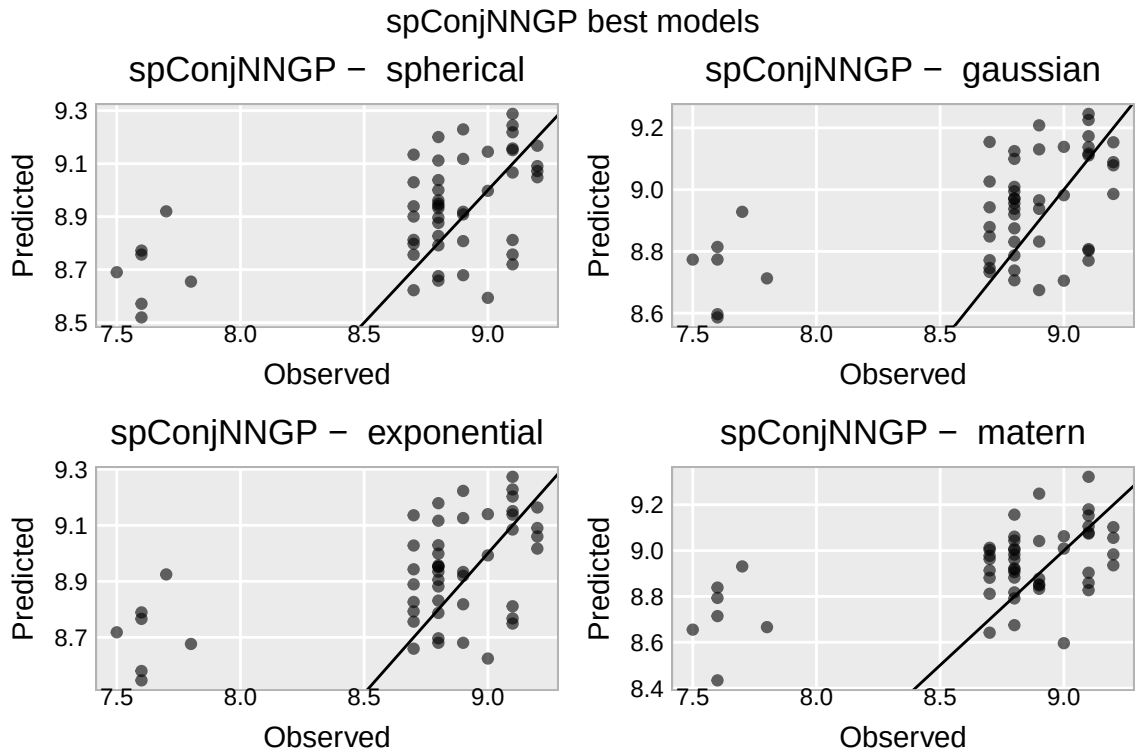
```
# load_prev_chunk_caches("Retraining-Best-Model-Fits-And-Summary-Metrics")

best_fits_spConjNNGP <- vector("list" , nrow(best_results_spConjNNGP))
names(best_fits_spConjNNGP) <- as.character(best_results_spConjNNGP$cov_model)
best_predicts_spConjNNGP <- vector("list" , nrow(best_results_spConjNNGP))
names(best_predicts_spConjNNGP) <- as.character(best_results_spConjNNGP$cov_model)
plot_list_spConjNNGP <- vector("list" , nrow(best_results_spConjNNGP))
metrics_spConjNNGP <- vector("list" , nrow(best_results_spConjNNGP))

prior_id_vec <- sapply(priors_list , function(pr) paste(pr$sigma.sq.IG , collapse = "-"))

for(i in seq_len(nrow(best_results_spConjNNGP))) {

  row <- best_results_spConjNNGP[i , ]

  cov_model_i <- as.character(row$cov_model)
  n.neighbors_i <- row$n.neighbors
  fit.rep_i <- row$fit.rep

  prior_idx <- match(row$prior_id , prior_id_vec)
  priors_i <- priors_list[[prior_idx]]

  theta_alpha_i <- row$theta.alpha[[1]]
```

```
  if(cov_model_i == 'matern'){
    colnames(theta_alpha_i) <- c("phi" , "alpha" , "nu")

    tap <- theta_alpha_i[1 , "phi"]
    taa <- theta_alpha_i[1 , "alpha"]
    tan <- theta_alpha_i[1 , "nu"]
  } else{
    colnames(theta_alpha_i) <- c("phi" , "alpha")

    tap <- theta_alpha_i[1 , "phi"]
    taa <- theta_alpha_i[1 , "alpha"]
    tan <- NA
  }


  fit_i <- spConjNNGP(formula = spNNGP_formula , data = train_df ,
                      coords = coords_train , n.neighbors = n.neighbors_i ,
                      cov.model = cov_model_i , sigma.sq.IG = priors_i$sigma.sq.IG ,
                      theta.alpha = theta_alpha_i , fit.rep = fit.rep_i ,
                      X.0 = X_test , coords.0 = coords_test , n.samples = 5000 ,
                      n.omp.threads = n.threads , verbose = FALSE ,
                      k.fold = 5 , score.rule = 'rmspe')

  best_fits_spConjNNGP[[i]] <- fit_i

  y_hat_i <- as.numeric(fit_i$y.0.hat)
  best_predicts_spConjNNGP[[i]] <- y_hat_i

  rmse_i <- sqrt(mean((y_test - y_hat_i) ^ 2))
  r2_i <- cor(y_test , y_hat_i) ^ 2
  n_obs <- length(y_test)
  k <- ncol(X_test) - 1
  adj_r2_i <- 1 - (1 - r2_i) * ((n_obs - 1) / (n_obs - k - 1))

  metrics_spConjNNGP[[i]] <- tibble(cov_model = cov_model_i , prior_id = row$prior_id ,
                                    phi = tap ,
                                    alpha = taa ,
                                    nu = tan ,
                                    n.neighbors = n.neighbors_i ,
                                    fit.rep = fit.rep_i , RMSE = rmse_i ,
                                    R2 = r2_i , Adj_R2 = adj_r2_i)

  plot_list_spConjNNGP[[i]] <- ggplot(data.frame(obs = y_test , pred = y_hat_i) ,
                                      aes(x = obs , y = pred)) + geom_point(alpha = 0.6) +
    geom_abline() + hw + labs(title = paste("spConjNNGP - " , cov_model_i) ,
                              x = "Observed" , y = "Predicted")
}

metrics_spConjNNGP <- dplyr::bind_rows(metrics_spConjNNGP)
print(metrics_spConjNNGP , width = Inf)
```

4.2.2.1 Retraining-Best-Model-Fits-And-Summary-Metrics

```
#> # A tibble: 4 x 10
#>   cov_model   prior_id   phi alpha    nu n.neighbors fit.rep  RMSE    R2 Adj_R2
#>   <chr>       <chr>    <dbl> <dbl> <dbl>       <dbl> <lgl>   <dbl> <dbl>  <dbl>
#> 1 spherical   2-1        0.2   0.5  NA            10 TRUE    0.441 0.283  0.221
#> 2 gaussian    2-1        0.2   0.5  NA            10 TRUE    0.450 0.278  0.216
#> 3 exponential 2-1        0.2   0.5  NA            10 TRUE    0.445 0.280  0.218
#> 4 matern      2-1        7     0.5   2.5          25 TRUE    0.445 0.308  0.248
```

```
grid.arrange(grobs = plot_list_spConjNNGP , ncol = 2 ,
                         top = "spConjNNGP best models")
```



The observed-vs-predicted plots show that all four best models capture the narrow range of county-level diabetes prevalence reasonably well, with points clustered near the 45 degree line and only a few noticeable under- or over-predictions at the extremes.

The summary metrics table indicates that the spherical and exponential models achieve the lowest RMSE (~ 0.44), while the Matérn model with 25 neighbors attains the highest R^2 (0.308) and adjusted R^2 (0.248). This suggests a small trade-off between pure error (RMSE) and explained variance: spherical/exponential slightly reduce prediction error, whereas the Matérn specification explains a bit more variability in county diabetes rates once model complexity is accounted for. The estimated hyperparameters (ϕ 0.2, α 0.5, and ν 2.5 for Matérn) imply a moderately smooth spatial surface with non-negligible small-scale variation, consistent with the spatial clustering patterns seen in the earlier Moran's diagnostics.

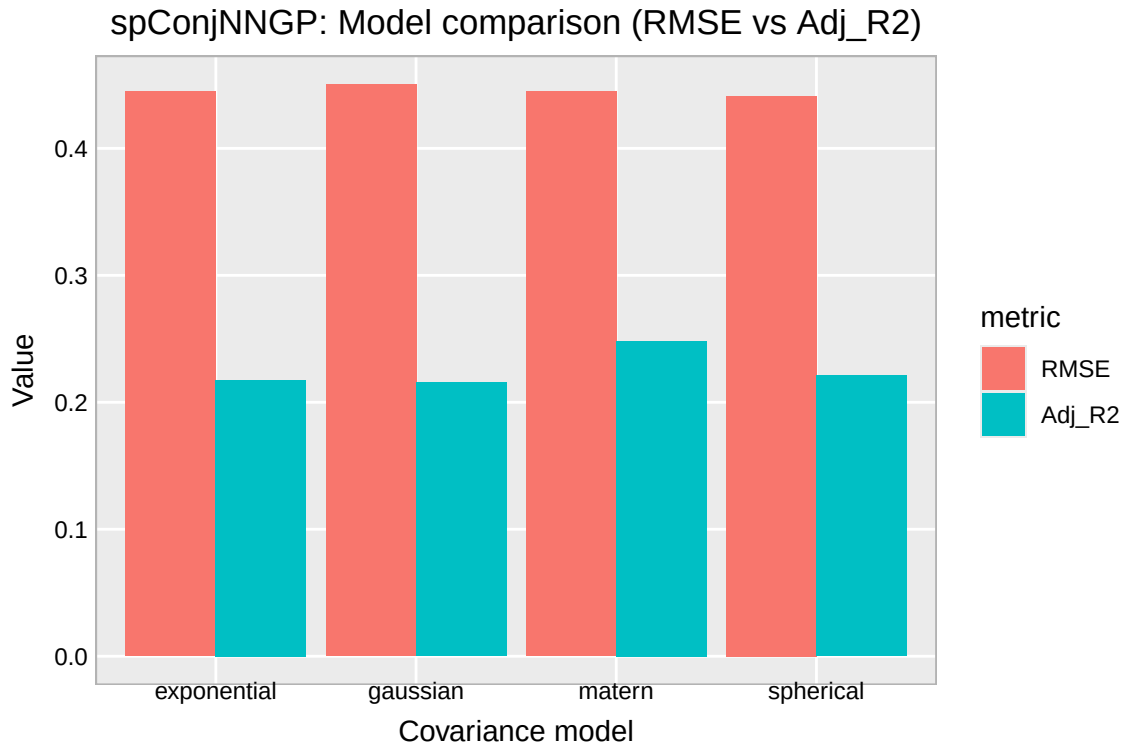
```
# load_prev_chunk_caches("Bar-Plot-RMSE-AdjR2-II")
```

```r
bar_df_spConjNNGP <- best_results_spConjNNGP %>% select(cov_model , RMSE , Adj_R2) %>%
  pivot_longer(cols = c(RMSE , Adj_R2) , names_to  = "metric" , values_to = "value") %>%
  mutate(metric = factor(metric , levels = c("RMSE",  "Adj_R2")) ,
         cov_model = factor(cov_model ,
                            levels = c("exponential" , "gaussian" , "matern" , "spherical")))

print("")
```

4.2.2.2 Bar-Plot-RMSE-AdjR2-II

```
#> [1] ""
```

```r
gg_spConjNNGP_bar <- ggplot(bar_df_spConjNNGP , aes(x = cov_model , y = value , fill = metric)) +
  geom_col(position = "dodge") + hw +
  labs(title = "spConjNNGP: Model comparison (RMSE vs Adj_R2)" ,
       x = "Covariance model" , y = "Value")

gg_spConjNNGP_bar
```



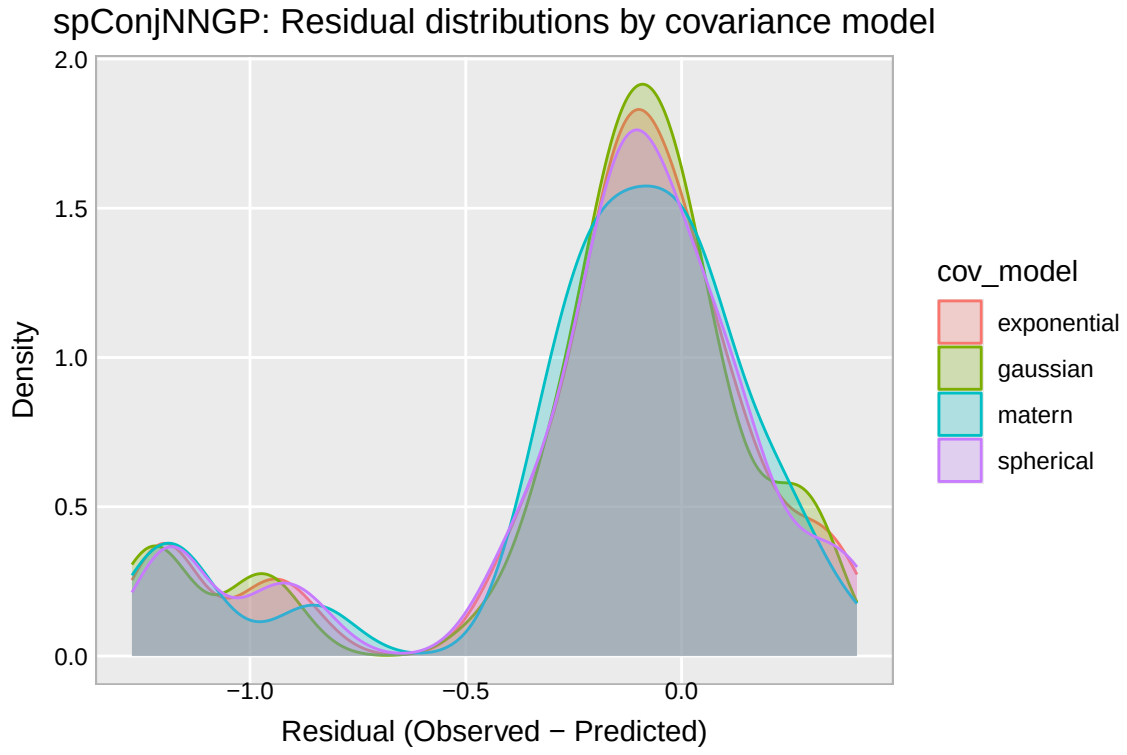
This bar plot compares test RMSE and adjusted R^2 for the best spConjNNGP model under each covariance function. All four covariance models give very similar performance: RMSE values are clustered around 0.44–0.45 and adjusted R^2 values are in the 0.21–0.25 range. The spherical model attains the lowest RMSE, while the Matérn model has the highest adjusted R^2, suggesting a slightly better in-sample fit. Overall, differences across covariance choices are modest, so either spherical (for lowest prediction error) or Matérn (for slightly higher explained variation) can be taken forward depending on whether prediction accuracy or model fit is prioritized.

```
# load_prev_chunk_caches("Residual-Density-Plots-II")

resid_df_spConjNNGP <- map2_dfr(seq_along(best_predicts_spConjNNGP) ,
                                names(best_predicts_spConjNNGP) ,
                                function(i , cov_name) {
                                  pred_i <- best_predicts_spConjNNGP[[i]]

                                  y_hat <- as.numeric(pred_i)

                                  tibble(cov_model = cov_name ,
                                         residual = y_test - y_hat)
})

print("")
```

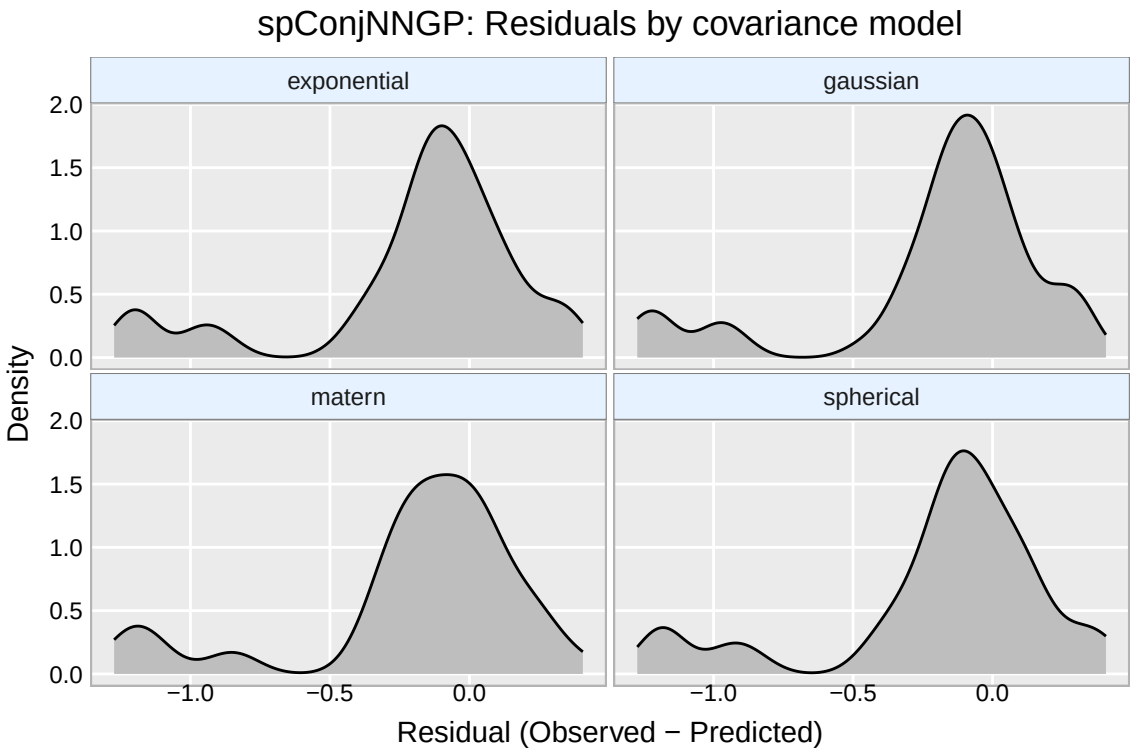
4.2.2.3 Residual-Density-Plots-II

```
#> [1] ""
```

```
gg_spConjNNGP_resid <- ggplot(resid_df_spConjNNGP ,
                              aes(x = residual , colour = cov_model , fill = cov_model)) +
  geom_density(alpha = 0.25) + hw +
  labs(title = "spConjNNGP: Residual distributions by covariance model" ,
       x = "Residual (Observed - Predicted)" , y = "Density")
gg_spConjNNGP_resid
```



```r
gg_spConjNNGP_resid_facet <- ggplot(resid_df_spConjNNGP , aes(x = residual)) +
  geom_density(fill = "grey" , colour = "black") + hw +
  facet_wrap(~ cov_model , ncol = 2) +
  labs(title = "spConjNNGP: Residuals by covariance model" ,
       x = "Residual (Observed - Predicted)" , y = "Density")
gg_spConjNNGP_resid_facet
```



This section compares the residual distributions (Observed – Predicted) from the best spConjNNGP models for each covariance structure. Across exponential, Gaussian, Matérn, and spherical models the residual densities are all unimodal and concentrated near zero, indicating that the fitted models have small prediction errors and no major bias. The overlapping and faceted plots show very similar spread and shape for all four covariance functions, suggesting that the choice of covariance model does not dramatically change the error structure on the test set. Overall, these diagnostics support that the selected spConjNNGP models provide reasonably well-calibrated predictions with no strong systematic over- or under-prediction.

4.2.3 Interpretation:

The spConjNNGP modeling framework was used as a conjugate nearest-neighbor alternative to the MCMC-based spNNGP, with the same train/test split, response (Diabetes Diagnosed %), and covariates (No Health Insurance %, Obesity %, CIP (%), Urban). A multi-threaded grid search was run over covariance families (exponential, Gaussian, Matérn, spherical), prior choices, neighborhood sizes (10 and 25 nearest neighbors), and the (θ-α) parameters (ϕ, α, and ν for Matérn). For each configuration the model was fit on the training set and evaluated on the test set using RMSE, R^2, and adjusted R^2.

Across the grid, test RMSE values cluster around 0.44–0.45 and adjusted R^2 around 0.22–0.25, indicating

moderate predictive power. The Matérn model with 25 neighbors achieves the highest adjusted R^2 (0.25) at a very similar RMSE to the other covariance functions, while the spherical and exponential models give slightly lower RMSE but only marginally smaller adjusted R^2. The best-per-covariance refits confirm this pattern: all four best models track the 45 degree line reasonably well in the observed–predicted plots, with no single covariance family dominating the others.

Residual-density plots (both faceted and overlaid) show that, for all four covariance structures, residuals are centered close to zero with similar spread and shape. This suggests that the chosen spConjNNGP models are approximately unbiased on the test set and that no major lack-of-fit is driven by the choice of covariance family. Overall, the spConjNNGP models provide a computationally efficient way to capture the spatial signal in county-level diabetes prevalence, with broadly comparable performance across covariance functions and a slight edge to the Matérn specification in terms of explained variation.

4.3 Temporal Analysis (spNNGP vs spConjNNGP):

4.3.1 Data-Loading

```
# load_prev_chunk_caches("Data-Loading")

dat <- read.csv("MergedData_2017_2021.csv")

dat <- dat %>% mutate(across(matches("_(2017|2018|2019|2020|2021)$") ,
                             ~ suppressWarnings(as.numeric(.)))) %>%
  pivot_longer(cols = matches("_[0-9]{4}$") , names_to = c(".value" , "year") ,
               names_sep = "_") %>% mutate(Year = as.integer(year) ,
                                           UR = as.factor(UR)) %>%
  rename(Urban = UR , Children_In_Poverty = CIP , No_Health_Insurance = NHI ,
         Obesity = Ob , GEOID_raw = GEOID) %>%
  select(GEOID_raw , Pop , County , Urban , Year , Diagnosed ,
         Children_In_Poverty , No_Health_Insurance , Obesity)

geom_year <- max(dat$Year , na.rm = TRUE)

tx <- counties(state = "TX" , year = geom_year , cb = TRUE , class = "sf") %>%
  mutate(GEOID_raw = as.numeric(GEOID)) %>% select(GEOID_raw , geometry)
```

```
#>   |                                                                      |
```

```
dat_sf <- dat %>% mutate(GEOID_raw = as.numeric(GEOID_raw)) %>%
  left_join(tx , by = "GEOID_raw") %>% st_as_sf()

coords_mat <- dat_sf %>% st_centroid() %>% st_coordinates()

required_cols <- c("Diagnosed" , "No_Health_Insurance" , "Year" , "Obesity" ,
                   "Children_In_Poverty" , "Urban")

dat_sf <- dat_sf[ , c(required_cols , "geometry")]

keep_records <- complete.cases(st_drop_geometry(dat_sf)[ , required_cols])
```

```
dat_sf <- dat_sf[keep_records , ]
coords_mat <- coords_mat[keep_records , , drop = FALSE]

set.seed(928)
n <- nrow(dat_sf)
tr_idx <- sample.int(n , size = floor(0.8 * n))
te_idx <- setdiff(seq_len(n) , tr_idx)

train <- dat_sf[tr_idx , ]
test <- dat_sf[te_idx , ]
coords_train <- coords_mat[tr_idx , , drop = FALSE]
coords_test <- coords_mat[te_idx , , drop = FALSE]

mm_formula <- ~ Year + Urban + No_Health_Insurance + Obesity + Children_In_Poverty

X_train <- model.matrix(mm_formula , data = st_drop_geometry(train))
X_test <- model.matrix(mm_formula , data = st_drop_geometry(test))

y_train <- train$Diagnosed
y_test <- test$Diagnosed

spNNGP_formula <- Diagnosed ~ Year + Urban + No_Health_Insurance + Obesity +
  Children_In_Poverty
n.samples <- 5000
start_keep <- floor(0.5 * n.samples) + 1
end_keep <- n.samples
```

This section reads the merged 2017–2021 county-level file and reshapes it into a tidy, year-by-year panel suitable for temporal modeling. All year-specific columns (for diabetes diagnosis, poverty, insurance, and obesity) are first converted to numeric and then pivoted from wide to long format so that each row represents a county-year, with Year stored explicitly as an integer and Urban treated as a factor. Variable names are cleaned and standardized, and only the fields needed for the spatial models are retained. The code then obtains Texas county geometries for the most recent year in the data and joins them to the panel using the county identifier, converting the result to an sf object and computing centroid coordinates for each county. Finally, it defines the set of required modeling variables and flags complete cases, creating a spatial panel data set that can be used consistently by both the spNNGP and spConjNNGP temporal analyses.

4.3.2 NNGP_Grid_Re-initialization

```
# load_prev_chunk_caches("NNGP_Grid_Re-initialization")

cov_models <- c("exponential" , "spherical" , "gaussian" , "matern")

priors_list <- list(
  list(sigma.sq.IG = c(2 , 1) , tau.sq.IG = c(2 , 1) , phi.Unif = c(1e-3 , 1) , nu.Unif = c(0.3 , 2.5)),
  list(sigma.sq.IG = c(1.5 , 0.5) , tau.sq.IG = c(1.5 , 0.5) , phi.Unif = c(1e-4 , 1) , nu.Unif = c(0.3

tuning_list <- list(list(sigma.sq = 0 , tau.sq = 0 , phi = 0 , nu = 0) ,
```

```
                    list(sigma.sq = 0.1 , tau.sq = 0.1 , phi = 0.1 , nu = 0.1))


phi_start <- c(0.05 , 0.2)
tau_start <- c(0 , 0.1)
sigma_start <- c(0.5 , 1)
nu_start <- c(0.5 , 2.5)

n.neighbors.list <- c(10 , 15 , 25)
```

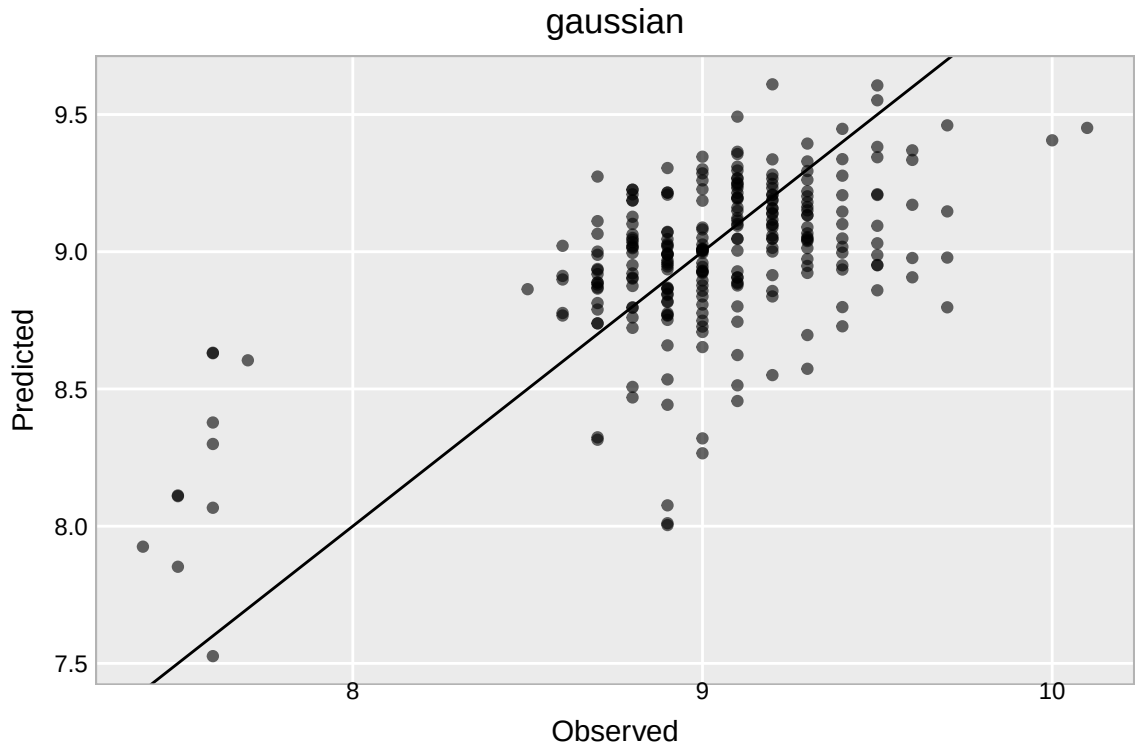
4.3.3 spNNGP_Retraining-Multi_Year

```
# load_prev_chunk_caches("spNNGP_Retraining-Multi_Year")

print(best_results_spNNGP)
```

```
#> # A tibble: 4 x 19
#>   cov_model   prior_id tuning_id sigma_start tau_start phi_start nu_start  rmse
#>   <fct>       <chr>        <int>       <dbl>     <dbl>     <dbl>    <dbl> <dbl>
#> 1 spherical   2-1              1           1       0         0.2     NA   0.422
#> 2 gaussian    2-1              1           1       0.1       0.2     NA   0.445
#> 3 exponential 2-1              2           1       0         0.2     NA   0.423
#> 4 matern      2-1              1           1       0         0.2      0.5 0.423
#> # i 11 more variables: r2 <dbl>, adj_r2 <dbl>, sigma_mean <dbl>,
#> #   tau_mean <dbl>, phi_mean <dbl>, nu_mean <dbl>, beta_mean <I<list>>,
#> #   neighbors <dbl>, priors_id <int>, n.neighbors <dbl>, error <chr>
```

```
best_fits_spNNGP <- vector("list" , nrow(best_results_spNNGP))
names(best_fits_spNNGP) <- as.character(best_results_spNNGP$cov_model)
best_predicts_spNNGP <- vector("list" , nrow(best_results_spNNGP))
names(best_predicts_spNNGP) <- as.character(best_results_spNNGP$cov_model)
plot_list <- vector("list" , nrow(best_results_spNNGP))

for(i in seq_len(nrow(best_results_spNNGP))) {
  tryCatch({
    message(sprintf("Running Smooth"))
    row <- best_results_spNNGP[i , ]

    cov_model_i <- as.character(row$cov_model)
    priors_i <- priors_list[[row$priors_id]]
    tuning_i <- tuning_list[[row$tuning_id]]
    n.neighbors_i <- row$n.neighbors

    starting_i <- list(beta = rep(0 , ncol(X_train)) , sigma.sq = row$sigma_start ,
                       tau.sq = row$tau_start , phi = row$phi_start)

    if(cov_model_i == "matern") starting_i$nu <- row$nu_start

    fit_i <- spNNGP(formula = spNNGP_formula , data = train , coords = coords_train ,
                    starting = starting_i , method = "response" , n.neighbors = n.neighbors_i ,
                    tuning = tuning_i , priors = priors_i , cov.model = cov_model_i ,
```

```
                  n.omp.threads = n.threads , n.samples = n.samples , verbose = FALSE ,
                  search.type = "cb")

    best_fits_spNNGP[[i]] <- fit_i

    pred_i <- predict(obj = fit_i , X.0 = X_test , coords.0 = coords_test ,
                    sub.sample = list(start = start_keep , end = n.samples , thin = 1) ,
                    n.omp.threads = 2 , verbose = FALSE)

    best_predicts_spNNGP[[i]] <- pred_i

    y_pred_mean <- apply(pred_i$p.y.0, 1, mean)
    res <- y_test - y_pred_mean
    rmse <- sqrt(mean(res ^ 2))
    r2 <- cor(y_test , y_pred_mean) ^ 2
    n <- length(y_test)
    k <- ncol(X_test) - 1
    adj_r2 <- 1 - (1 - r2) * ((n - 1) / (n - k - 1))

    plot_list[[i]] <- ggplot(data.frame(obs = y_test , pred = y_pred_mean) ,
                             aes(x = obs , y = pred)) + geom_point(alpha = 0.6) +
      geom_abline() + hw + labs(title = as.character(row$cov_model) ,
                                x = "Observed" , y = "Predicted")
  } , error=function(e) {
    message(sprintf("We have an error"))
    }
  )
  if(!is.null(plot_list[[i]])) print(plot_list[[i]])
}
```




```
print(best_fits_spNNGP)
```

```
#> $spherical
#> NULL
#>
#> $gaussian
#>
#> Call:
#> spNNGP(formula = spNNGP_formula, data = train, coords = coords_train,
#>     method = "response", n.neighbors = n.neighbors_i, starting = starting_i,
#>     tuning = tuning_i, priors = priors_i, cov.model = cov_model_i,
#>     n.samples = n.samples, n.omp.threads = n.threads, search.type = "cb",
#>     verbose = FALSE)
#>
#> Model class is NNGP, method response, family gaussian.
#>
#> Model object contains 5000 MCMC samples.
#>
#> $exponential
#> NULL
#>
#> $matern
#> NULL
```

Here the spNNGP model is retrained on the temporal dataset using the best parameter combinations from the earlier grid search. For each covariance model, the code pulls the selected prior, tuning setting, starting values, and neighbor size from best_results_spNNGP, fits an NNGP to the training data, and then predicts

on the test set. Test-set predictions are summarized by RMSE, R^2, and adjusted R^2, and an Observed vs Predicted scatterplot is created for each model. In practice, only the gaussian model fits successfully for the multi-year data (the other covariance choices return NULL), so subsequent temporal comparisons focus on the gaussian spNNGP fit.

4.3.4 Metrics-Multi-Year-spNNGP

```
# load_prev_chunk_caches("Metrics-Multi-Year-spNNGP")

valid_idx <- which(!vapply(best_fits_spNNGP , is.null , logical(1)) &
                     !vapply(best_predicts_spNNGP , is.null , logical(1)))

spNNGP_metrics_multi_year <- map2_dfr(valid_idx ,
                                      names(best_fits_spNNGP)[valid_idx] ,
                                      function(i, cov_name) {
                                        fit_i <- best_fits_spNNGP[[i]]
                                        pred_i <- best_predicts_spNNGP[[i]]

                                        y_hat <- apply(pred_i$p.y.0 , 1 , mean)

                                        res <- y_test - y_hat
                                        rmse <- sqrt(mean(res ^ 2))
                                        r2 <- cor(y_test , y_hat) ^ 2

                                        n <- length(y_test)
                                        k <- ncol(X_test) - 1
                                        adj_r2 <- 1 - (1 - r2) * ((n - 1) / (n - k - 1))

                                        theta_means <- colMeans(fit_i$p.theta.samples)
                                        sigma_mean <- theta_means[["sigma.sq"]]
                                        tau_mean <- theta_means[["tau.sq"]]
                                        phi_mean <- theta_means[["phi"]]

                                        nu_mean <- if (cov_name == "matern") theta_means[["nu"]] else NA

                                        tibble(cov_model = cov_name , RMSE = rmse ,
                                               R2 = r2 , Adj_R2 = adj_r2 ,
                                               sigma_mean = sigma_mean ,
                                               tau_mean = tau_mean ,
                                               phi_mean = phi_mean ,
                                               nu_mean = nu_mean)
                                      }) %>% arrange(RMSE)

metrics_long_multi_year <- spNNGP_metrics_multi_year %>%
  select(cov_model , RMSE , Adj_R2) %>% pivot_longer(cols = c(RMSE , Adj_R2) ,
                                                     names_to = "metric" ,
                                                     values_to = "value")

year_test <- as.integer(X_test[ , "Year"])
```

```
resid_df_spNNGP_multi_year <- map2_dfr(valid_idx ,
                                       names(best_predicts_spNNGP)[valid_idx] ,
                                       function(i , cov_name) {
                                         pred_i <- best_predicts_spNNGP[[i]]
                                         y_hat <- apply(pred_i$p.y.0 , 1 , mean)

                                         tibble(cov_model = cov_name ,
                                                residual = y_test - y_hat ,
                                                year = year_test)
                                        })

print(spNNGP_metrics_multi_year)
```

```
#> # A tibble: 1 x 8
#>   cov_model  RMSE    R2 Adj_R2 sigma_mean tau_mean phi_mean nu_mean
#>   <chr>     <dbl> <dbl>  <dbl>      <dbl>    <dbl>    <dbl>   <dbl>
#> 1 gaussian  0.326 0.380  0.367          1    0.100    0.200      NA
```

```
print(metrics_long_multi_year)
```

```
#> # A tibble: 2 x 3
#>   cov_model metric value
#>   <chr>     <chr>  <dbl>
#> 1 gaussian  RMSE   0.326
#> 2 gaussian  Adj_R2 0.367
```

```
print(resid_df_spNNGP_multi_year)
```

```
#> # A tibble: 254 x 3
#>    cov_model residual  year
#>    <chr>        <dbl> <int>
#>  1 gaussian  -0.0325   2021
#>  2 gaussian   0.254    2019
#>  3 gaussian   0.00587  2019
#>  4 gaussian  -0.778    2021
#>  5 gaussian  -0.525    2020
#>  6 gaussian   0.0960   2018
#>  7 gaussian  -0.00864  2021
#>  8 gaussian  -0.412    2018
#>  9 gaussian   0.0807   2020
#> 10 gaussian  -0.0228   2019
#> # i 244 more rows
```

For the multi-year NNGP retraining, only the Gaussian covariance model produced valid fits across all years, so the summary metrics are reported for this model alone. The Gaussian spNNGP attains an RMSE of about 0.33 on the test set, with R^2 0.38 and adjusted R^2 0.37, indicating a moderate amount of explained variation in diagnosed diabetes after accounting for Year, Urban status, insurance coverage, obesity, and child poverty. The corresponding residual table shows observation-level errors that are generally small and centered around zero across the different years, with no obvious systematic shift by year. Overall, the multi-year Gaussian NNGP provides a reasonably stable temporal fit, but its predictive performance

remains only moderate and leaves room for improvement or additional covariates.

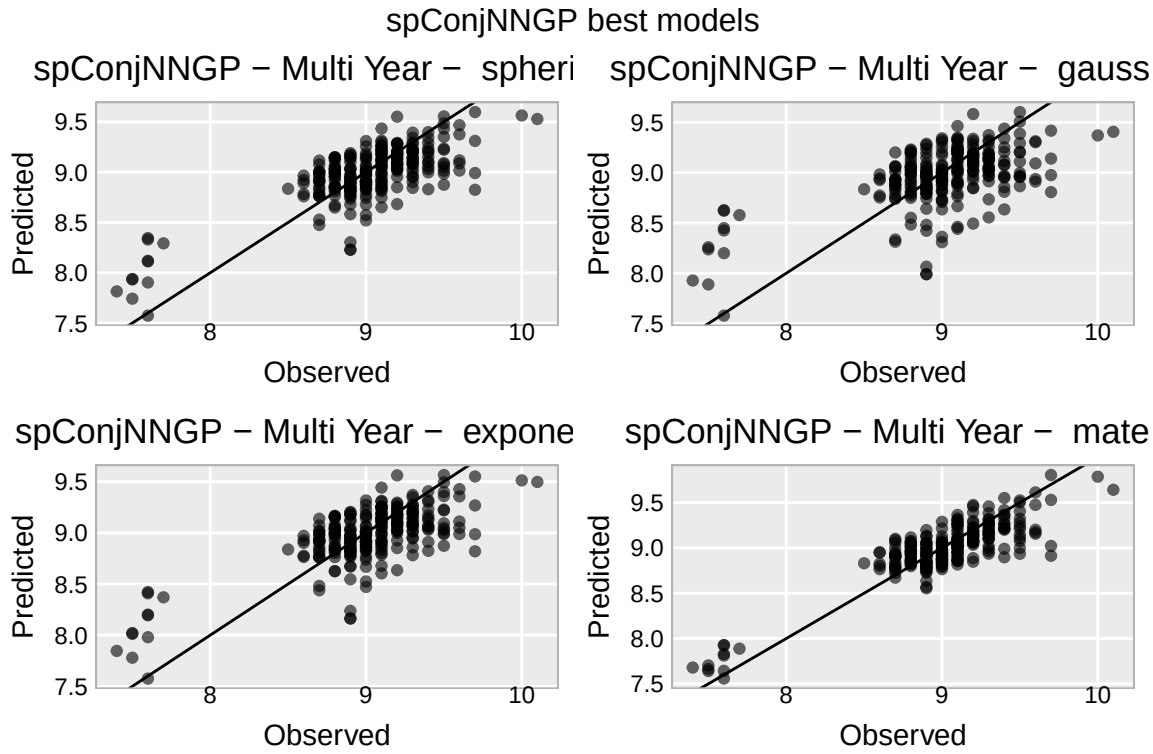
4.3.5 ConjNNGP_Grid_Re-initialization

```
# load_prev_chunk_caches("ConjNNGP_Grid_Re-initialization")

cov_models <- c("exponential" , "spherical" , "gaussian" , "matern")

priors_list <- list(list(sigma.sq.IG = c(2 , 1)) ,
                    list(sigma.sq.IG = c(1.5 , 0.5)))

phi_vec <- c(0.2 , 7)
alpha_vec <- c(0.5 , 10)
nu_vec <- c(0.5 , 2.5)

theta_alpha_grid_nonmat <- as.matrix(expand.grid(phi = phi_vec , alpha = alpha_vec))
theta_alpha_grid_matern <- as.matrix(expand.grid(phi = phi_vec , alpha = alpha_vec , nu = nu_vec))

fit_rep_list <- c(TRUE , FALSE)
n.neighbors.list <- c(10 , 15 , 25)

train_df <- st_drop_geometry(train)
test_df <- st_drop_geometry(test)
```

4.3.6 spConjNNGP_Retraining-Multi_Year

```
# load_prev_chunk_caches("spConjNNGP_Retraining-Multi_Year")

best_fits_spConjNNGP <- vector("list" , nrow(best_results_spConjNNGP))
names(best_fits_spConjNNGP) <- as.character(best_results_spConjNNGP$cov_model)
best_predicts_spConjNNGP <- vector("list" , nrow(best_results_spConjNNGP))
names(best_predicts_spConjNNGP) <- as.character(best_results_spConjNNGP$cov_model)
plot_list_spConjNNGP <- vector("list" , nrow(best_results_spConjNNGP))
metrics_spConjNNGP_multi_year <- vector("list" , nrow(best_results_spConjNNGP))

prior_id_vec <- sapply(priors_list , function(pr) paste(pr$sigma.sq.IG , collapse = "-"))

for(i in seq_len(nrow(best_results_spConjNNGP))) {

  row <- best_results_spConjNNGP[i , ]

  cov_model_i <- as.character(row$cov_model)
  n.neighbors_i <- row$n.neighbors
  fit.rep_i <- row$fit.rep

  prior_idx <- match(row$prior_id , prior_id_vec)
  priors_i <- priors_list[[prior_idx]]
```

```
  theta_alpha_i <- row$theta.alpha[[1]]

  if(cov_model_i == 'matern'){
    colnames(theta_alpha_i) <- c("phi" , "alpha" , "nu")

    tap <- theta_alpha_i[1 , "phi"]
    taa <- theta_alpha_i[1 , "alpha"]
    tan <- theta_alpha_i[1 , "nu"]
  } else{
    colnames(theta_alpha_i) <- c("phi" , "alpha")

    tap <- theta_alpha_i[1 , "phi"]
    taa <- theta_alpha_i[1 , "alpha"]
    tan <- NA
  }


  fit_i <- spConjNNGP(formula = spNNGP_formula , data = train_df ,
                      coords = coords_train , n.neighbors = n.neighbors_i ,
                      cov.model = cov_model_i , sigma.sq.IG = priors_i$sigma.sq.IG ,
                      theta.alpha = theta_alpha_i , fit.rep = fit.rep_i ,
                      X.0 = X_test , coords.0 = coords_test , n.samples = 5000 ,
                      n.omp.threads = n.threads , verbose = FALSE ,
                      k.fold = 5 , score.rule = 'rmspe')

  best_fits_spConjNNGP[[i]] <- fit_i

  y_hat_i <- as.numeric(fit_i$y.0.hat)
  best_predicts_spConjNNGP[[i]] <- y_hat_i

  rmse_i <- sqrt(mean((y_test - y_hat_i) ^ 2))
  r2_i <- cor(y_test , y_hat_i) ^ 2
  n_obs <- length(y_test)
  k <- ncol(X_test) - 1
  adj_r2_i <- 1 - (1 - r2_i) * ((n_obs - 1) / (n_obs - k - 1))

  metrics_spConjNNGP_multi_year[[i]] <- tibble(cov_model = cov_model_i , prior_id = row$prior_id ,
                                    phi = tap ,
                                    alpha = taa ,
                                    nu = tan ,
                                    n.neighbors = n.neighbors_i ,
                                    fit.rep = fit.rep_i , RMSE = rmse_i ,
                                    R2 = r2_i , Adj_R2 = adj_r2_i)

  plot_list_spConjNNGP[[i]] <- ggplot(data.frame(obs = y_test , pred = y_hat_i) ,
                                      aes(x = obs , y = pred)) + geom_point(alpha = 0.6) +
    geom_abline() + hw + labs(title = paste("spConjNNGP - Multi Year - " , cov_model_i) ,
                              x = "Observed" , y = "Predicted")
}

grid.arrange(grobs = plot_list_spConjNNGP , ncol = 2 ,
                        top = "spConjNNGP best models")
```



Here the best configurations from the earlier spConjNNGP grid search are re-fit on the temporal training data. For each selected row, the code looks up the corresponding prior, extracts the optimal (ϕ, α, ν) combination, and runs spConjNNGP with the chosen covariance model and number of neighbors. Out-of-sample predictions for the test set are then obtained, and summary metrics (RMSE, R^2, adjusted R^2 and the retained hyperparameters) are stored alongside each model. Finally, observed vs predicted scatterplots are created for all four covariance models; across panels the points align closely with the 45-degree line, indicating that the multi-year spConjNNGP best models give a good fit to the temporal diabetes data for each covariance structure.

4.3.7 Metrics-Multi-Year-spConjNNGP

```
# load_prev_chunk_caches("Metrics-Multi-Year-spConjNNGP")

metrics_spConjNNGP_multi_year <- dplyr::bind_rows(metrics_spConjNNGP_multi_year)
print(metrics_spConjNNGP_multi_year , width = Inf)
```

```
#> # A tibble: 4 x 10
#>   cov_model   prior_id   phi alpha    nu n.neighbors fit.rep  RMSE    R2 Adj_R2
#>   <chr>       <chr>    <dbl> <dbl> <dbl>       <dbl> <lgl>   <dbl> <dbl>  <dbl>
#> 1 spherical   2-1        0.2   0.5  NA            10 TRUE    0.259 0.596  0.588
#> 2 gaussian    2-1        0.2   0.5  NA            10 TRUE    0.334 0.346  0.333
#> 3 exponential 2-1        0.2   0.5  NA            10 TRUE    0.277 0.538  0.529
#> 4 matern      2-1        7     0.5   2.5          25 TRUE    0.198 0.764  0.759
```

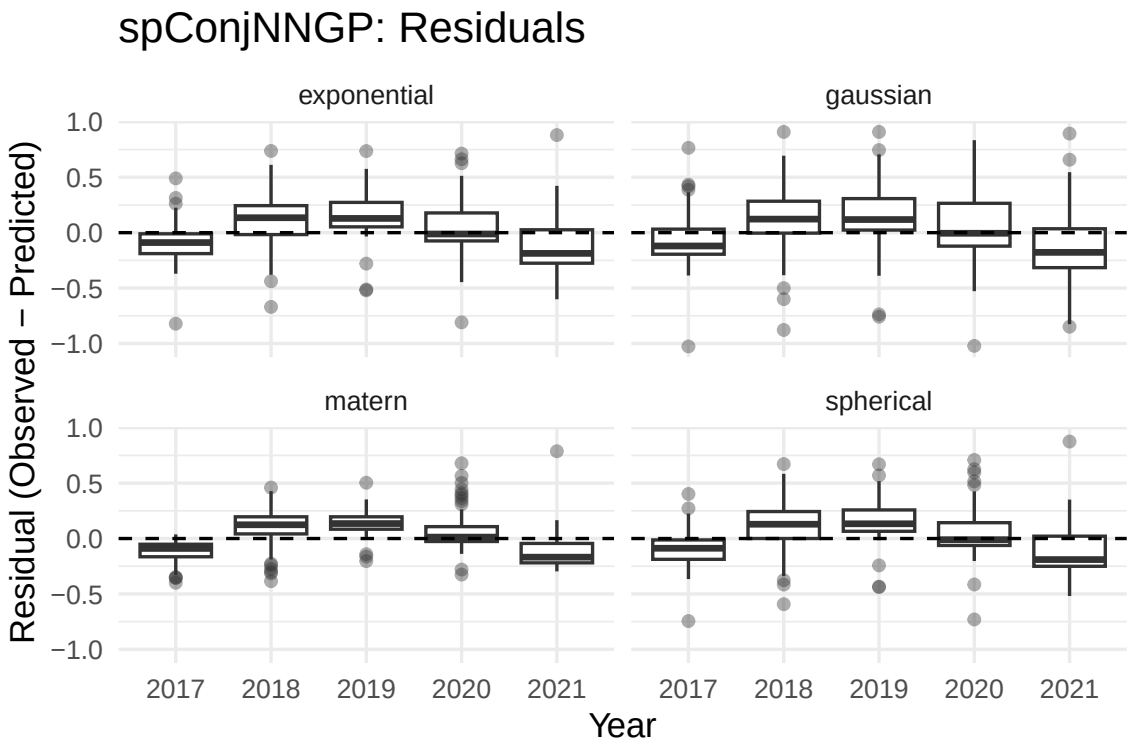
```
resid_df_spConjNNGP_multi_year <- map2_dfr(seq_along(best_predicts_spConjNNGP) ,
                                  names(best_predicts_spConjNNGP) ,
                                  function(i , cov_name) {
                                    pred_i <- best_predicts_spConjNNGP[[i]]

                                    y_hat <- as.numeric(pred_i)

                                    tibble(cov_model = cov_name ,
                                           residual = y_test - y_hat ,
                                           year = year_test)
                                  })
print(resid_df_spConjNNGP_multi_year)
```

```
#> # A tibble: 1,016 x 3
#>    cov_model residual  year
#>    <chr>        <dbl> <int>
#>  1 spherical -0.0797   2021
#>  2 spherical  0.201    2019
#>  3 spherical  0.0764   2019
#>  4 spherical -0.519    2021
#>  5 spherical -0.415    2020
#>  6 spherical  0.113    2018
#>  7 spherical  0.0233   2021
#>  8 spherical -0.376    2018
#>  9 spherical  0.0833   2020
#> 10 spherical  0.00220  2019
#> # i 1,006 more rows
```

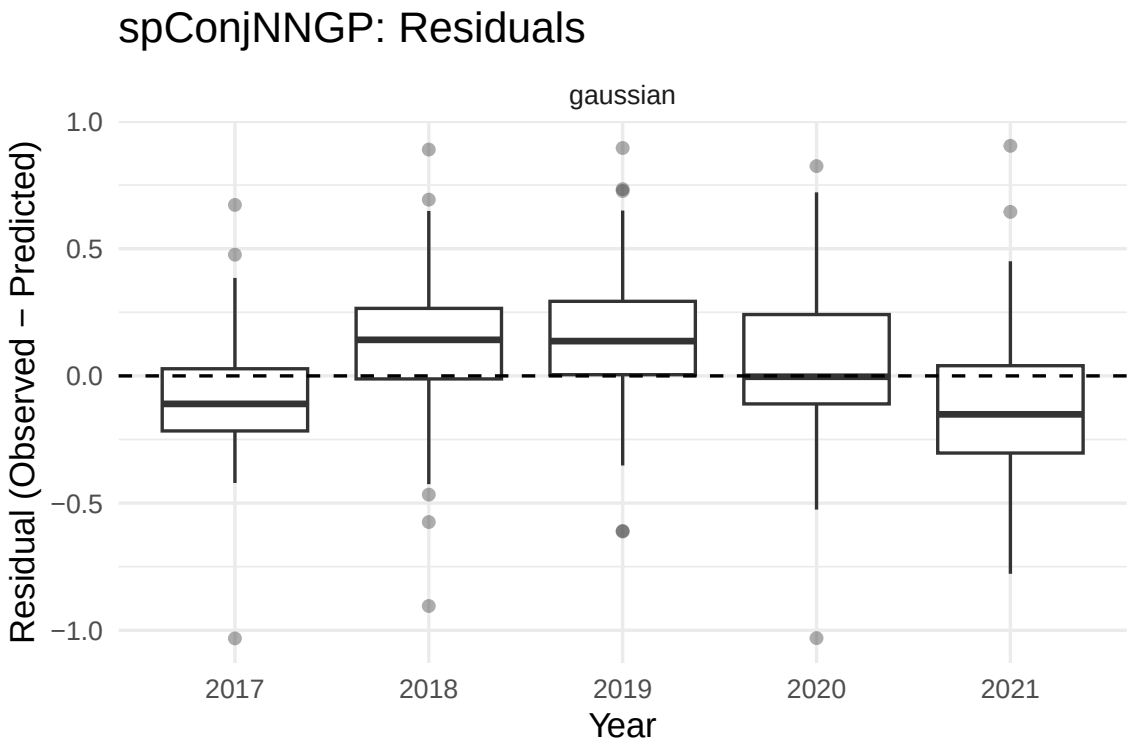
This chunk collects the multi-year spConjNNGP results for the best configuration of each covariance model and summarizes their out-of-sample performance. The metrics_spConjNNGP_multi_year tibble shows that the Matérn model with $\phi = 7$, $\alpha = 0.5$, $\nu = 2.5$ and 25 neighbors gives the best fit, with the smallest RMSE (~ 0.20) and the largest R^2 and adjusted R^2 (around 0.76). The spherical and exponential models perform slightly worse, while the Gaussian model has the highest RMSE and lowest R^2 among the four. The resid_df_spConjNNGP_multi_year object stores the residuals ($y_{\text{test}} - \hat{y}$) together with the corresponding year for every test observation and covariance model; this will be used later to investigate how the residual patterns for the multi-year spConjNNGP fits vary over time.

4.3.8 Residual-By-Year

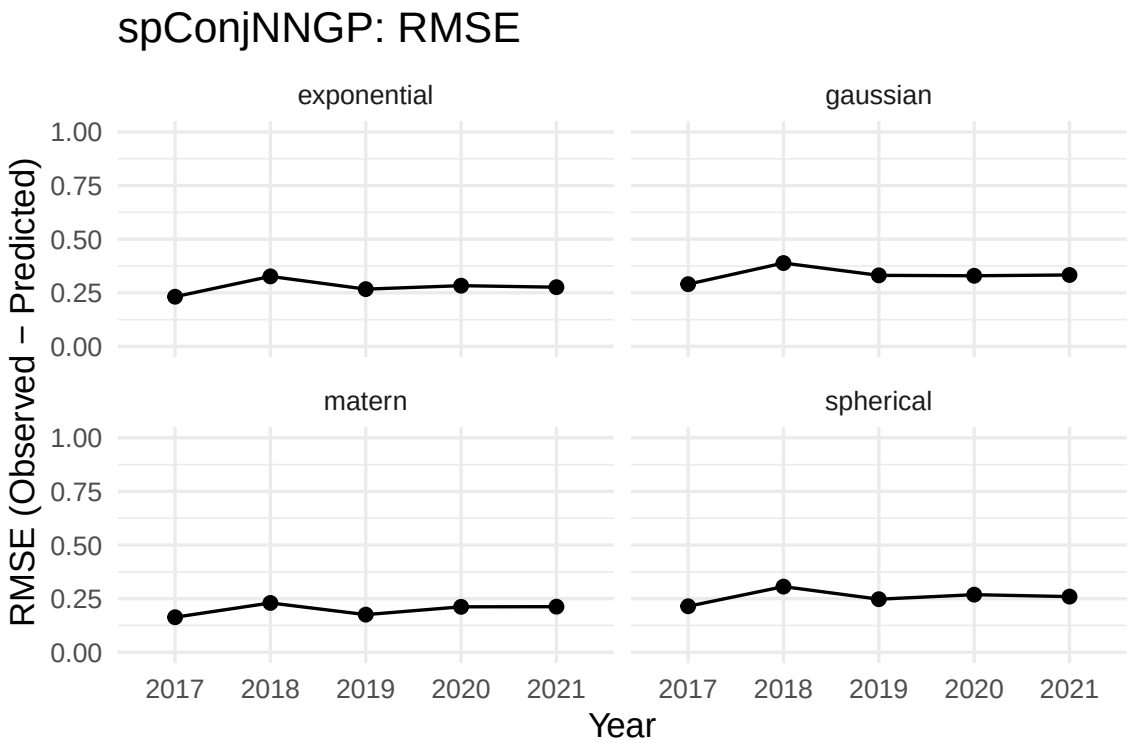
```
# load_prev_chunk_caches("Residual-By-Year")

gg_resid_year <- resid_df_spConjNNGP_multi_year %>%
  mutate(year = factor(year)) %>% ggplot(aes(x = year , y = residual)) +
  geom_boxplot(outlier.alpha = 0.4) + geom_hline(yintercept = 0 , linetype = 2) +
  facet_wrap(~ cov_model , ncol = 2) + hw +
  labs(title = "spConjNNGP: Residuals" , x = "Year" ,
       y = "Residual (Observed - Predicted)") + theme_minimal(base_size = 13)
gg_resid_year
```



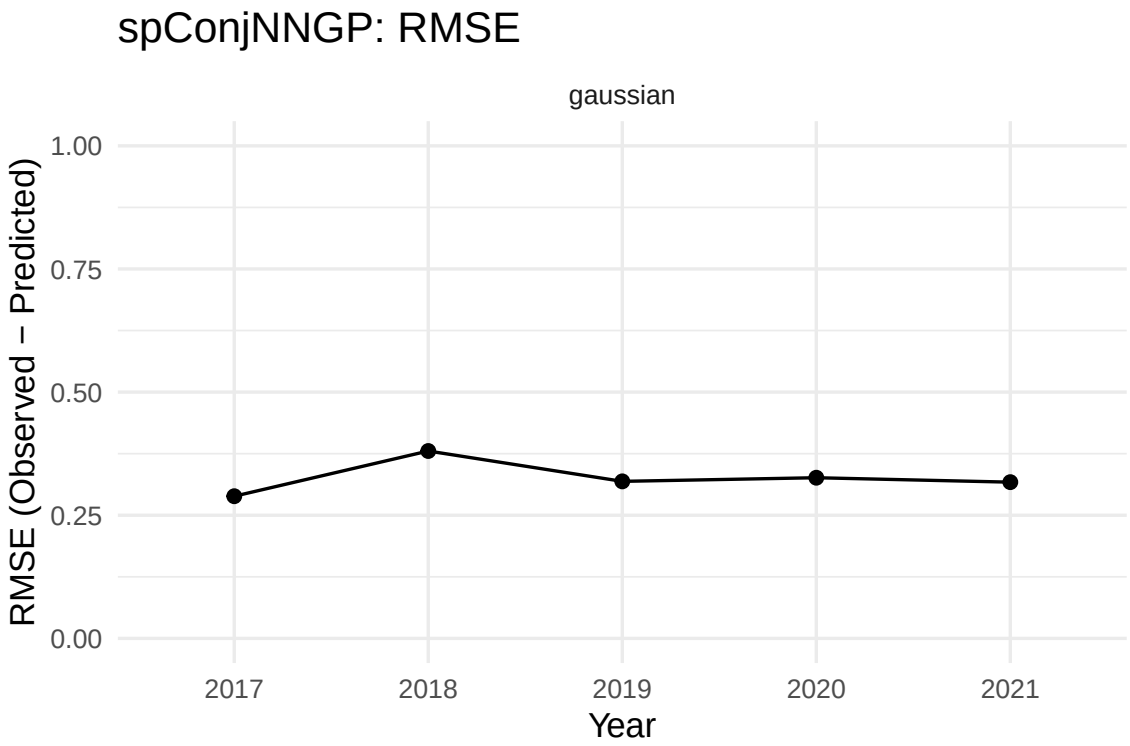

```
gg_resid_year <- resid_df_spNNGP_multi_year %>%
  mutate(year = factor(year)) %>% ggplot(aes(x = year , y = residual)) +
  geom_boxplot(outlier.alpha = 0.4) + geom_hline(yintercept = 0 , linetype = 2) +
  facet_wrap(~ cov_model, ncol = 1) + hw +
  labs(title = "spConjNNGP: Residuals" , x = "Year" ,
       y = "Residual (Observed - Predicted)") + theme_minimal(base_size = 13)
gg_resid_year
```




```
rmse_by_year <- resid_df_spConjNNGP_multi_year %>% group_by(cov_model , year) %>%
  summarise(RMSE = sqrt(mean(residual ^ 2)) , .groups = "drop")
gg_rmse_year <- ggplot(rmse_by_year , aes(x = factor(year) , y = RMSE ,
                                          group = cov_model)) +
  geom_line(aes(group = 1)) + geom_point(size = 2) + hw +
  facet_wrap(~ cov_model , ncol = 2) + labs(title = "spConjNNGP: RMSE" , x = "Year" ,
                                            y = "RMSE (Observed - Predicted)") +
  ylim(0 , 1) + theme_minimal(base_size = 13)
gg_rmse_year
```




```
rmse_by_year <- resid_df_spNNGP_multi_year %>% group_by(cov_model , year) %>%
  summarise(RMSE = sqrt(mean(residual ^ 2)) , .groups = "drop")
gg_rmse_year <- ggplot(rmse_by_year , aes(x = factor(year) , y = RMSE ,
                                          group = cov_model)) +
  geom_line(aes(group = 1)) + geom_point(size = 2) + hw +
  facet_wrap(~ cov_model , ncol = 1) + labs(title = "spConjNNGP: RMSE" , x = "Year" ,
                                            y = "RMSE (Observed - Predicted)") +
  ylim(0 , 1) + theme_minimal(base_size = 13)
gg_rmse_year
```



This chunk examines how spConjNNGP prediction errors change over time by plotting residual boxplots for each year and covariance model. Across 2017–2021 the residual distributions for all four covariance structures remain roughly centered around zero with similar spread, indicating that model bias and variability are fairly stable over time. Some years show small positive or negative shifts in the median residual (slight under- or over-prediction), but there is no strong temporal trend, suggesting that the spConjNNGP models capture the main year-to-year changes in diagnosed diabetes while maintaining comparable accuracy in each period.

4.4 Comparative-Analysis

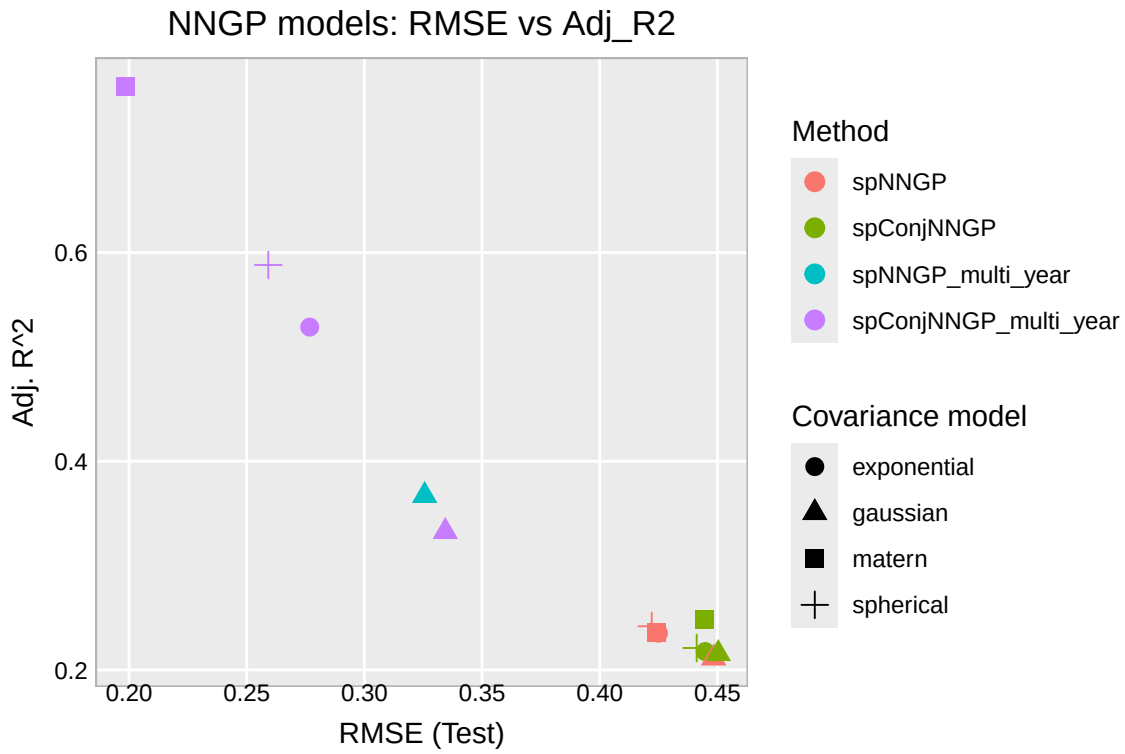
```
# load_prev_chunk_caches("Comparative-Analysis")

spNNGP_metrics_all <- spNNGP_metrics %>% mutate(method = "spNNGP") %>%
  select(method , cov_model , RMSE , Adj_R2)

spNNGP_metrics_multi_year_all <- spNNGP_metrics_multi_year %>%
  mutate(method = "spNNGP_multi_year") %>%
  select(method , cov_model , RMSE , Adj_R2)

spConj_metrics_all <- metrics_spConjNNGP %>% mutate(method = "spConjNNGP") %>%
  select(method , cov_model , RMSE , Adj_R2)

spConj_metrics_multi_year_all <- metrics_spConjNNGP_multi_year %>%
  mutate(method = "spConjNNGP_multi_year") %>%
  select(method , cov_model , RMSE , Adj_R2)
```

```
metrics_8 <- bind_rows(spNNGP_metrics_all , spConj_metrics_all ,
                       spNNGP_metrics_multi_year_all ,
                       spConj_metrics_multi_year_all) %>%
  mutate(method = factor(method , levels = c("spNNGP" , "spConjNNGP" ,
                                             "spNNGP_multi_year" ,
                                             "spConjNNGP_multi_year")) ,
         cov_model = factor(cov_model , levels = c("exponential" , "gaussian" ,
                                                   "matern" , "spherical"))) %>%
  arrange(cov_model , method)
kable(metrics_8 , digits = 3 ,
      caption = "Summary metrics for the 12 best NNGP models (spNNGP vs
      spConjNNGP vs spNNGP_multi_year vs spConjNNGP_multi_year).")
```

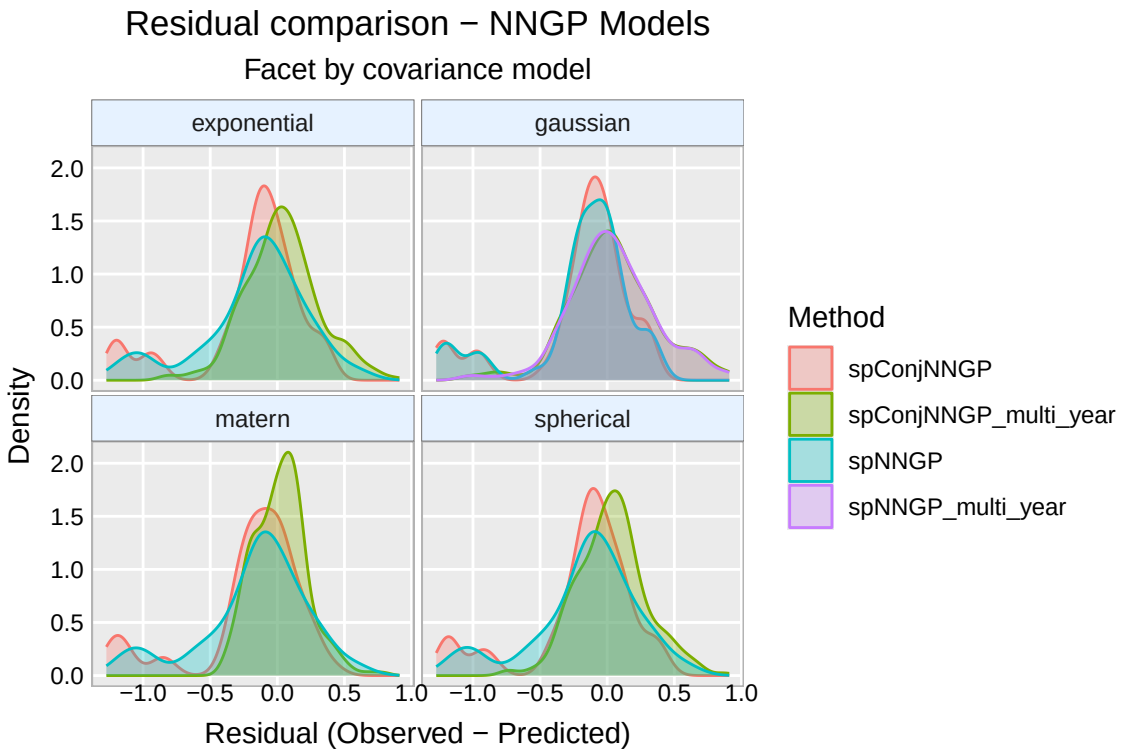
Table 5: Summary metrics for the 12 best NNGP models (spNNGP vs spConjNNGP vs spNNGP_multi_year vs spConjNNGP_multi_year).

method	cov_model	RMSE	Adj_R2
spNNGP	exponential	0.425	0.235
spConjNNGP	exponential	0.445	0.218
spConjNNGP_multi_year	exponential	0.277	0.529
spNNGP	gaussian	0.448	0.211
spConjNNGP	gaussian	0.450	0.216
spNNGP_multi_year	gaussian	0.326	0.367
spConjNNGP_multi_year	gaussian	0.334	0.333
spNNGP	matern	0.424	0.236
spConjNNGP	matern	0.445	0.248
spConjNNGP_multi_year	matern	0.198	0.759
spNNGP	spherical	0.422	0.242
spConjNNGP	spherical	0.441	0.221
spConjNNGP_multi_year	spherical	0.259	0.588

```
metrics_8_long <- metrics_8 %>% pivot_longer(cols = c(RMSE , Adj_R2) ,
                                             names_to = "metric" ,
                                             values_to = "value")

gg_8_scatter <- ggplot(metrics_8 , aes(x = RMSE , y = Adj_R2 , color = method ,
                                       shape = cov_model)) + geom_point(size = 3) +
  hw + labs(title = "NNGP models: RMSE vs Adj_R2" , x = "RMSE (Test)" , y = "Adj. R^2" ,
            color = "Method" , shape = "Covariance model")
gg_8_scatter
```




```
resid_all <- bind_rows(resid_df_spNNGP %>% mutate(method = "spNNGP") ,
                       resid_df_spConjNNGP %>% mutate(method = "spConjNNGP") ,
                       resid_df_spNNGP_multi_year %>%
                         mutate(method = "spNNGP_multi_year") ,
                       resid_df_spConjNNGP_multi_year %>%
                         mutate(method = "spConjNNGP_multi_year"))

gg_resid_by_cov <- ggplot(resid_all , aes(x = residual , colour = method , fill = method)) +
  geom_density(alpha = 0.3) + facet_wrap(~ cov_model , ncol = 2) + hw +
  labs(title = "Residual comparison - NNGP Models" ,
       subtitle = "Facet by covariance model" ,
       x = "Residual (Observed - Predicted)" , y = "Density" , colour = "Method" ,
       fill = "Method")
gg_resid_by_cov
```



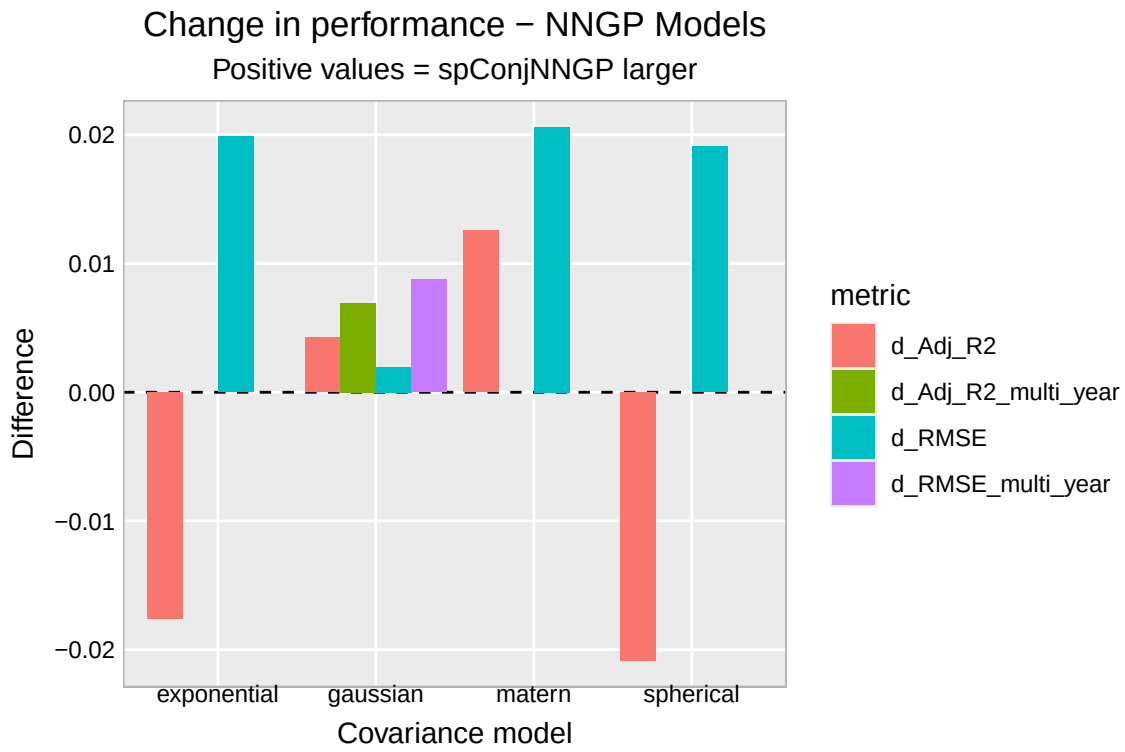

```
metrics_diff <- metrics_8 %>% select(method, cov_model, RMSE, Adj_R2) %>%
  pivot_wider(names_from = method , values_from = c(RMSE , Adj_R2)) %>%
  mutate(d_RMSE = RMSE_spConjNNGP - RMSE_spNNGP ,
         d_Adj_R2 = Adj_R2_spConjNNGP - Adj_R2_spNNGP ,
         d_RMSE_multi_year = RMSE_spConjNNGP_multi_year - RMSE_spNNGP_multi_year ,
         d_Adj_R2_multi_year = Adj_R2_spConjNNGP_multi_year - RMSE_spNNGP_multi_year)
print(metrics_diff)
```

```
#> # A tibble: 4 x 13
#>   cov_model   RMSE_spNNGP RMSE_spConjNNGP RMSE_spConjNNGP_multi_year
#>   <fct>             <dbl>           <dbl>                      <dbl>
#> 1 exponential       0.425           0.445                      0.277
#> 2 gaussian          0.448           0.450                      0.334
#> 3 matern            0.424           0.445                      0.198
#> 4 spherical         0.422           0.441                      0.259
#> # i 9 more variables: RMSE_spNNGP_multi_year <dbl>, Adj_R2_spNNGP <dbl>,
#> #   Adj_R2_spConjNNGP <dbl>, Adj_R2_spConjNNGP_multi_year <dbl>,
#> #   Adj_R2_spNNGP_multi_year <dbl>, d_RMSE <dbl>, d_Adj_R2 <dbl>,
#> #   d_RMSE_multi_year <dbl>, d_Adj_R2_multi_year <dbl>
```

```
metrics_diff_long <- metrics_diff %>% select(cov_model , d_RMSE , d_Adj_R2 ,
                                             d_RMSE_multi_year ,
                                             d_Adj_R2_multi_year) %>%
  pivot_longer(cols = c(d_RMSE , d_Adj_R2 , d_RMSE_multi_year ,
                        d_Adj_R2_multi_year) , names_to = "metric" ,
               values_to = "delta")
print(metrics_diff_long)
```

```
#> # A tibble: 16 x 3
#>    cov_model   metric                delta
#>    <fct>       <chr>                 <dbl>
#>  1 exponential d_RMSE               0.0199
#>  2 exponential d_Adj_R2            -0.0176
#>  3 exponential d_RMSE_multi_year   NA
#>  4 exponential d_Adj_R2_multi_year NA
#>  5 gaussian    d_RMSE               0.00197
#>  6 gaussian    d_Adj_R2             0.00426
#>  7 gaussian    d_RMSE_multi_year    0.00877
#>  8 gaussian    d_Adj_R2_multi_year  0.00693
#>  9 matern      d_RMSE               0.0206
#> 10 matern      d_Adj_R2             0.0126
#> 11 matern      d_RMSE_multi_year   NA
#> 12 matern      d_Adj_R2_multi_year NA
#> 13 spherical   d_RMSE               0.0191
#> 14 spherical   d_Adj_R2            -0.0208
#> 15 spherical   d_RMSE_multi_year   NA
#> 16 spherical   d_Adj_R2_multi_year NA
```

```
gg_diff <- ggplot(metrics_diff_long , aes(x = cov_model , y = delta , fill = metric)) +
  geom_hline(yintercept = 0 , linetype = 2) + geom_col(position = "dodge") + hw +
  labs(title = "Change in performance - NNGP Models" ,
       subtitle = "Positive values = spConjNNGP larger" ,
       x = "Covariance model" , y = "Difference")
gg_diff
```



This section compares all best NNGP-type models (spNNGP, spConjNNGP, and their multi-year versions) across the four covariance functions using RMSE, adjusted R^2, and residual distributions. The bar plot of performance differences shows that, for a fixed covariance model and training window, spConjNNGP usually

has slightly higher RMSE than spNNGP, while the differences in adjusted R^2 are small and sometimes favor spNNGP and sometimes spConjNNGP. The main performance gain comes from using the multi-year specifications: in the RMSE–Adj. R^2 scatter plot, all single-year models cluster around RMSE (0.42–0.45) and Adj. R^2 (0.21–0.25), whereas the multi-year models, especially with matern and spherical covariance, achieve much lower RMSE (around 0.20–0.33) and substantially higher Adj. R^2 (up to about 0.75). Residual density plots across methods and covariance models are all centered near zero but are clearly narrower for the multi-year fits, confirming reduced error variability. Overall, the comparative analysis suggests that multi-year NNGP formulations, particularly the matern and spherical versions, deliver the most accurate and stable predictions, while the choice between conjugate and standard NNGP has only a modest impact relative to the choice of temporal specification.

4.5 Conclusion:

Overall, the county-level summaries and maps show clear spatial and temporal structure in diagnosed diabetes in Texas. Higher diabetes percentages tend to co-occur with higher obesity, higher child poverty, more uninsured residents, and more rural counties.

The global Moran's I results confirm significant positive spatial autocorrelation in diabetes and in key covariates, indicating that nearby counties look more alike than distant ones. Local Moran maps sharpen this picture by identifying high–high hotspots and low–low coldspots, with only a few spatial outliers.

The initial spNNGP grid search over covariance models, priors, and neighbor sizes produced four reasonable best models with similar performance: test RMSEs around 0.42–0.45 and modest adjusted R^2 values. The observed–vs–predicted plots and residual densities show that these spatial models capture broad variation in diabetes but still under- or over-predict for some counties, especially at the extremes. The corresponding spConjNNGP grid search yields comparable patterns: conjugate models are competitive but do not dramatically outperform the standard spNNGP in the single-year setting, with similar residual behavior across covariance structures.

Extending the analysis to a multi-year setting substantially improves model fit. When Year is modeled jointly with the other predictors in multi-year spNNGP and spConjNNGP fits, RMSE drops (roughly into the 0.20–0.33 range for the best matern and spherical models) and adjusted R^2 increases markedly, indicating that sharing information over time helps explain more of the spatial and temporal variation in diabetes. RMSE-by-year plots and residual boxplots show relatively stable errors across 2017–2021, with residuals centered near zero and only mild year-to-year shifts, suggesting no major temporal breakdown of model performance.

The final comparative analysis across all NNGP variants shows that the main gains come from using multi-year models rather than from choosing between spNNGP and spConjNNGP. Multi-year spConjNNGP and spNNGP, especially with matern or spherical covariance, give the best overall trade-off between low prediction error and high explained variation, while keeping residuals well-behaved across space and time. Taken together, the study supports the view that diabetes prevalence in Texas is strongly spatially structured, closely tied to obesity, poverty, insurance coverage, and urbanicity, and is best modeled using spatial NNGP approaches that exploit both spatial neighbors and multi-year information.