Cálculo de Largura de Banda para GWR

Seu Nome/Projeto

2025-06-17

Contents

```
# 1. Carregar bibliotecas necessárias
message("Carregando bibliotecas essenciais para análise espacial e GWR...")
```

```
## Carregando bibliotecas essenciais para análise espacial e GWR...
```

```
library(sf) # Para manipulação de dados espaciais vetoriais (Simple Features)
```

```
## Linking to GEOS 3.13.1, GDAL 3.10.2, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
library(GWmodel) # Para Geographically Weighted Regression (GWR) e funções relacionadas
```

```
## Carregando pacotes exigidos: robustbase
```

```
## Carregando pacotes exigidos: sp
```

```
## Carregando pacotes exigidos: Rcpp
```

```
## Welcome to GWmodel version 2.4-2.
```

```
library(dplyr) # Para manipulação e transformação de dados
```

```
##
## Anexando pacote: 'dplyr'
```

```
## Os seguintes objetos são mascarados por 'package:stats':
##
##     filter, lag
```

```
## Os seguintes objetos são mascarados por 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2) # Para visualização e criação de mapas
library(sp) # Necessário para o pacote GWmodel, que utiliza objetos Spatial*DataFrame
library(spdep) # Para testes de autocorrelação espacial como o I de Moran <--- ADICIONADO
```

```
## Carregando pacotes exigidos: spData
```

```
## To access larger datasets in this package, install the spDataLarge
## package with: 'install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')'
```

```r
message("Bibliotecas carregadas com sucesso.")
```

```
## Bibliotecas carregadas com sucesso.
```

```r
# --- INÍCIO: Definição dos caminhos dos arquivos ---
# É uma boa prática centralizar a definição de caminhos para facilitar a manutenção
# e a portabilidade do script. Considere usar 'here::here()' para caminhos relativos
# ao projeto, o que torna o script mais robusto em diferentes ambientes.
# Exemplo: path_base <- here::here("data", "shp")
# path_roubos <- file.path(path_base, "roubo.shp")
# path_drogas <- file.path(path_base, "drogas.shp")
# path_area_estudo <- file.path(path_base, "centro_expandido", "centro_expandido_dissolve.shp")

path_roubos <- "C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda/Documentos/mba/r
path_drogas <- "C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda/Documentos/mba/r
path_area_estudo <- "C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda/Documentos/
# --- FIM: Definição dos caminhos dos arquivos ---

# 2. Ler os arquivos shapefile
message("Iniciando a leitura dos arquivos shapefile...")
```

```
## Iniciando a leitura dos arquivos shapefile...
```

```r
tryCatch({
  # st_read com quiet = TRUE suprime mensagens de progresso, mantendo o output limpo.
  pontos_roubo <- st_read(path_roubos, quiet = TRUE)
  pontos_drogas <- st_read(path_drogas, quiet = TRUE)
  area_estudo <- st_read(path_area_estudo, quiet = TRUE)
}, error = function(e) {
  # Em caso de erro na leitura (e.g., arquivo não encontrado, corrompido),
  # o script é interrompido com uma mensagem clara.
  stop("Erro crítico ao ler um ou mais arquivos shapefile. Verifique os caminhos e a integridade dos ar
})
message("Arquivos shapefile carregados com sucesso.")
```

```
## Arquivos shapefile carregados com sucesso.
```

```r
# 3. Definir CRS projetado alvo e transformar camadas
# Para análises de distância e área, como GWR e criação de grades, é CRÍTICO usar
# um CRS (Coordinate Reference System) projetado (em metros, quilômetros, etc.),
```

```r
# e não um CRS geográfico (em graus de latitude/longitude).
crs_projetado_epsg <- 31983 # SIRGAS 2000 / UTM zone 23S - um CRS comum para o Brasil.
crs_projetado_desejado <- st_crs(crs_projetado_epsg)
message(paste("CRS projetado desejado para a análise de distância e área: EPSG:", crs_projetado_epsg))
```

```
## CRS projetado desejado para a análise de distância e área: EPSG: 31983
```

```r
# Função auxiliar para transformar CRS de forma segura e com feedback
transformar_crs_se_necessario <- function(sf_object, target_crs_obj, nome_camada) {
  if (st_crs(sf_object) != target_crs_obj) {
    message(paste("Transformando CRS da camada '", nome_camada, "' (EPSG:", st_crs(sf_object)$epsg, ") p
    return(st_transform(sf_object, crs = target_crs_obj))
  } else {
    message(paste("CRS da camada '", nome_camada, "' já é o desejado (EPSG:", st_crs(target_crs_obj)$eps
    return(sf_object)
  }
}

tryCatch({
  area_estudo <- transformar_crs_se_necessario(area_estudo, crs_projetado_desejado, "area_estudo")
  pontos_roubo <- transformar_crs_se_necessario(pontos_roubo, crs_projetado_desejado, "pontos_roubo")
  pontos_drogas <- transformar_crs_se_necessario(pontos_drogas, crs_projetado_desejado, "pontos_drogas")
}, error = function(e) {
  stop("Erro durante a transformação de CRS de uma ou mais camadas: ", e$message)
})
```

```
## Transformando CRS da camada ' area_estudo ' (EPSG: 4326 ) para EPSG: 31983 ...

## Transformando CRS da camada ' pontos_roubo ' (EPSG: 4326 ) para EPSG: 31983 ...

## Transformando CRS da camada ' pontos_drogas ' (EPSG: 4326 ) para EPSG: 31983 ...
```

```r
# Verificação final para garantir que todos os CRS foram harmonizados
if (st_crs(area_estudo) != crs_projetado_desejado || st_crs(pontos_roubo) != crs_projetado_desejado || s
  stop("Falha crítica ao harmonizar CRS para todas as camadas. Verifique as configurações de CRS.")
} else {
  message(paste("Todas as camadas foram harmonizadas com sucesso para o CRS projetado (EPSG:", st_crs(ar
}
```

```
## Todas as camadas foram harmonizadas com sucesso para o CRS projetado (EPSG: 31983 ).
```

```r
# Confirmação explícita de que o CRS é projetado para a criação da grade
if (st_is_longlat(area_estudo)) {
    stop("O CRS da área de estudo ainda é geográfico (latitude/longitude). A criação da grade e a análi
} else {
    message(paste("CRS para criação da grade é projetado (EPSG:", st_crs(area_estudo)$epsg, "), adequado
}
```

```
## CRS para criação da grade é projetado (EPSG: 31983 ), adequado para cálculos de distância e área.
```

```r
# --- Função Auxiliar para Impressão Robusta de Diagnósticos ---
# Verifica se o valor é numérico e finito antes de imprimir
print_diagnostic_robust <- function(value, name, description) {
    if (!is.null(value) && is.numeric(value) && is.finite(value)) {
        cat(paste0(name, ": ", round(value, 4), "\n")) # Aumenta precisão para AIC/R2/etc.
        cat(paste0(" -> ", description, "\n"))
    } else {
        # Tenta extrair o valor mesmo que não seja numérico/finito para mostrar o que encontrou
        val_str <- if (is.null(value)) "NULL" else as.character(value)
        cat(paste0(name, ": Não disponível ou não numérico/finito (Valor: ", val_str, ").\n"))
        cat(paste0(" -> ", description, " (Não calculado/Reportado)\n"))
    }
}

# Nota: Certifique-se de que o pacote 'e1071' está instalado para Assimetria e Curtose dos Resíduos
# install.packages('e1071') # Rode isto no console R se não tiver instalado
# library(e1071) # Carregue o pacote no início do seu script
```

```r
# 4. Criar uma grade hexagonal de análise sobre a área de estudo
cell_size_m <- 800
message(paste("Criando grade hexagonal de análise com 'side length' de", cell_size_m, "metros..."))
```

```
## Criando grade hexagonal de análise com 'side length' de 800 metros...
```

```r
area_estudo_union <- st_union(area_estudo)
grade_base <- st_make_grid(area_estudo_union, cellsize = cell_size_m, what = "polygons", square = FALSE)
grade_sf_obj <- st_sf(id_celula_grade_temp = 1:length(grade_base), geometry = grade_base)
grade_area_estudo_raw <- st_intersection(grade_sf_obj, area_estudo_union)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

```r
grade_area_estudo <- grade_area_estudo_raw[!st_is_empty(grade_area_estudo_raw), ]
grade_area_estudo <- grade_area_estudo[st_is_valid(grade_area_estudo$geometry), ]

if(nrow(grade_area_estudo) == 0) {
    stop("Nenhuma célula da grade intersecta a área de estudo ou todas as geometrias resultantes são inv
}
grade_area_estudo$id_celula_grade <- 1:nrow(grade_area_estudo)
grade_area_estudo <- grade_area_estudo %>% dplyr::select(id_celula_grade, geometry)
message(paste("Grade hexagonal de análise criada com", nrow(grade_area_estudo), "células válidas dentro
```

```
## Grade hexagonal de análise criada com 406 células válidas dentro da área de estudo.
```

```r
# Bloco 5: Agregar os pontos à grade e aplicar transformação log1p
message("Iniciando a agregação dos pontos de roubo e drogas às células da grade...")
```

```
## Iniciando a agregação dos pontos de roubo e drogas às células da grade...
```

```r
agregar_pontos_a_grade <- function(pontos_sf, grade_sf, nome_variavel_contagem) {
  pontos_na_grade <- suppressMessages(st_join(pontos_sf, grade_sf, join = st_intersects))
  contagem <- pontos_na_grade %>%
    st_drop_geometry() %>%
    filter(!is.na(id_celula_grade)) %>%
    group_by(id_celula_grade) %>%
    summarise(!!sym(nome_variavel_contagem) := n(), .groups = 'drop')
  return(contagem)
}

contagem_roubos <- agregar_pontos_a_grade(pontos_roubo, grade_area_estudo, "n_roubos")
contagem_drogas <- agregar_pontos_a_grade(pontos_drogas, grade_area_estudo, "n_drogas")

dados_gwr_sf <- grade_area_estudo %>%
  left_join(contagem_roubos, by = "id_celula_grade") %>%
  left_join(contagem_drogas, by = "id_celula_grade")

dados_gwr_sf$n_roubos[is.na(dados_gwr_sf$n_roubos)] <- 0
dados_gwr_sf$n_drogas[is.na(dados_gwr_sf$n_drogas)] <- 0
message("Dados de roubos e drogas agregados às células da grade.")
```

```
## Dados de roubos e drogas agregados às células da grade.
```

```r
message("Aplicando transformação log1p às variáveis de contagem (n_roubos, n_drogas)...")
```

```
## Aplicando transformação log1p às variáveis de contagem (n_roubos, n_drogas)...
```

```r
dados_gwr_sf <- dados_gwr_sf %>%
  mutate(
    log1p_n_roubos = log1p(n_roubos),
    log1p_n_drogas = log1p(n_drogas)
  )
message("Transformação log1p aplicada. Novas colunas criadas: 'log1p_n_roubos' e 'log1p_n_drogas'.")
```

```
## Transformação log1p aplicada. Novas colunas criadas: 'log1p_n_roubos' e 'log1p_n_drogas'.
```

```r
formula_gwr <- log1p_n_roubos ~ log1p_n_drogas
message(paste("Fórmula GWR definida como:", deparse(formula_gwr)))
```

```
## Fórmula GWR definida como: log1p_n_roubos ~ log1p_n_drogas
```

```r
# 6. Preparar dados para GWR
message("Preparando dados para GWR: convertendo para formato SpatialPointsDataFrame (SPDF)...")
```

```
## Preparando dados para GWR: convertendo para formato SpatialPointsDataFrame (SPDF)...
```

```r
dados_gwr_sf_validos <- NULL
dados_spdf_gwr <- NULL

tryCatch({
```

```
  dados_gwr_sf_validos <- dados_gwr_sf
  if(nrow(dados_gwr_sf_validos) == 0) stop("Nenhuma célula com geometria válida após a agregação. Impos

  message("Calculando pontos representativos (st_point_on_surface) para as células da grade...")
  locais_regressao_sf_centroids <- st_point_on_surface(dados_gwr_sf_validos)
  coords_locais_regressao <- st_coordinates(locais_regressao_sf_centroids)

  if(any(is.na(coords_locais_regressao))) {
      stop("Coordenadas NA encontradas após st_point_on_surface. Isso pode indicar geometrias problemáti
  }

  message("Extraindo atributos (incluindo colunas transformadas log1p) para o SPDF...")
  dados_atributos_para_spdf <- st_drop_geometry(dados_gwr_sf_validos)

  dados_spdf_gwr <- SpatialPointsDataFrame(coords = coords_locais_regressao,
                                           data = dados_atributos_para_spdf,
                                           proj4string = CRS(st_crs(dados_gwr_sf_validos)$proj4string))

  message(paste("Dados SPDF preparados com sucesso. Número de Pontos SPDF para GWR:", length(dados_spdf
  message(paste("Nomes das colunas de atributos em dados_spdf_gwr@data:", paste(names(dados_spdf_gwr@da

  if(!("log1p_n_roubos" %in% names(dados_spdf_gwr@data) && "log1p_n_drogas" %in% names(dados_spdf_gwr@d
      warning("AVISO: Colunas 'log1p_n_roubos' ou 'log1p_n_drogas' não encontradas em dados_spdf_gwr@da
  } else {
      message("Confirmação: Colunas 'log1p_n_roubos' e 'log1p_n_drogas' estão presentes em dados_spdf_g
  }
}, error = function(e) {
  stop(paste("Erro ao preparar dados para GWR (conversão para SPDF):", e$message))
})
```

```
## Calculando pontos representativos (st_point_on_surface) para as células da grade...

## Warning: st_point_on_surface assumes attributes are constant over geometries

## Extraindo atributos (incluindo colunas transformadas log1p) para o SPDF...

## Dados SPDF preparados com sucesso. Número de Pontos SPDF para GWR: 406

## Nomes das colunas de atributos em dados_spdf_gwr@data: id_celula_grade, n_roubos, n_drogas, log1p_n_
```

```
## Confirmação: Colunas 'log1p_n_roubos' e 'log1p_n_drogas' estão presentes em dados_spdf_gwr@data.
```

```
# --- INÍCIO: Inspeção da Esparsidade dos Dados ---
message("------------------------------------------------------")
```

```
## ------------------------------------------------------
```

```
message("Inspeção da Esparsidade dos Dados para GWR:")
```

```
## Inspeção da Esparsidade dos Dados para GWR:
```

```r
message("------------------------------------------------------")
```

```
## ------------------------------------------------------
```

```r
if (!is.null(dados_spdf_gwr) && nrow(dados_spdf_gwr@data) > 0) {
    cat("Resumo estatístico para 'n_roubos' (variável dependente):\n")
    print(summary(dados_spdf_gwr$n_roubos))
    cat("\nFrequência de células com e sem ocorrências de roubos:\n")
    print(table(Com_Roubos = dados_spdf_gwr$n_roubos > 0))
    message(paste("\nPorcentagem de células SEM roubos:",
              round(sum(dados_spdf_gwr$n_roubos == 0) / length(dados_spdf_gwr$n_roubos) * 100, 2), "%"))

    cat("\nResumo estatístico para 'n_drogas' (variável independente):\n")
    print(summary(dados_spdf_gwr$n_drogas))
    cat("\nFrequência de células com e sem ocorrências de drogas:\n")
    print(table(Com_Drogas = dados_spdf_gwr$n_drogas > 0))
    message(paste("\nPorcentagem de células SEM drogas:",
              round(sum(dados_spdf_gwr$n_drogas == 0) / length(dados_spdf_gwr$n_drogas) * 100, 2), "%"))

    message(paste("\nNúmero de células com n_roubos = 0 E n_drogas = 0:",
              sum(dados_spdf_gwr$n_roubos == 0 & dados_spdf_gwr$n_drogas == 0)))
    message(paste("Porcentagem de células com n_roubos = 0 E n_drogas = 0:",
              round(sum(dados_spdf_gwr$n_roubos == 0 & dados_spdf_gwr$n_drogas == 0) / length(dados_spdf

    message(paste("\nTotal de células (pontos de regressão) na análise GWR:", length(dados_spdf_gwr)))
    if (length(dados_spdf_gwr) < 50) {
        warning("O número de células para a análise GWR é baixo (< 50). Isso pode afetar a estabilidade
    }
} else {
    warning("dados_spdf_gwr está vazio ou nulo. Não foi possível inspecionar a esparsidade dos dados.")
}
```

```
## Resumo estatístico para 'n_roubos' (variável dependente):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   4.335   3.000 177.000
##
## Frequência de células com e sem ocorrências de roubos:
## Com_Roubos
## FALSE  TRUE
##   159   247

##
## Porcentagem de células SEM roubos: 39.16 %

##
## Resumo estatístico para 'n_drogas' (variável independente):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    25.0    60.0   128.5   129.5  1552.0
##
## Frequência de células com e sem ocorrências de drogas:
## Com_Drogas
## FALSE  TRUE
##    19   387
```

```
##
## Porcentagem de células SEM drogas: 4.68 %

##
## Número de células com n_roubos = 0 E n_drogas = 0: 16

## Porcentagem de células com n_roubos = 0 E n_drogas = 0: 3.94 %

##
## Total de células (pontos de regressão) na análise GWR: 406
```

```
message("------------------------------------------------------\n")
```

```
## ------------------------------------------------------
```

```
# --- FIM: Inspeção da Esparsidade dos Dados ---

# 7. Calcular a largura de banda ótima e matriz de distância
message(paste("Fórmula GWR definida como:", deparse(formula_gwr)))
```

```
## Fórmula GWR definida como: log1p_n_roubos ~ log1p_n_drogas
```

```
if(is.null(dados_spdf_gwr) || length(dados_spdf_gwr) < 30) { # Mínimo sugerido para GWR robusto
    stop(paste("Número insuficiente de pontos (<30) para GWR ou dados_spdf_gwr é nulo. Atualmente:",
               ifelse(is.null(dados_spdf_gwr), 0, length(dados_spdf_gwr)),
               "\nConsidere aumentar 'cell_size_m' ou verificar a área de estudo e a agregação de dados
}

vars_na_formula <- all.vars(formula_gwr)
if(!all(vars_na_formula %in% names(dados_spdf_gwr@data))){
    colunas_faltantes <- setdiff(vars_na_formula, names(dados_spdf_gwr@data))
    stop(paste("ERRO CRÍTICO: Variáveis da fórmula ('", paste(colunas_faltantes, collapse=", "), "') nã
} else {
    message(paste("Confirmação: Todas as variáveis da fórmula ('", paste(vars_na_formula, collapse=", "
}
```

```
## Confirmação: Todas as variáveis da fórmula (' log1p_n_roubos, log1p_n_drogas ') estão presentes em 'd
```

```
# --- INÍCIO DA CORREÇÃO: Calcular a matriz de distância ---
message("Calculando a matriz de distância para a seleção da largura de banda...")
```

```
## Calculando a matriz de distância para a seleção da largura de banda...
```

```
dMat_calibracao <- NULL
if (is.null(dados_spdf_gwr) || nrow(dados_spdf_gwr) == 0) {
    stop("dados_spdf_gwr está vazio ou nulo. Não é possível calcular a matriz de distância.")
}
coords_para_dmat <- coordinates(dados_spdf_gwr)
if(any(is.na(coords_para_dmat))) {
    stop("Coordenadas NA encontradas em dados_spdf_gwr. Impossível calcular a matriz de distância.")
```

```
}

# Verifica se o CRS é geográfico (longlat=TRUE) ou projetado (longlat=FALSE)
# O CRS foi definido como projetado no Bloco 3.
crs_spdf <- st_crs(dados_spdf_gwr)
is_longlat_check <- FALSE # Default para projetado
if (!is.na(crs_spdf) && !is.null(crs_spdf)) {
    is_longlat_check <- st_is_longlat(crs_spdf)
    if (is.na(is_longlat_check)) { # Se st_is_longlat retorna NA
        warning("Não foi possível determinar programaticamente se o CRS é longlat (st_is_longlat retorne
        is_longlat_check <- FALSE
    }
} else {
    warning("CRS de dados_spdf_gwr é NA ou NULL. Assumindo projetado (longlat=FALSE). Verifique a prepa
    is_longlat_check <- FALSE
}
message(paste("Para o cálculo da matriz de distância: longlat =", is_longlat_check))
```

```
## Para o cálculo da matriz de distância: longlat = FALSE
```

```
tryCatch({
  dMat_calibracao <- GWmodel::gw.dist(
    dp.locat = coords_para_dmat,
    # rp.locat = coords_para_dmat, # Não é necessário para bw.gwr/bw.ggwr se dp.locat é usado para cálc
    p = 2, # Distância Euclidiana
    theta = 0, # Nenhuma rotação
    longlat = is_longlat_check
  )
  message("Matriz de distância 'dMat_calibracao' calculada com sucesso.")
  message(paste("Dimensões da dMat_calibracao:", paste(dim(dMat_calibracao), collapse = "x")))
}, error = function(e) {
  stop(paste("Erro ao calcular a matriz de distância 'dMat_calibracao' com gw.dist:", e$message))
})
```

```
## Matriz de distância 'dMat_calibracao' calculada com sucesso.

## Dimensões da dMat_calibracao: 406x406
```

```
if (is.null(dMat_calibracao) || !is.matrix(dMat_calibracao) || nrow(dMat_calibracao) != length(dados_sp
    stop("A matriz de distância 'dMat_calibracao' não foi criada corretamente ou tem dimensões inválidas
}
# --- FIM DA CORREÇÃO ---

message("Iniciando a busca pela largura de banda adaptativa ótima...")
```

```
## Iniciando a busca pela largura de banda adaptativa ótima...
```

```
bw_adaptativa <- NULL
error_message_bw <- ""
kernel_usado_para_bw <- "gaussian" # Default
```

```r
tentativas_bw_gwr <- list(
  list(approach = "AIC", kernel = "gaussian"),
  list(approach = "CV", kernel = "gaussian"),
  list(approach = "AIC", kernel = "bisquare"),
  list(approach = "CV", kernel = "bisquare")
)

tentativas_bw_ggwr <- list(
  list(approach = "AICc", kernel = "gaussian"),
  list(approach = "CV", kernel = "gaussian"),
  list(approach = "AICc", kernel = "bisquare"),
  list(approach = "CV", kernel = "bisquare")
)

funcao_bw_usada <- ""

message("\n--- Priorizando GWmodel::bw.gwr (específico para GWR Gaussiano) ---")
```

```
##
## --- Priorizando GWmodel::bw.gwr (específico para GWR Gaussiano) ---
```

```r
for (i in 1:length(tentativas_bw_gwr)) {
  params <- tentativas_bw_gwr[[i]]
  message(paste0("\n--- Tentativa com bw.gwr ", i, ": approach = '", params$approach, "', kernel = '", p
  tryCatch({
    bw_adaptativa_temp <- GWmodel::bw.gwr(
      formula = formula_gwr,
      data = dados_spdf_gwr,
      dMat = dMat_calibracao, # AGORA dMat_calibracao EXISTE
      approach = params$approach,
      kernel = params$kernel,
      adaptive = TRUE
    )
    if (!is.null(bw_adaptativa_temp) && is.numeric(bw_adaptativa_temp) && bw_adaptativa_temp > 0 && bw_a
      bw_adaptativa <- bw_adaptativa_temp
      kernel_usado_para_bw <- params$kernel
      funcao_bw_usada <- "bw.gwr"
      message(paste("Sucesso com bw.gwr na Tentativa ", i, "! Largura de banda:", round(bw_adaptativa, 
      break
    } else {
      msg <- paste0("bw.gwr Tentativa ", i, " (", params$approach, ", ", params$kernel, ") não retornou
      message(msg)
      error_message_bw <- paste0(error_message_bw, msg, "; ")
    }
  }, error = function(e) {
    message(paste("ERRO com bw.gwr na Tentativa ", i, " (", params$approach, ", ", params$kernel, "):",
    error_message_bw <- paste0(error_message_bw, "Erro bw.gwr (", params$approach, ",", params$kernel, 
  })
  if (!is.null(bw_adaptativa)) break
}
```

```
##
## --- Tentativa com bw.gwr 1: approach = 'AIC', kernel = 'gaussian' ---
```

```
## Adaptive bandwidth (number of nearest neighbours): 258 AICc value: 1027.573
## Adaptive bandwidth (number of nearest neighbours): 167 AICc value: 1013.533
## Adaptive bandwidth (number of nearest neighbours): 110 AICc value: 995.649
## Adaptive bandwidth (number of nearest neighbours): 75 AICc value: 977.411
## Adaptive bandwidth (number of nearest neighbours): 53 AICc value: 957.1655
## Adaptive bandwidth (number of nearest neighbours): 40 AICc value: 950.0465
## Adaptive bandwidth (number of nearest neighbours): 31 AICc value: 929.7529
## Adaptive bandwidth (number of nearest neighbours): 26 AICc value: 929.3024
## Adaptive bandwidth (number of nearest neighbours): 22 AICc value: 928.6423
## Adaptive bandwidth (number of nearest neighbours): 20 AICc value: 924.5499
## Adaptive bandwidth (number of nearest neighbours): 18 AICc value: 911.9409
## Adaptive bandwidth (number of nearest neighbours): 18 AICc value: 911.9409

## Sucesso com bw.gwr na Tentativa  1 ! Largura de banda: 18 vizinhos, kernel: gaussian
```

```r
if (is.null(bw_adaptativa)) {
    message("\n--- bw.gwr falhou ou não encontrou largura de banda. Tentando GWmodel::bw.ggwr (Gaussian
    for (i in 1:length(tentativas_bw_ggwr)) {
      params <- tentativas_bw_ggwr[[i]]
      message(paste0("\n--- Tentativa com bw.ggwr ", i, ": approach = '", params$approach, "', kernel =
      tryCatch({
        bw_adaptativa_temp <- GWmodel::bw.ggwr(
          formula = formula_gwr,
          data = dados_spdf_gwr,
          dMat = dMat_calibracao, # AGORA dMat_calibracao EXISTE
          approach = params$approach,
          kernel = params$kernel,
          adaptive = TRUE
        )
        if (!is.null(bw_adaptativa_temp) && is.numeric(bw_adaptativa_temp) && bw_adaptativa_temp > 0 &&
          bw_adaptativa <- bw_adaptativa_temp
          kernel_usado_para_bw <- params$kernel
          funcao_bw_usada <- "bw.ggwr"
          message(paste("Sucesso com bw.ggwr na Tentativa ", i, "! Largura de banda:", round(bw_adaptati
          break
        } else {
          msg <- paste0("bw.ggwr Tentativa ", i, " (", params$approach, ", ", params$kernel, ") não reto
          message(msg)
          error_message_bw <- paste0(error_message_bw, msg, "; ")
        }
      }, error = function(e) {
        message(paste("ERRO com bw.ggwr na Tentativa ", i, " (", params$approach, ", ", params$kernel,
        if (grepl("the condition has length > 1", e$message, fixed = TRUE)) {
            message("AVISO: O erro 'the condition has length > 1' persistiu mesmo sem especificar 'famil
        }
        error_message_bw <- paste0(error_message_bw, "Erro bw.ggwr (", params$approach, ",", params$kern
      })
      if (!is.null(bw_adaptativa)) break
    }
}

if (is.null(bw_adaptativa)) {
  cat("--------------------------------------------------------------------------------\n")
  cat("ATENÇÃO: Não foi possível calcular automaticamente a largura de banda ótima (bw_adaptativa)\n")
```

```
  cat("após todas as tentativas com bw.gwr e bw.ggwr (para modelo Gaussiano).\n")
  cat("Mensagens de erro/aviso acumuladas das tentativas:\n", error_message_bw, "\n\n")
  cat("Este problema frequentemente ocorre devido a:\n")
  cat("  a) Esparsidade dos dados (mesmo após transformação log1p, muitos valores 'log1p(0)=0' podem ser
  cat("  b) Colinearidade local: Variáveis independentes fortemente correlacionadas em algumas vizinhanç
  cat("  c) Número insuficiente de observações (células) para o GWR, especialmente se a largura de banda
  cat("  d) Se o erro 'valor ausente onde TRUE/FALSE necessário' ou 'the condition has length > 1' persi

  cat("Próximas Etapas Sugeridas para Solução:\n")
  cat("1. REVISE CUIDADOSAMENTE O OUTPUT DA 'Inspeção da Esparsidade dos Dados' e as mensagens de erro d
  cat("2. TENTAR UMA LARGURA DE BANDA ADAPTATIVA MANUALMENTE MAIOR (PARA DIAGNÓSTICO):\n")
  cat("#   bw_adaptativa <- round(0.1 * nrow(dados_spdf_gwr@data)) # Exemplo: 10% dos pontos\n")
  cat('#   kernel_usado_para_bw <- "gaussian" \n')
  cat('#   funcao_bw_usada <- "manual"\n')
  cat('#   message(paste("Usando largura de banda adaptativa definida manualmente:", bw_adaptativa, "viz
  cat("3. AUMENTAR O TAMANHO DA CÉLULA DA GRADE ('cell_size_m').\n")
  cat("4. RECONSIDERAR AS VARIÁVEIS DO MODELO OU A TRANSFORMAÇÃO.\n")
  cat("5. VERIFICAR A VERSÃO DO PACOTE GWmodel E ATUALIZAR SE POSSÍVEL.\n\n")
  cat("-------------------------------------------------------------------------------------------\n")
  stop("Cálculo automático da largura de banda falhou. Analise as mensagens e sugestões acima para depur
} else {
  message(paste("\nLargura de banda adaptativa ótima calculada com sucesso usando", funcao_bw_usada, ":
              "vizinhos, utilizando o kernel:", kernel_usado_para_bw, "."))
  bw_adaptativa <- round(bw_adaptativa, 0) # Garante que seja um inteiro
  if (bw_adaptativa < 2) { # Uma largura de banda de 1 vizinho não faz sentido
      warning(paste0("A largura de banda calculada (", bw_adaptativa, ") é muito pequena. Pode indicar p
      # Exemplo: bw_adaptativa <- max(bw_adaptativa, 20)
  }
  message(paste("Largura de banda final (arredondada para o inteiro mais próximo):", bw_adaptativa, "viz
}
```

```
##
## Largura de banda adaptativa ótima calculada com sucesso usando bw.gwr : 18 vizinhos, utilizando o ker

## Largura de banda final (arredondada para o inteiro mais próximo): 18 vizinhos.
```

```
message(paste("\nCONFIRMAÇÃO ANTES DO GWR: Usando largura de banda adaptativa (número de vizinhos):", bw
              "e kernel:", kernel_usado_para_bw, "para gwr.basic (modelo Gaussiano implícito)."))
```

```
##
## CONFIRMAÇÃO ANTES DO GWR: Usando largura de banda adaptativa (número de vizinhos): 18 e kernel: gauss
```

```
if (is.null(bw_adaptativa) || !is.numeric(bw_adaptativa) || bw_adaptativa <= 0) {
    stop("ERRO CRÍTICO: 'bw_adaptativa' é inválida (NULL, não numérica ou não positiva) antes de iniciar
}
if (is.null(kernel_usado_para_bw) || !kernel_usado_para_bw %in% c("gaussian", "bisquare", "tricube", "bo
    stop(paste("ERRO CRÍTICO: 'kernel_usado_para_bw' ('", kernel_usado_para_bw, "') é inválido para gwr
}
```

```
# 8. Executar o GWR usando gwr.basic
message("Executando a Regressão Geograficamente Ponderada (GWR) com gwr.basic (modelo Gaussiano implíci
```

```
## Executando a Regressão Geograficamente Ponderada (GWR) com gwr.basic (modelo Gaussiano implícito)...
```

```
gwr_resultado_lista <- NULL

tryCatch({
  gwr_resultado_lista <- GWmodel::gwr.basic(
    formula = formula_gwr,
    data = dados_spdf_gwr,
    bw = bw_adaptativa,
    kernel = kernel_usado_para_bw,
    adaptive = TRUE,
    dMat = dMat_calibracao
  )
}, error = function(e) {
  cat("Erro crítico ao executar gwr.basic (modelo Gaussiano implícito):\n", e$message, "\n")

  if (grepl("the condition has length > 1", e$message, fixed = TRUE)) {
    message("AVISO: O erro 'the condition has length > 1' persistiu em gwr.basic mesmo sem especificar
  }
  stop(paste("Falha ao executar gwr.basic (modelo Gaussiano implícito). Verifique a largura de banda, o
             "Causas comuns (além de bugs no pacote) incluem: singularidade devido a poucos vizinhos, da
})

if (is.null(gwr_resultado_lista) || is.null(gwr_resultado_lista$SDF)) {
  stop("Falha ao executar gwr.basic (modelo Gaussiano implícito) ou o resultado não contém o componente
} else {
  message("Análise GWR (gwr.basic, modelo Gaussiano implícito) concluída com sucesso. Processando os res
}
```

```
## Análise GWR (gwr.basic, modelo Gaussiano implícito) concluída com sucesso. Processando os resultados
```

```
gwr_resultados_sdf <- gwr_resultado_lista$SDF
message(paste("Resultados GWR (SDF) obtidos. Nomes das colunas no SDF do GWR:", paste(names(gwr_resultad
```

```
## Resultados GWR (SDF) obtidos. Nomes das colunas no SDF do GWR: Intercept, log1p_n_drogas, y, yhat, re
```

```
# Verificar se a variável dependente no SDF corresponde à VD da fórmula
var_dependente_formula <- all.vars(formula_gwr)[1]
if ("y" %in% names(gwr_resultados_sdf)) {
  if (all(gwr_resultados_sdf$y == dados_spdf_gwr@data[[var_dependente_formula]], na.rm = TRUE)) {
    message(paste0("Confirmação: A coluna 'y' no SDF do GWR corresponde aos valores de '", var_dependent
  } else {
    warning(paste0("AVISO: A coluna 'y' no SDF do GWR pode não corresponder exatamente aos valores de '
  }
} else {
  warning("AVISO: Coluna 'y' (variável dependente) não encontrada no SDF do GWR.")
}
```

```
## Confirmação: A coluna 'y' no SDF do GWR corresponde aos valores de 'log1p_n_roubos' dos dados de entr
```

```r
# Verificar se a coluna da variável independente principal da fórmula existe nos resultados
var_independente_principal_formula <- all.vars(formula_gwr)[2]
if (!(var_independente_principal_formula %in% names(gwr_resultados_sdf))) {
  warning(paste0("AVISO: A coluna do coeficiente para '", var_independente_principal_formula, "' não fo
}


if(nrow(dados_gwr_sf_validos) == nrow(gwr_resultados_sdf)) {
    # Extrair os dados do SDF do GWR para um data.frame
    gwr_resultados_df_para_join <- as.data.frame(gwr_resultados_sdf)

    # Adicionar 'id_celula_grade' de 'dados_gwr_sf_validos' aos resultados do GWR
    if ("id_celula_grade" %in% names(dados_gwr_sf_validos)) {
        gwr_resultados_df_para_join$id_celula_grade <- dados_gwr_sf_validos$id_celula_grade
        message("Coluna 'id_celula_grade' adicionada aos resultados do GWR a partir de 'dados_gwr_sf_va
    } else {
        stop("ERRO CRÍTICO: 'id_celula_grade' não encontrada em 'dados_gwr_sf_validos'. Impossível adic
    }

    # Verificar se 'id_celula_grade' está presente
    if (!"id_celula_grade" %in% names(gwr_resultados_df_para_join)) {
        stop("Falha inesperada ao adicionar 'id_celula_grade' a 'gwr_resultados_df_para_join'. Verifique
    }

    # Renomear a coluna do coeficiente 'n_drogas' para 'coef_n_drogas'
    if ("n_drogas" %in% names(gwr_resultados_df_para_join)) {
        names(gwr_resultados_df_para_join)[names(gwr_resultados_df_para_join) == "n_drogas"] <- "coef_n_
        message("Coluna do coeficiente para 'n_drogas' nos resultados GWR renomeada para 'coef_n_drogas
    } else {
        warning("Coluna do coeficiente para 'n_drogas' não encontrada diretamente nos resultados do GWR
    }

    # Definir as colunas de resultados do GWR que serão mantidas
    cols_gwr_a_manter <- c("id_celula_grade",
                           "Intercept", "Intercept_TV",
                           "coef_n_drogas", "n_drogas_TV",
                           "Local_R2", "y", "yhat", "residual")

    # Filtrar colunas para o join
    cols_existentes_no_join_df <- intersect(cols_gwr_a_manter, names(gwr_resultados_df_para_join))
    gwr_resultados_filtrados_df <- gwr_resultados_df_para_join %>%
                                   dplyr::select(dplyr::all_of(cols_existentes_no_join_df))

    # Realizar o join dos resultados do GWR com o objeto sf dos polígonos da grade
    resultados_gwr_poligonos_sf <- dplyr::left_join(dados_gwr_sf_validos,
                                                    gwr_resultados_filtrados_df,
                                                    by = "id_celula_grade")

    message("Resultados GWR combinados com os polígonos originais da grade usando 'id_celula_grade'.")

    # Verificação pós-join
    if (any(is.na(resultados_gwr_poligonos_sf$Local_R2))) {
        warning("NAs encontrados em 'Local_R2' após o join. Isso pode indicar problemas na correspondên
```

```
    }

} else {
    stop(paste("Número de linhas nos resultados GWR (", nrow(gwr_resultados_sdf),
               ") não corresponde aos dados de polígonos originais (", nrow(dados_gwr_sf_validos),
               "). Não é possível combinar os resultados de forma segura. Verifique a integridade dos da
}
```

```
## Coluna 'id_celula_grade' adicionada aos resultados do GWR a partir de 'dados_gwr_sf_validos'.

## Warning: Coluna do coeficiente para 'n_drogas' não encontrada diretamente nos
## resultados do GWR. Verifique os nomes das variáveis no modelo GWR e o output do
## GWmodel.

## Resultados GWR combinados com os polígonos originais da grade usando 'id_celula_grade'.
```

```
# --- Bloco 9: Combinar Resultados GWR aos Polígonos e Gerar Mapas Essenciais (SOMENTE RESULTADOS GWR)
# Combina os resultados tabulares do GWR (coeficientes, R2, etc.) de volta ao objeto espacial dos políg
# e gera os mapas essenciais para visualização e análise, FOCANDO APENAS NOS RESULTADOS DO MODELO GWR.
cat("\n\n--- BL. 9: COMBINANDO RESULTADOS GWR E GERANDO MAPAS ESSENCIAIS (SOMENTE RESULTADOS GWR) ---")
```

```
##
##
## --- BL. 9: COMBINANDO RESULTADOS GWR E GERANDO MAPAS ESSENCIAIS (SOMENTE RESULTADOS GWR) ---
```

```
cat("\n----------------------------------------------------------------------------------------------\n")
```

```
##
## ----------------------------------------------------------------------------------------------
```

```
# --- 9.1. Verificar objetos necessários ---
# Verifica se os principais objetos gerados nos blocos anteriores existem, não são NULL e não estão vaz
message("--- 9.1. Verificando objetos necessários ---")
```

```
## --- 9.1. Verificando objetos necessários ---
```

```
if (!exists("dados_gwr_sf_validos") || is.null(dados_gwr_sf_validos) || nrow(dados_gwr_sf_validos) == 0)
    stop("ERRO CRÍTICO Bloco 9: O objeto 'dados_gwr_sf_validos' (polígonos da grade com dados originais)
}
if (!exists("gwr_resultado_lista") || is.null(gwr_resultado_lista) || is.null(gwr_resultado_lista$SDF)
    stop("ERRO CRÍTICO Bloco 9: O objeto 'gwr_resultado_lista' (resultado do GWR) não foi encontrado, é
}

# Verifica se o objeto da área de estudo (contorno para mapas) existe. Emite apenas aviso se não.
if (!exists("area_estudo_union") || is.null(area_estudo_union)) {
     warning("AVISO Bloco 9: Objeto 'area_estudo_union' (contorno da área de estudo) não encontrado. A
     area_estudo_union_mapa <- NULL # Define como NULL para o bloco de plotagem
} else {
    # Se existir, verifica se o CRS coincide com os resultados GWR antes de usá-lo nos mapas
    if (st_crs(area_estudo_union) != st_crs(dados_gwr_sf_validos)) { # Compara com dados_gwr_sf_validos
```

```
        warning("AVISO Bloco 9: CRS da área de estudo difere do CRS dos resultados GWR. A área de estud
        area_estudo_union_mapa <- NULL
    } else {
        area_estudo_union_mapa <- area_estudo_union # Usa o objeto se o CRS for compatível
    }
}

message("Objetos de dados necessários para o Bloco 9 encontrados. Prosseguindo com a combinação de resu
```

```
## Objetos de dados necessários para o Bloco 9 encontrados. Prosseguindo com a combinação de resultados
```

```
# --- 9.2. Preparar Resultados GWR para Join ---
# O SDF do GWR contém os resultados (coeficientes, R2, resíduos) nos centróides.
# Precisamos extrair estes dados (em formato data.frame) para juntá-los aos polígonos SF originais.
message("\n--- 9.2. Extraindo resultados do GWR (SDF) para data.frame ---")
```

```
##
## --- 9.2. Extraindo resultados do GWR (SDF) para data.frame ---
```

```
gwr_resultados_df_para_join <- as.data.frame(gwr_resultado_lista$SDF)

# Adicionar a chave de join ('id_celula_grade') ao data.frame de resultados do GWR.
# Esta coluna DEVE existir em 'dados_gwr_sf_validos' (creada no Bloco 4/5).
# Assumimos que a ordem das linhas no SDF corresponde à ordem em dados_spdf_gwr,
# pois o SDF foi criado a partir de dados_gwr_sf_validos, que por sua vez veio diretamente de dados_gwr
# Embora o join por ID seja mais seguro, garantir a ordem pode ser uma camada extra de verificação se n
# Aqui, confiamos no join por ID após adicionar a coluna.
if ("id_celula_grade" %in% names(dados_gwr_sf_validos)) {
     # Adiciona a coluna id_celula_grade ao dataframe de resultados do GWR
     gwr_resultados_df_para_join$id_celula_grade <- dados_gwr_sf_validos$id_celula_grade
     message("Coluna 'id_celula_grade' adicionada aos resultados do GWR (DF) para join.")
} else {
     stop("ERRO CRÍTICO Bloco 9: A coluna 'id_celula_grade' NÃO foi encontrada no objeto SF 'dados_gwr_s
}
```

```
## Coluna 'id_celula_grade' adicionada aos resultados do GWR (DF) para join.
```

```
# --- 9.3. Renomear Colunas de Resultados GWR para Clareza e Padronização (CORRIGIDO v5) ---
# Renomeia colunas do data.frame extraído do SDF para nomes mais descritivos antes do join.
# Isso evita conflitos de nomes e deixa claro que são resultados do GWR.
message("\n--- 9.3. Renomeando colunas de resultados GWR ---")
```

```
##
## --- 9.3. Renomeando colunas de resultados GWR ---
```

```
# Nome da variável independente principal da fórmula original (e.g. "log1p_n_drogas")
# ESSA VARIÁVEL DEVE TER SIDO DEFINIDA NO SEU SCRIPT ANTES DESTE BLOCO (e.g., no Bloco 5 ou 7)
# Ex: var_indep_na_formula <- all.vars(formula_gwr)[2]
if (!exists("var_indep_na_formula") || is.null(var_indep_na_formula)) {
     # Tenta definir aqui se não foi definida antes, baseando-se na formula_gwr
```

```
     if (exists("formula_gwr") && !is.null(formula_gwr) && length(all.vars(formula_gwr)) > 1) {
          var_indep_na_formula <- all.vars(formula_gwr)[2]
          message(paste0("Variável 'var_indep_na_formula' definida dentro do Bloco 9 como: '", var_indep
     } else {
         stop("ERRO CRÍTICO Bloco 9: Variável 'var_indep_na_formula' não definida e não pôde ser inferi
     }
} else {
    message(paste0("Variável 'var_indep_na_formula' encontrada (valor: '", var_indep_na_formula, "').")
}
```

```
## Variável 'var_indep_na_formula' definida dentro do Bloco 9 como: 'log1p_n_drogas' baseada em formula_
```

```
# Define os nomes originais das colunas de resultados no SDF do GWR (verificar output GWmodel se difere
# Estes são os nomes que esperamos encontrar NO DATA.FRAME gwr_resultados_df_para_join antes de renomea
nome_coef_original_sdf <- var_indep_na_formula # O nome do coeficiente é o nome da variável no modelo
nome_tval_original_sdf <- paste0(var_indep_na_formula, "_TV") # Nome padrão para o t-valor
nome_intercept_original_sdf <- "Intercept" # Nome padrão do intercepto
nome_intercept_tv_original_sdf <- "Intercept_TV" # Nome padrão para o t-valor do intercepto
nome_local_r2_original_sdf <- "Local_R2" # Nome padrão para o R2 Local
nome_yhat_original_sdf <- "yhat" # Nome padrão para o valor previsto
nome_residual_original_sdf <- "residual" # Nome padrão para o resíduo
nome_y_observado_original_sdf <- "y" # Nome padrão para a VD observada na escala transformada


# Define os NOVOS nomes padronizados desejados no data.frame para o join
# *** DEFINIÇÃO EXPLÍCITA DE TODAS AS VARIÁVEIS ANTES DE CRIAR O VETOR DE MAPEAMENTO ***
# Garante que estas variáveis existem no ambiente.
novo_nome_coef <- paste0("GWR_coef_", var_indep_na_formula) # Ex: "GWR_coef_log1p_n_drogas"
novo_nome_tval <- paste0("GWR_tval_", var_indep_na_formula) # Ex: "GWR_tval_log1p_n_drogas"
novo_nome_intercept <- "GWR_coef_Intercept" # Ex: "GWR_coef_Intercept"
novo_nome_intercept_tv <- "GWR_tval_Intercept" # Ex: "GWR_tval_Intercept"
novo_nome_r2_local <- "GWR_Local_R2" # Ex: "GWR_Local_R2"
novo_nome_yhat <- "GWR_yhat" # Ex: "GWR_yhat"
novo_nome_residual <- "GWR_residual" # Ex: "GWR_residual"
novo_nome_y_observado <- "GWR_y_observed_transf" # Nome para a coluna 'y' do SDF, que é a VD observada


# Cria o mapeamento correto para dplyr::rename: c(novo_nome = "nome_original")
# Usa setNames() para criar o vetor nomeado onde os nomes vêm das variáveis.
nomes_originais_esperados <- c(
    nome_coef_original_sdf,
    nome_tval_original_sdf,
    nome_intercept_original_sdf,
    nome_intercept_tv_original_sdf,
    nome_local_r2_original_sdf,
    nome_yhat_original_sdf,
    nome_residual_original_sdf,
    nome_y_observado_original_sdf
)

novos_nomes_desejados <- c(
    novo_nome_coef,
    novo_nome_tval,
```

```
    novo_nome_intercept,
    novo_nome_intercept_tv,
    novo_nome_r2_local,
    novo_nome_yhat,
    novo_nome_residual,
    novo_nome_y_observado
)

# Cria o vetor de mapeamento onde nomes = novos_nomes, valores = nomes_originais_esperados
names_mapping_for_rename <- setNames(nomes_originais_esperados, novos_nomes_desejados)


# Filtra o mapeamento: mantém apenas as entradas onde o NOME ORIGINAL (o valor no vetor names_mapping_f
# realmente existe nos nomes das colunas do data.frame gwr_resultados_df_para_join.
# Isso previne o erro "Can't rename columns that don't exist".
names_to_check_in_df <- as.character(names_mapping_for_rename) # Extrai os nomes originais esperados
existing_original_names <- intersect(names_to_check_in_df, names(gwr_resultados_df_para_join)) # Verifi

# Cria o mapeamento filtrado: apenas entradas cujo nome original existe no DF
# names_mapping_for_rename é um vetor nomeado (nomes=novos nomes, valores=nomes originais)
# Precisamos filtrar baseado nos *valores* que existem em existing_original_names
names_mapping_for_rename_existing <- names_mapping_for_rename[names_mapping_for_rename %in% existing_ori


# Aplica a renomeação usando o mapeamento filtrado.
# A função rename espera c(novo_nome = "nome_original"). Nosso vetor names_mapping_for_rename_existing
# tem nomes = novo_nome e valores = nome_original. O !!! expande corretamente.
if (length(names_mapping_for_rename_existing) > 0) {
    # Renomeia o data.frame. O operador !!! expande o vetor nomeado para argumentos new_name = old_name
    gwr_resultados_df_para_join <- dplyr::rename(gwr_resultados_df_para_join, !!!names_mapping_for_renar

    # Mensagem de confirmação das colunas renomeadas (mostra "nome_original -> novo_nome")
    # Para mostrar "nome_original -> novo_nome": os VALUES do vetor são os nomes originais, os NAMES sã
    rename_message_text <- paste(as.character(names_mapping_for_rename_existing), "->", names(names_mapp
    message(paste0("Colunas de resultados GWR renomeadas no data.frame: ", rename_message_text))
} else {
    message("Nenhuma coluna de resultado GWR padrão encontrada (com os nomes originais esperados) que e
}
```

```
## Colunas de resultados GWR renomeadas no data.frame: log1p_n_drogas -> GWR_coef_log1p_n_drogas, log1p_
```

```
# Verifica AGORA (após a tentativa de renomear) se as colunas com os NOVOS nomes esperados existem.
# Isso confirma quais colunas estarão disponíveis para o join e plotagem.
expected_cols_renamed <- c(novo_nome_coef, novo_nome_tval, novo_nome_r2_local, novo_nome_residual,
                           novo_nome_yhat, novo_nome_intercept, novo_nome_intercept_tv, novo_nome_y_obse
missing_expected_cols_after_rename <- setdiff(expected_cols_renamed, names(gwr_resultados_df_para_join))

if (length(missing_expected_cols_after_rename) > 0) {
     warning(paste0("AVISO Bloco 9: As seguintes colunas de resultados GWR esperadas (APÓS renomeação) f
} else {
    message("Todas as colunas de resultados GWR esperadas encontradas (com os novos nomes) e prontas pa
}
```

```
## Todas as colunas de resultados GWR esperadas encontradas (com os novos nomes) e prontas para join.
```

```r
message("Colunas de resultados GWR preparadas para join com nomes padronizados.")
```

```
## Colunas de resultados GWR preparadas para join com nomes padronizados.
```

```r
# --- 9.4. Realizar o Join Final ao Objeto SF ---
# Junta o data.frame de resultados GWR (agora com nomes padronizados) ao objeto SF dos polígonos origina
# O objeto SF base 'dados_gwr_sf_validos' já contém as geometrias e as variáveis originais/transformada
# Seleciona apenas as colunas de resultados GWR renomeadas (e o id_celula_grade) do data.frame para o j
# para evitar duplicar colunas originais já presentes em 'dados_gwr_sf_validos'.
message("\n--- 9.4. Realizando o join dos resultados GWR ao objeto SF ---")
```

```
##
## --- 9.4. Realizando o join dos resultados GWR ao objeto SF ---
```

```r
# Colunas do DF de resultados GWR a serem mantidas no join (id + colunas renomeadas que existem após re
# Pega todos os nomes no DF de resultados que NÃO são id_celula_grade.
cols_gwr_data_only_to_keep <- setdiff(names(gwr_resultados_df_para_join), "id_celula_grade")

# Realiza o left_join
resultados_gwr_poligonos_sf <- dplyr::left_join(
    dados_gwr_sf_validos, # Objeto SF base com polígonos originais e dados originais
    gwr_resultados_df_para_join %>% dplyr::select(dplyr::all_of(c("id_celula_grade", cols_gwr_data_only
    by = "id_celula_grade" # Chave de join
)

message("Resultados GWR combinados com sucesso aos polígonos originais da grade no objeto SF 'resultados
```

```
## Resultados GWR combinados com sucesso aos polígonos originais da grade no objeto SF 'resultados_gwr_
```

```r
# Verificação rápida pós-join para NAs em uma coluna chave (usando o novo nome esperado)
# Verifica se a coluna existe antes de checar NAs ou emitir aviso.
if (novo_nome_r2_local %in% names(resultados_gwr_poligonos_sf) && any(is.na(resultados_gwr_poligonos_sf
    warning(paste0("AVISO Bloco 9: Foram encontrados NAs na coluna '", novo_nome_r2_local, "' após o jo
} else if (!(novo_nome_r2_local %in% names(resultados_gwr_poligonos_sf))) {
    # AVISO: A coluna R2 Local renomeada nem chegou no objeto final após o join.
     warning(paste0("AVISO Bloco 9: A coluna de resultados GWR esperada '", novo_nome_r2_local, "' não
}


# --- 9.5. Preparar a Área de Estudo para Plotagem (se existir) ---
# Se area_estudo_union foi carregado e tem CRS compatível, ele será usado como contorno nos mapas.
message("\n--- 9.5. Preparando contorno da área de estudo para mapas ---")
```

```
##
## --- 9.5. Preparando contorno da área de estudo para mapas ---
```

```r
# area_estudo_union_mapa já foi definido e verificado no Bloco 9.1


# --- 9.6. Gerar e Salvar Mapas Essenciais (SOMENTE RESULTADOS GWR) ---
message("\n--- 9.6. Iniciando a geração dos mapas essenciais de resultados GWR ---")
```

```
##
## --- 9.6. Iniciando a geração dos mapas essenciais de resultados GWR ---

# Define o diretório para salvar os mapas. Sugestão: usar o mesmo diretório dos shapefiles de entrada.
# Você pode alterar 'path_area_estudo' para outra variável que contenha o caminho base desejado.
output_map_dir <- dirname(path_area_estudo) # Ex: pega o diretório do arquivo de área de estudo
if (!dir.exists(output_map_dir)) {
    warning(paste0("AVISO Bloco 9: Diretório de saída para mapas NÃO EXISTE: '", output_map_dir, "'. Os
    output_map_dir_valid <- FALSE # Flag para não tentar salvar
} else {
     message(paste0("Mapas serão salvos no diretório: '", output_map_dir, "'\n"))
     output_map_dir_valid <- TRUE # Flag para permitir salvar
}

## Mapas serão salvos no diretório: 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing

# ========================================================================
# ADIÇÃO PARA DEBUG: Imprimir variáveis antes da lista map_configs
# Confirma se as variáveis 'novo_nome_...' estão definidas neste ponto
cat("\n--- Debug: Verificando variáveis 'novo_nome_...' antes da lista map_configs ---\n")


##
## --- Debug: Verificando variáveis 'novo_nome_...' antes da lista map_configs ---

print(paste0("Valor de novo_nome_coef: ", if(exists("novo_nome_coef")) novo_nome_coef else "NÃO DEFINID

## [1] "Valor de novo_nome_coef: GWR_coef_log1p_n_drogas"

print(paste0("Valor de novo_nome_tval: ", if(exists("novo_nome_tval")) novo_nome_tval else "NÃO DEFINID

## [1] "Valor de novo_nome_tval: GWR_tval_log1p_n_drogas"

print(paste0("Valor de novo_nome_intercept: ", if(exists("novo_nome_intercept")) novo_nome_intercept els

## [1] "Valor de novo_nome_intercept: GWR_coef_Intercept"

print(paste0("Valor de novo_nome_tval_intercept: ", if(exists("novo_nome_tval_intercept")) novo_nome_tva

## [1] "Valor de novo_nome_tval_intercept: NÃO DEFINIDO"

print(paste0("Valor de novo_nome_r2_local: ", if(exists("novo_nome_r2_local")) novo_nome_r2_local else

## [1] "Valor de novo_nome_r2_local: GWR_Local_R2"

print(paste0("Valor de novo_nome_yhat: ", if(exists("novo_nome_yhat")) novo_nome_yhat else "NÃO DEFINID

## [1] "Valor de novo_nome_yhat: GWR_yhat"
```

```r
print(paste0("Valor de novo_nome_residual: ", if(exists("novo_nome_residual")) novo_nome_residual else "
```

```
## [1] "Valor de novo_nome_residual: GWR_residual"
```

```r
print(paste0("Valor de novo_nome_y_observado: ", if(exists("novo_nome_y_observed")) novo_nome_y_observe
```

```
## [1] "Valor de novo_nome_y_observado: NÃO DEFINIDO"
```

```r
cat("--------------------------------------------------------------------------------\n\n")
```

```
## --------------------------------------------------------------------------------
```

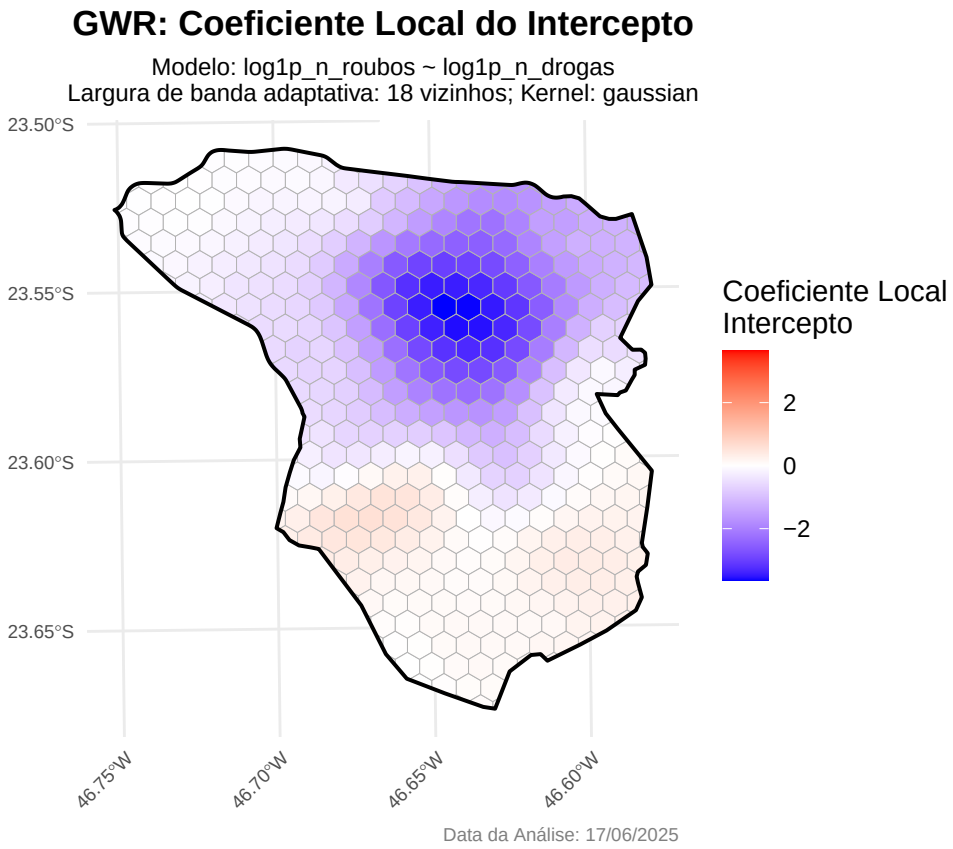
```r
# ================================================================================


# Nomes das colunas a serem mapeadas e seus títulos/descrições para o loop de plotagem
# ESTA LISTA AGORA CONTÉM APENAS OS MAPAS DE RESULTADOS GWR CONFORME SOLICITADO
map_configs <- list(
    # --- 1. Mapas para Analisar a Relação Drogas-Crimes (Coeficientes e Significância): ---
    # c. Intercept
    # Descrição: Este é o intercepto local estimado para cada célula da grade no GWR.
    # Motivo para Mapear: Indica o valor esperado de log1p_n_roubos quando todas as variáveis independe
    list(col = novo_nome_intercept, name = paste0("Coeficiente Local\nIntercepto"), title = paste0("GWR
         desc = "Mostra o valor base estimado para os roubos (na escala log1p) em cada localidade quand

    # Significância do Intercepto Local (Adicionado por completude dos resultados GWR, embora não lista
    # É útil para saber onde o intercepto é estatisticamente diferente de zero.
    list(col = novo_nome_intercept_tv, name = paste0("Valor-t Local\nIntercepto"), title = paste0("GWR:
         desc = "Indica a significância estatística do Intercepto local. Valores |t| > ~1.96 sugerem sig

    # a. coef_log1p_n_drogas
    # Descrição: Este é o coeficiente local estimado (b) para a relação entre log1p_n_drogas (indicador
    # Motivo para Mapear: Mostra como a força da relação entre drogas e crimes varia no espaço. Permite
    list(col = novo_nome_coef, name = paste0("Coeficiente Local\n(", var_indep_na_formula, ")"), title =
         desc = paste0("Mede a força e direção da associação local entre '", var_indep_na_formula, "' e

    # b. log1p_n_drogas_TV
    # Descrição: Este é o valor-t local associado ao coeficiente coef_log1p_n_drogas.
    # Motivo para Mapear: Permite identificar onde o coeficiente local é estatisticamente significativo
    # Saída esperada: Um mapa binário ou categorizado (a escala divergente ajuda a ver positivo/negativ
    list(col = novo_nome_tval, name = paste0("Valor-t Local\n(", var_indep_na_formula, ")"), title = pas
         desc = paste0("Indica onde o coeficiente local de '", var_indep_na_formula, "' é estatisticame

    # --- 2. Mapas de Ajuste e Diagnóstico do Modelo: ---
    # a. Local_R2
    # Descrição: Este é o R² local, que informa quanto da variação local na variável dependente (log1p_
    # Motivo para Mapear: Quão bem o modelo performa em cada área? Áreas com Local_R2 mais altos reflet
    list(col = novo_nome_r2_local, name = "R2 Local\n(Ajuste do Modelo)", title = "GWR: R2 Local (Poder
         desc = "Mostra a proporção da variação local da VD explicada pelo modelo. Áreas com R2 alto = M

    # c. yhat (Colocado antes de residual porque é o valor que gera o residual)
```

```
    # Descrição: Este é o valor previsto do modelo para cada hexágono (log1p_n_roubos previsto com base
    # Motivo para Mapear: Permite visualizar os valores previstos de crime diretamente. Útil para compa
     list(col = novo_nome_yhat, name = paste0("VD Prevista\n(Escala Transf.)"), title = paste0("GWR: VD
          desc = paste0("Valores de '", all.vars(formula_gwr)[1], "' previstos pelo modelo GWR local.")

    # Variável Dependente Observada Transformada (Para comparação visual com yhat e residual)
    # É crucial mapear a VD observada na mesma escala para entender yhat e residual.
    # Descrição: A variável dependente observada na escala transformada (log1p_n_roubos).
    # Motivo para Mapear: Comparação visual direta com os valores previstos (yhat) e para entender os r
    list(col = novo_nome_y_observado, name = paste0("VD Observada\n(Escala Transf.)"), title = paste0("
         desc = paste0("Valores de '", all.vars(formula_gwr)[1], "' observados (input para o GWR)."),

    # b. residual
    # Descrição: Este é o resíduo local, calculado como a diferença entre o valor observado (log1p_n_ro
    # Motivo para Mapear: A análise dos resíduos pode revelar padrões espaciais não capturados pelo mod
    list(col = novo_nome_residual, name = paste0("Resíduo Local"), title = paste0("GWR: Resíduos do Mod
         desc = "Diferença entre valor observado e previsto (VD transformada). Mostra onde o modelo sub
)

# Loop para gerar e salvar cada mapa definido em map_configs
for (map_cfg in map_configs) {
    col_name <- map_cfg$col # Nome da coluna no objeto SF final
    map_title <- map_cfg$title # Título principal do mapa
    legend_name <- map_cfg$name # Nome para a legenda da cor
    map_type <- map_cfg$type # Tipo de escala (sequential, diverging, sequential_0_1)
    file_suffix <- map_cfg$file_suffix # Sufixo para o nome do arquivo PNG
    # Descrição é útil nos comentários do código, mas não usada na plotagem direta aqui.

    # Verifica novamente se a coluna existe no objeto SF COMBINADO antes de tentar plotar
    # Usa [[col_name]] para acessar a coluna pelo nome da string de forma segura
    if (!(col_name %in% names(resultados_gwr_poligonos_sf))) {
        warning(paste0("AVISO Bloco 9: Coluna '", col_name, "' não encontrada no objeto 'resultados_gwr
        next # Pula para a próxima configuração de mapa no loop
    }

    message(paste0(" -> Gerando mapa para: '", legend_name, "' (coluna: '", col_name, "')..."))

    tryCatch({
        # --- Configura a escala de cor baseada no tipo ---
        if (map_type == "sequential") {
            # Escala sequencial padrão (ex: Viridis)
            color_scale <- scale_fill_viridis_c(name = legend_name, option = "viridis", na.value = "grey
        } else if (map_type == "diverging") {
            # Escala divergente (centralizada em 0)
            # Calcula o limite máximo absoluto para centralizar a escala de cor
            # Acessa a coluna de dados de forma segura com [[col_name]]
            max_abs_val <- max(abs(resultados_gwr_poligonos_sf[[col_name]]), na.rm = TRUE)
            # Define limites para a escala, garantindo que não sejam infinitos ou muito pequenos
            plot_limits <- c(-max_abs_val, max_abs_val)
            if (!is.finite(max_abs_val) || max_abs_val == 0) {
                plot_limits <- c(-1, 1) # Limite padrão seguro se valores são zero/infinito
            } else if (max_abs_val < 0.1) {
                plot_limits <- c(-0.1, 0.1) # Ajusta limite se valores forem muito pequenos
```

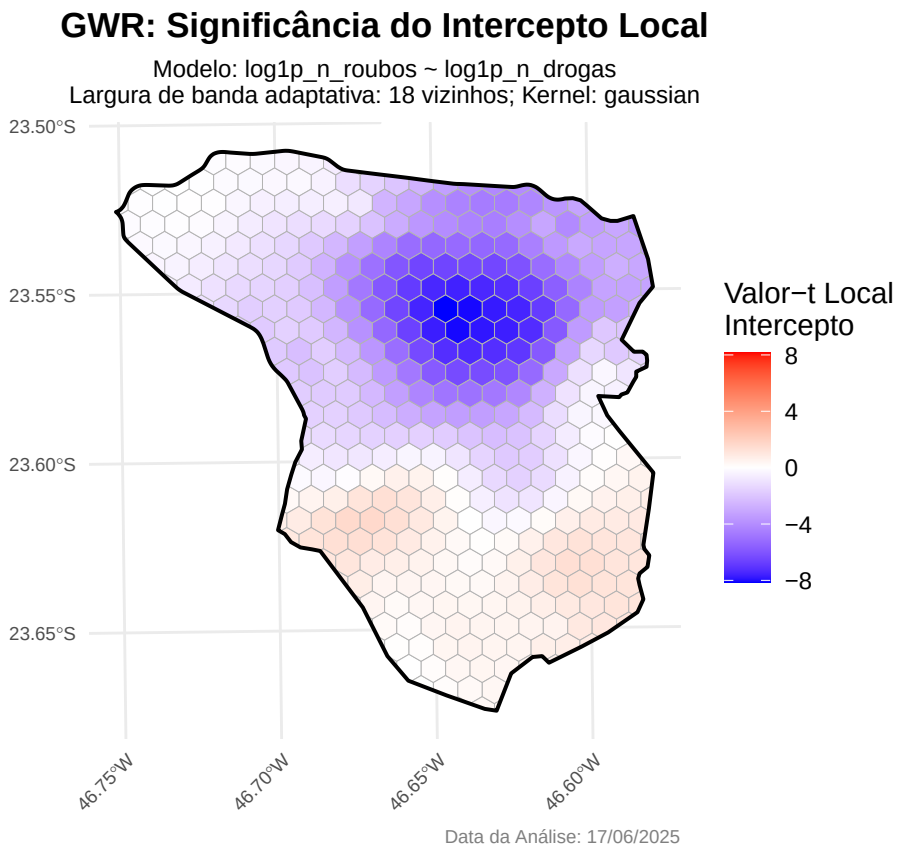
```
        }

        color_scale <- scale_fill_gradient2(
            name = legend_name,
            low = "blue", mid = "white", high = "red", # Azul (negativo) - Branco (zero) - Vermelho
            midpoint = 0, # Centraliza em zero
            limits = plot_limits, # Define os limites da escala
            space = "Lab",
            na.value = "grey80", # Cor para valores NA
            oob = scales::squish # Comprime valores fora do limite para o limite mais próximo
        )
    } else if (map_type == "sequential_0_1") {
         # Escala sequencial específica para valores entre 0 e 1 (como R2)
         # Trunca valores < 0 para 0 para visualização no mapa (R2 não deve ser negativo)
         plot_data_r2 <- resultados_gwr_poligonos_sf
         # Acessa a coluna de dados de forma segura com [[col_name]]
         plot_data_r2[[col_name]] <- ifelse(plot_data_r2[[col_name]] < 0, 0, plot_data_r2[[col_name]
         # Define limite superior do R2 (máx entre 1 e o valor máximo observado >=0)
         lim_max_r2 <- max(1, max(plot_data_r2[[col_name]], na.rm = TRUE))
         color_scale <- scale_fill_viridis_c(
             name = legend_name,
             option = "plasma", # Outra opção de paleta sequencial
             limits = c(0, lim_max_r2), # Limita a escala de 0 até o valor máximo (ou 1)
             labels = scales::percent_format(accuracy = 1), # Formata legenda como porcentagem
             na.value = "grey80", # Cor para valores NA
             oob = scales::squish # Comprime valores fora do limite
         )
         # Usa os dados com R2 truncado para plotar, mas o objeto original permanece inalterado
         data_to_plot <- plot_data_r2
    } else {
        # Tipo de mapa desconhecido, usa escala sequencial padrão
        warning(paste0("Tipo de mapa desconhecido: '", map_type, "'. Usando escala sequencial padrã
        color_scale <- scale_fill_viridis_c(name = legend_name, na.value = "grey80")
    }

    # Define os dados a serem plotados (usa plot_data_r2 se for tipo sequential_0_1, senão usa o ob
    data_to_plot <- if (map_type == "sequential_0_1") plot_data_r2 else resultados_gwr_poligonos_sf

    # --- Cria o objeto ggplot para o mapa atual ---
    p <- ggplot() +
         # Camada dos polígonos com os resultados GWR ou dados originais
         # Usa .data[[col_name]] dentro do aes para acessar a coluna pela string col_name
         geom_sf(data = data_to_plot, aes(fill = .data[[col_name]]), color = "grey70", linewidth = 0
         # Camada de contorno da área de estudo (se o objeto existir e for válido)
         {if(!is.null(area_estudo_union_mapa)) geom_sf(data = area_estudo_union_mapa, fill = NA, col
         # Aplica a escala de cor configurada acima
         color_scale +
         # Define títulos e legendas para o mapa
         labs(
             title = map_title,
             subtitle = paste0("Modelo: ", deparse(formula_gwr), # Mostra a fórmula usada no GWR
                               "\nLargura de banda adaptativa: ", round(gwr_resultado_lista$GW.argu
             caption = paste("Data da Análise:", format(Sys.Date(), "%d/%m/%Y")) # Data atual da an
```

```
            ) +
            # Aplica um tema minimalista para mapas
            theme_minimal(base_size = 11) +
            theme(
                plot.title = element_text(hjust = 0.5, face = "bold"), # Título centralizado e em negr
                plot.subtitle = element_text(hjust = 0.5, size = 9, lineheight = 0.9), # Subtítulo cen
                plot.caption = element_text(hjust = 1, size = 7, color = "grey50"), # Legenda no canto
                legend.position = "right", # Posição da legenda
                axis.text.x = element_text(angle = 45, hjust = 1, size = 7), # Rótulos do eixo X rotac
                axis.text.y = element_text(size = 7) # Rótulos do eixo Y
            ) +
            # Remove rótulos dos eixos (coordenadas) para um visual de mapa limpo
            labs(x = NULL, y = NULL)

        # Imprime o mapa (útil para visualização imediata no RStudio antes de salvar)
        print(p)

        # --- Salva o mapa em arquivo PNG ---
        if (output_map_dir_valid) { # Verifica se o diretório de saída é válido antes de tentar salvar
            file_name <- file.path(output_map_dir, paste0(file_suffix, ".png")) # Usa o sufixo definido
            # Salva o plot p no caminho e nome especificados
            ggsave(file_name, plot = p, width = 8, height = 7, units = "in", dpi = 300) # Define tamanh
            message(paste0("   Mapa salvo com sucesso em '", file_name, "'.\n"))
        } else {
             message("   Diretório de saída não definido ou não encontrado/válido. O mapa não foi salvo
        }

    }, error = function(e_plot) {
        # Captura e reporta erros específicos durante a plotagem ou salvamento de um mapa
        cat(paste0("ERRO ao gerar ou salvar o mapa para '", legend_name, "' (coluna '", col_name, "'):\n
        cat(paste("     Mensagem de erro:", e_plot$message, "\n\n"))
    })
}
```

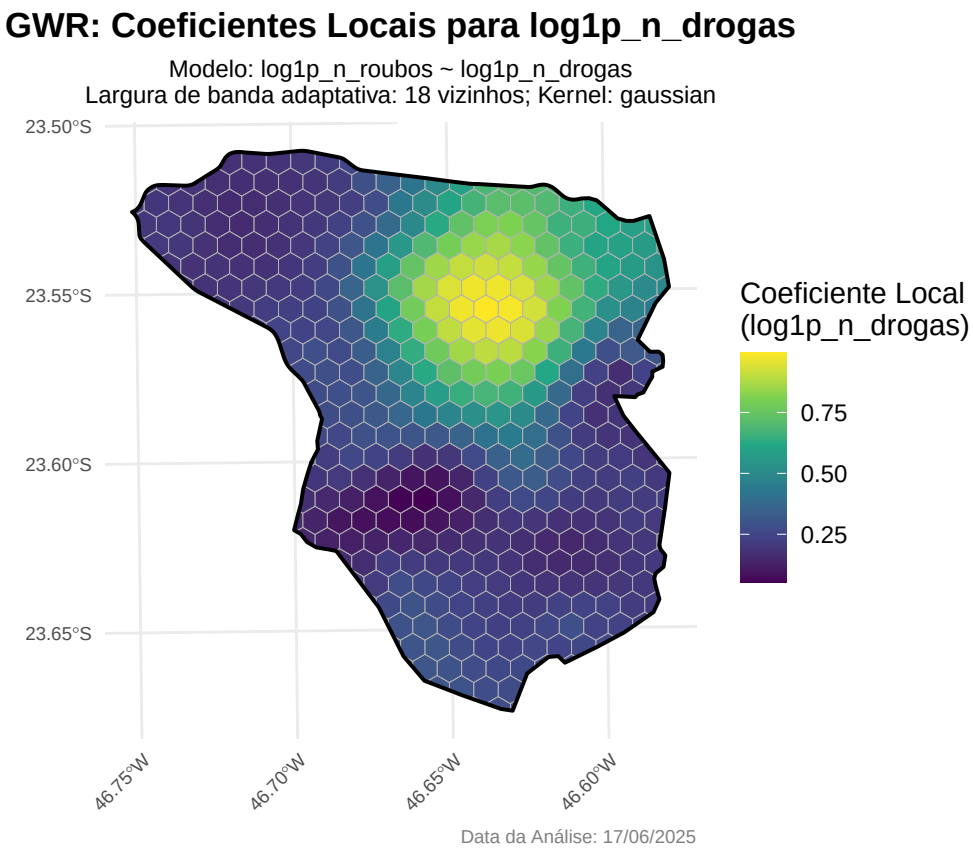
```
##  -> Gerando mapa para: 'Coeficiente Local
## Intercepto' (coluna: 'GWR_coef_Intercept')...
```



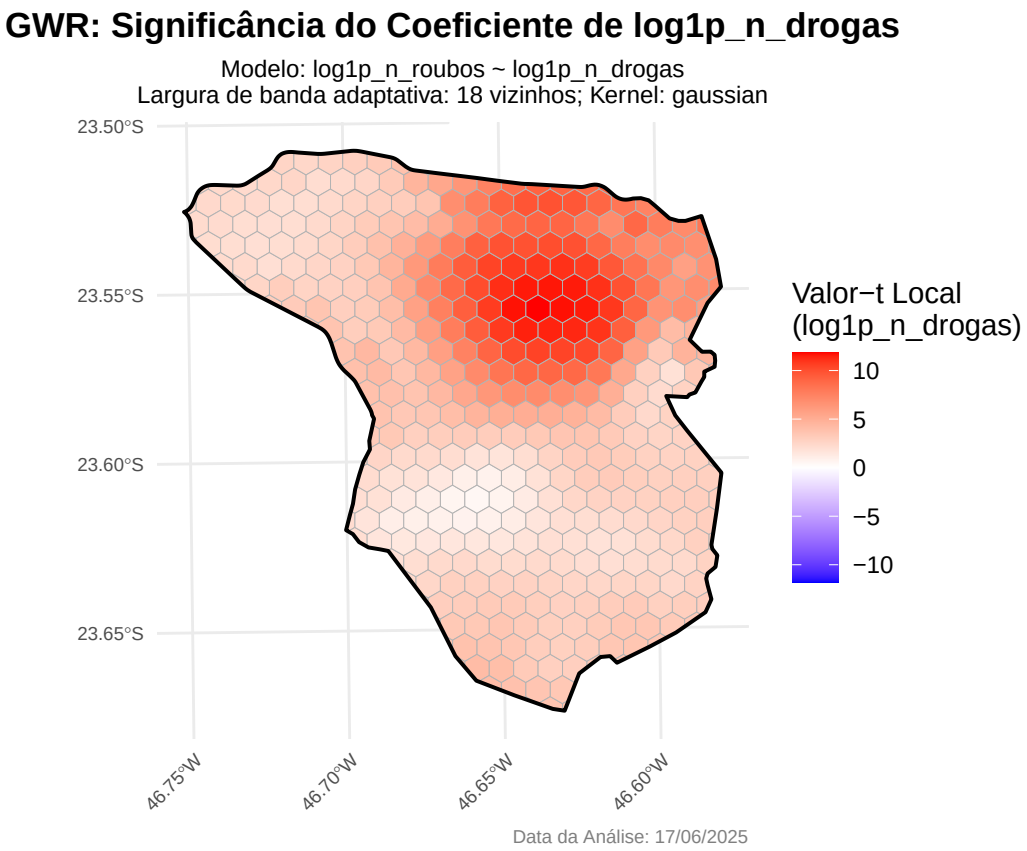

```
##    Mapa salvo com sucesso em 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda

##  -> Gerando mapa para: 'Valor-t Local
## Intercepto' (coluna: 'GWR_tval_Intercept')...
```



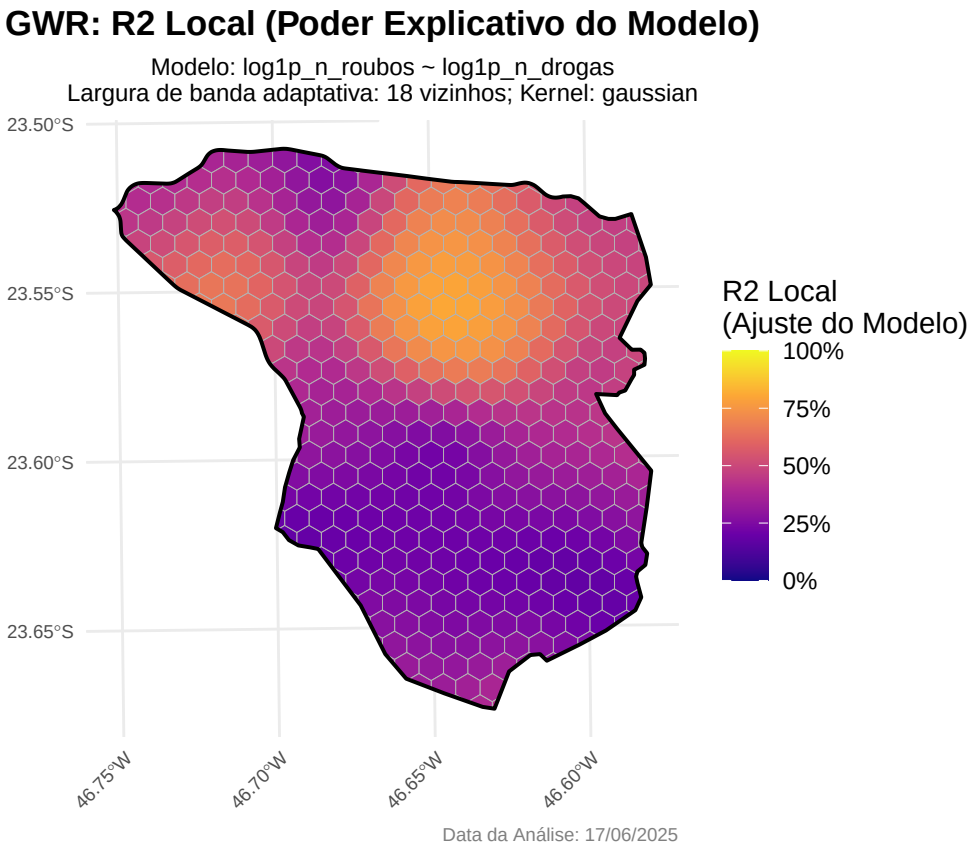

```
##    Mapa salvo com sucesso em 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda

##  -> Gerando mapa para: 'Coeficiente Local
## (log1p_n_drogas)' (coluna: 'GWR_coef_log1p_n_drogas')...
```




```
##    Mapa salvo com sucesso em 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda

##  -> Gerando mapa para: 'Valor-t Local
## (log1p_n_drogas)' (coluna: 'GWR_tval_log1p_n_drogas')...
```

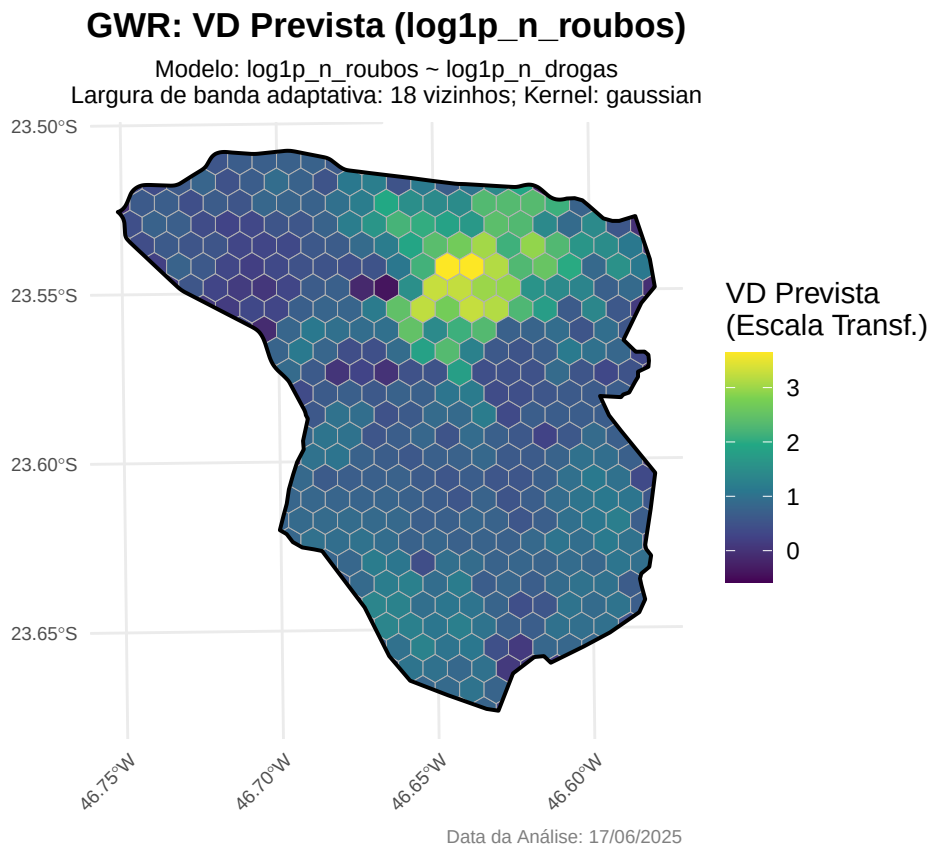



```
##    Mapa salvo com sucesso em 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda

##  -> Gerando mapa para: 'R2 Local
## (Ajuste do Modelo)' (coluna: 'GWR_Local_R2')...
```

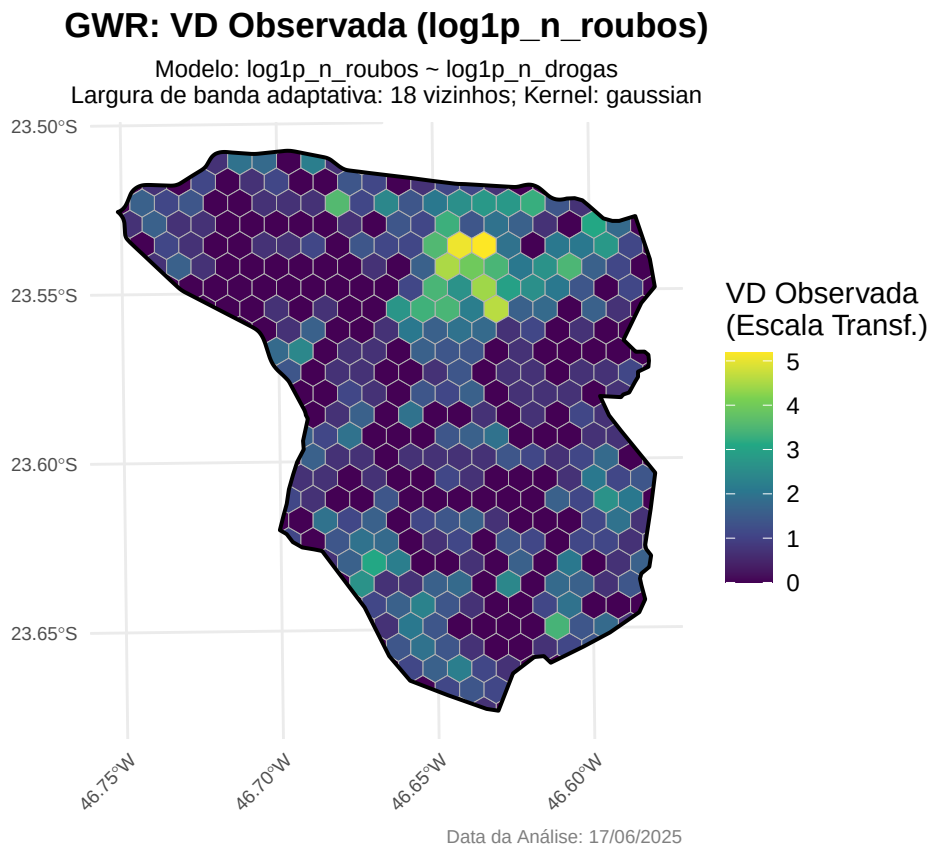



```
##    Mapa salvo com sucesso em 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda
```

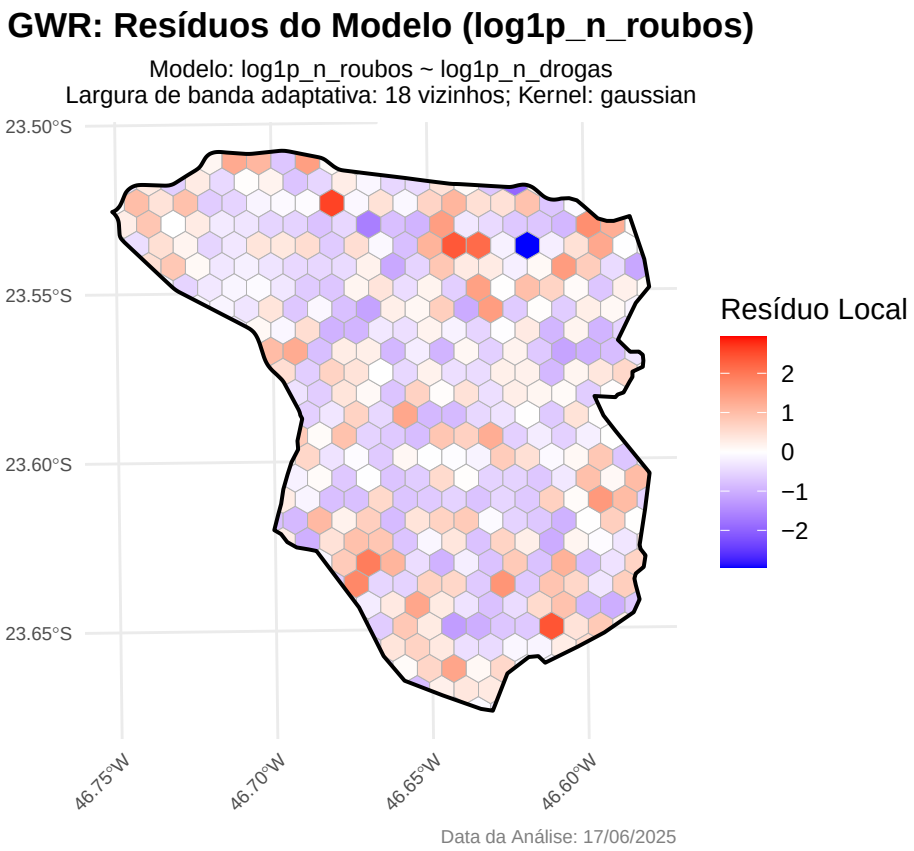
```
##  -> Gerando mapa para: 'VD Prevista
## (Escala Transf.)' (coluna: 'GWR_yhat')...
```




```
##    Mapa salvo com sucesso em 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda

##  -> Gerando mapa para: 'VD Observada
## (Escala Transf.)' (coluna: 'GWR_y_observed_transf')...
```




```
##    Mapa salvo com sucesso em 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda

##  -> Gerando mapa para: 'Resíduo Local' (coluna: 'GWR_residual')...
```




```
##    Mapa salvo com sucesso em 'C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda
```

```
message("\n--- Geração de mapas de resultados GWR concluída. Todos os mapas definidos foram processados
```

```
##
## --- Geração de mapas de resultados GWR concluída. Todos os mapas definidos foram processados. ---
```

```
# --- Fim do Bloco 9 ---
cat("\n--- Fim do BL. 9 ---\n")
```

```
##
## --- Fim do BL. 9 ---
```

```
cat("--------------------\n")
```

```
## --------------------
```

```
#--- Bloco 9: Combinar Resultados GWR aos Polígonos e Visualizar ---

message("\n--- Iniciando Bloco 9: Combinando Resultados GWR aos Polígonos e Preparando Visualizações ---
```

```
##
## --- Iniciando Bloco 9: Combinando Resultados GWR aos Polígonos e Preparando Visualizações ---
```

```r
# Verificar se os objetos necessários existem
if (!exists("dados_gwr_sf_validos") || is.null(dados_gwr_sf_validos)) {
    stop("ERRO CRÍTICO Bloco 9: 'dados_gwr_sf_validos' não encontrado ou é NULL.")
}
if (!exists("gwr_resultados_sdf") || is.null(gwr_resultados_sdf)) {
    stop("ERRO CRÍTICO Bloco 9: 'gwr_resultados_sdf' (output do GWR) não encontrado ou é NULL.")
}

# Combinar os resultados do GWR (que estão em formato SPDF, nos centróides)
# de volta aos polígonos originais (dados_gwr_sf_validos).
# É crucial garantir que a correspondência entre os resultados e os polígonos da grade seja correta.
# Como dados_spdf_gwr (entrada para gwr.basic) foi criado diretamente a partir de dados_gwr_sf_validos
# (na mesma ordem e com os mesmos pontos representativos), a ordem das linhas DEVE ser a mesma.
# No entanto, um join explícito por 'id_celula_grade' é mais robusto.

resultados_gwr_poligonos_sf <- NULL # Inicializar

if(nrow(dados_gwr_sf_validos) == nrow(gwr_resultados_sdf)) {
    message("Número de linhas corresponde entre polígonos originais e resultados GWR. Prosseguindo com a

    # Extrair os dados (atributos) do SDF do GWR para um data.frame.
    gwr_resultados_df_para_join <- as.data.frame(gwr_resultados_sdf)

    # Adicionar 'id_celula_grade' de 'dados_gwr_sf_validos' aos resultados do GWR para um join robusto.
    # Assumimos que a ordem das linhas em dados_gwr_sf_validos corresponde à ordem em gwr_resultados_sd
    # porque gwr_resultados_sdf foi gerado a partir de dados_spdf_gwr, que por sua vez
    # foi gerado a partir de dados_gwr_sf_validos.
    if ("id_celula_grade" %in% names(dados_gwr_sf_validos)) {
        gwr_resultados_df_para_join$id_celula_grade <- dados_gwr_sf_validos$id_celula_grade
        message("Coluna 'id_celula_grade' adicionada aos resultados do GWR (gwr_resultados_df_para_join
    } else {
        stop("ERRO CRÍTICO Bloco 9: 'id_celula_grade' não encontrada em 'dados_gwr_sf_validos'. Impossí
    }

    # --- INÍCIO DA MODIFICAÇÃO: Renomear coluna do coeficiente e valor-t da VI ---
    # A VI na fórmula é all.vars(formula_gwr)[2], que deve ser "log1p_n_drogas"
    var_indep_na_formula <- all.vars(formula_gwr)[2] # Ex: "log1p_n_drogas"

    nome_coef_original_no_sdf <- var_indep_na_formula
    novo_nome_coef_para_join <- paste0("coef_", var_indep_na_formula) # Ex: "coef_log1p_n_drogas"

    nome_tval_original_no_sdf <- paste0(var_indep_na_formula, "_TV") # Ex: "log1p_n_drogas_TV"
    # Não vamos renomear a coluna do valor-t, mas vamos referenciá-la pelo nome correto.

    if (nome_coef_original_no_sdf %in% names(gwr_resultados_df_para_join)) {
        names(gwr_resultados_df_para_join)[names(gwr_resultados_df_para_join) == nome_coef_original_no_s
        message(paste0("Coluna do coeficiente para '", nome_coef_original_no_sdf, "' nos resultados GWR
    } else {
        warning(paste0("Coluna do coeficiente '", nome_coef_original_no_sdf, "' não encontrada diretamer
    }
    # --- FIM DA MODIFICAÇÃO ---

    # Definir as colunas de resultados do GWR que serão mantidas após o join.
```

```
    # 'y' no SDF do GWR corresponde à variável dependente da fórmula (log1p_n_roubos).
    cols_gwr_a_manter <- c("id_celula_grade",
                           "Intercept", "Intercept_TV",
                           novo_nome_coef_para_join, nome_tval_original_no_sdf, # Usando os nomes corre
                           "Local_R2", "y", "yhat", "residual")

    # Filtrar colunas para o join, selecionando apenas as relevantes.
    # Usar `all_of` garante que apenas as colunas existentes e listadas sejam selecionadas,
    # gerando um erro se alguma coluna essencial estiver faltando.
    cols_existentes_no_join_df <- intersect(cols_gwr_a_manter, names(gwr_resultados_df_para_join))

    # Checar se colunas essenciais para plotagem estão presentes
    if (!(novo_nome_coef_para_join %in% cols_existentes_no_join_df)) {
        warning(paste0("AVISO Bloco 9: Coluna do coeficiente '", novo_nome_coef_para_join, "' não está
    }
    if (!(nome_tval_original_no_sdf %in% cols_existentes_no_join_df)) {
        warning(paste0("AVISO Bloco 9: Coluna do valor-t '", nome_tval_original_no_sdf, "' não está pres
    }
     if (!("Local_R2" %in% cols_existentes_no_join_df)) {
        warning(paste0("AVISO Bloco 9: Coluna 'Local_R2' não está presente para o join. O mapa de R² Lo
    }

    gwr_resultados_filtrados_df <- gwr_resultados_df_para_join %>%
                                   dplyr::select(dplyr::all_of(cols_existentes_no_join_df))

    # Realizar o join dos resultados do GWR com o objeto sf dos polígonos da grade.
    # dados_gwr_sf_validos já contém as colunas originais e as transformadas log1p.
    # Vamos selecionar explicitamente as colunas de dados_gwr_sf_validos para evitar duplicatas desnece
    # mantendo as colunas originais de contagem (n_roubos, n_drogas) e as transformadas (log1p_n_roubos
    # para referência e análise.

    resultados_gwr_poligonos_sf <- dplyr::left_join(
        dados_gwr_sf_validos, # Contém id_celula_grade, geometry, n_roubos, n_drogas, log1p_n_roubos, l
        gwr_resultados_filtrados_df, # Contém id_celula_grade e resultados GWR selecionados
        by = "id_celula_grade"
    )

    message("Resultados GWR combinados com os polígonos originais da grade usando 'id_celula_grade'.")

    # Verificação pós-join: NAs em colunas críticas podem indicar problemas de correspondência.
    if (any(is.na(resultados_gwr_poligonos_sf$Local_R2))) {
        warning("NAs encontrados em 'Local_R2' após o join. Isso pode indicar problemas na correspondên
    }

} else {
    stop(paste("Número de linhas nos resultados GWR (", nrow(gwr_resultados_sdf),
               ") não corresponde aos dados de polígonos originais (", nrow(dados_gwr_sf_validos),
               "). Não é possível combinar os resultados de forma segura. Verifique a integridade dos da
}
```

```
## Número de linhas corresponde entre polígonos originais e resultados GWR. Prosseguindo com a combinaçã
```

```
## Coluna 'id_celula_grade' adicionada aos resultados do GWR (gwr_resultados_df_para_join) para join.
```

```
## Coluna do coeficiente para 'log1p_n_drogas' nos resultados GWR renomeada para 'coef_log1p_n_drogas'.

## Resultados GWR combinados com os polígonos originais da grade usando 'id_celula_grade'.
```

```
# --- Visualizar os resultados ---
message("\n--- Preparando visualizações dos resultados do GWR (modelo com variáveis log1p) ---")
```

```
##
## --- Preparando visualizações dos resultados do GWR (modelo com variáveis log1p) ---
```

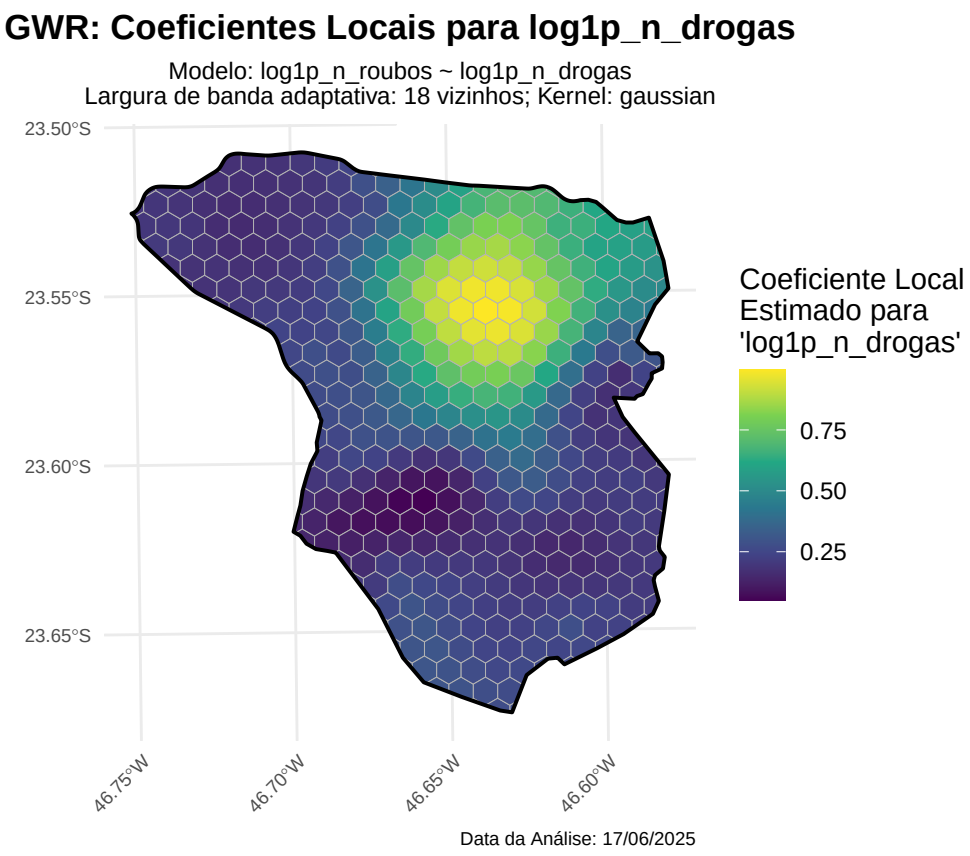
```
if (!is.null(resultados_gwr_poligonos_sf) && nrow(resultados_gwr_poligonos_sf) > 0) {
    message("Iniciando a criação dos mapas de resultados GWR...")
    cat("Primeiras linhas dos resultados GWR combinados com os polígonos (atributos):\n")
    print(head(st_drop_geometry(resultados_gwr_poligonos_sf)))
    col_names_results <- names(resultados_gwr_poligonos_sf)
    message(paste("\nNomes das colunas nos resultados finais (polígonos):\n", paste(col_names_results,

    # --- INÍCIO DA MODIFICAÇÃO: Atualizar nomes de colunas e display para variáveis transformadas ---
    # var_indep_na_formula foi definido acima como all.vars(formula_gwr)[2] (ex: "log1p_n_drogas")
    coef_col_name_plot <- novo_nome_coef_para_join # Ex: "coef_log1p_n_drogas"
    t_val_col_name_plot <- nome_tval_original_no_sdf # Ex: "log1p_n_drogas_TV"

    var_independente_nome_amigavel <- paste0(var_indep_na_formula) # Ex: "log1p_n_drogas"
    formula_display <- deparse(formula_gwr) # Ex: "log1p_n_roubos ~ log1p_n_drogas"
    # --- FIM DA MODIFICAÇÃO ---

    # -- Mapa 1: Coeficientes Locais da variável transformada --
    if (coef_col_name_plot %in% col_names_results) {
      message(paste("Criando mapa para os coeficientes locais da variável '", var_independente_nome_ami
      tryCatch({
        mapa_coef_drogas <- ggplot() +
          geom_sf(data = resultados_gwr_poligonos_sf, aes(fill = .data[[coef_col_name_plot]]), color =
          geom_sf(data = area_estudo_union, fill = NA, color = "black", linewidth = 0.7) +
          scale_fill_viridis_c(name = paste0("Coeficiente Local\nEstimado para\n'", var_independente_no
          labs(
            title = paste("GWR: Coeficientes Locais para", var_independente_nome_amigavel),
            subtitle = paste0("Modelo: ", formula_display,
                              "\nLargura de banda adaptativa: ", bw_adaptativa, " vizinhos; Kernel: ",
            caption = paste("Data da Análise:", format(Sys.Date(), "%d/%m/%Y"))
          ) +
          theme_minimal(base_size = 11) +
          theme(
            plot.title = element_text(hjust = 0.5, face = "bold"),
            plot.subtitle = element_text(hjust = 0.5, size = 9),
            plot.caption = element_text(hjust = 1, size = 7, lineheight = 1.1),
            legend.position = "right",
            axis.text.x = element_text(angle = 45, hjust = 1, size=7),
            axis.text.y = element_text(size=7)
          )
        print(mapa_coef_drogas)
        ggsave(paste0("mapa_coef_",var_indep_na_formula, "_gwr.png"), plot = mapa_coef_drogas, width = 8
        message(paste0("Mapa dos coeficientes locais de '", var_independente_nome_amigavel, "' salvo co
      }, error = function(e_plot) {
```

```
    cat(paste("Erro ao gerar ou salvar o mapa dos coeficientes locais de '", var_independente_nome_a
  })
} else {
  warning(paste0("Coluna do coeficiente '", coef_col_name_plot, "' não encontrada nos resultados. O
}

# -- Mapa 2: R² Local --
r2_col_name <- "Local_R2"
if (r2_col_name %in% col_names_results) {
  message(paste("Criando mapa para o R² Local (coluna '", r2_col_name, "')...", sep=""))
  tryCatch({
    resultados_gwr_poligonos_sf$Local_R2_plot <- ifelse(resultados_gwr_poligonos_sf[[r2_col_name]] <
    lim_max_r2 <- max(1, max(resultados_gwr_poligonos_sf$Local_R2_plot, na.rm = TRUE))

    mapa_r2_local <- ggplot() +
      geom_sf(data = resultados_gwr_poligonos_sf, aes(fill = Local_R2_plot), color = "grey70", linew
      geom_sf(data = area_estudo_union, fill = NA, color = "black", linewidth = 0.7) +
      scale_fill_viridis_c(name = "R² Local\n(Ajuste do Modelo)", option = "plasma", limits = c(0, 1
                           labels = scales::percent_format(accuracy = 1)) +
      labs(
        title = "GWR: R² Local (Poder Explicativo do Modelo)",
        subtitle = paste0("Modelo: ", formula_display,
                          "\nLargura de banda adaptativa: ", bw_adaptativa, " vizinhos; Kernel: ", k
        caption = paste("Valores de R² Local < 0 foram truncados para 0 na visualização.\nData da An
      ) +
      theme_minimal(base_size = 11) +
       theme(
        plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 9),
        plot.caption = element_text(hjust = 1, size = 7, lineheight = 0.9),
        legend.position = "right",
        axis.text.x = element_text(angle = 45, hjust = 1, size=7),
        axis.text.y = element_text(size=7)
      )
    print(mapa_r2_local)
    ggsave(paste0("mapa_r2_local_gwr_",var_indep_na_formula,".png"), plot = mapa_r2_local, width = 8
    message(paste0("Mapa do R² local salvo como mapa_r2_local_gwr_",var_indep_na_formula,".png."))
  }, error = function(e_plot) {
    cat(paste("Erro ao gerar ou salvar o mapa do R² local:\n", e_plot$message, "\n"))
  })
} else {
  warning(paste0("Coluna '", r2_col_name, "' não encontrada nos resultados. O mapa de R² Local não s
}

# -- Mapa 3: Valores-t Locais do Coeficiente da variável transformada --
if (t_val_col_name_plot %in% col_names_results) {
  message(paste("Criando mapa para os valores-t locais do coeficiente de '", var_independente_nome_a
  tryCatch({
    lim_max_abs_t <- max(abs(resultados_gwr_poligonos_sf[[t_val_col_name_plot]]), na.rm = TRUE)
    if (!is.finite(lim_max_abs_t) || is.na(lim_max_abs_t) || lim_max_abs_t == 0) {
        lim_max_abs_t <- 2
    }
```

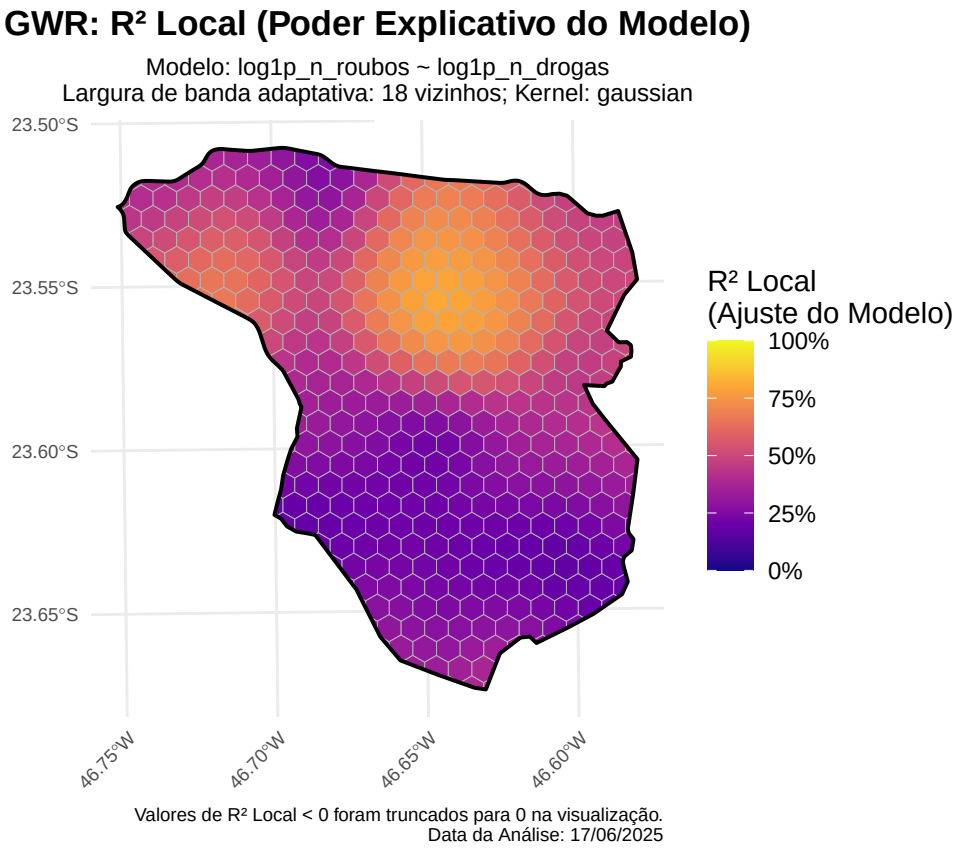
```
        mapa_t_valor_drogas <- ggplot() +
          geom_sf(data = resultados_gwr_poligonos_sf, aes(fill = .data[[t_val_col_name_plot]]), color =
          geom_sf(data = area_estudo_union, fill = NA, color = "black", linewidth = 0.7) +
          scale_fill_gradient2(
            name = paste0("Valor-t Local\n(", var_independente_nome_amigavel, ")"),
            low = "blue", mid = "white", high = "red",
            midpoint = 0,
            limits = c(-lim_max_abs_t, lim_max_abs_t),
            oob = scales::squish
          ) +
          labs(
            title = paste("GWR: Significância do Coeficiente de", var_independente_nome_amigavel),
            subtitle = paste0("Valores-t locais. |t| > ~1.96 sugere significância a p < 0.05 (bilateral
                              "\nLargura de banda: ", bw_adaptativa, " vizinhos; Kernel: ", kernel_usado
                              "\nModelo: ", formula_display),
            caption = paste("Data da Análise:", format(Sys.Date(), "%d/%m/%Y"))
          ) +
          theme_minimal(base_size = 11) +
          theme(
            plot.title = element_text(hjust = 0.5, face = "bold"),
            plot.subtitle = element_text(hjust = 0.5, size = 9, lineheight = 0.9),
            plot.caption = element_text(hjust = 1, size = 7),
            legend.position = "right",
            axis.text.x = element_text(angle = 45, hjust = 1, size=7),
            axis.text.y = element_text(size=7)
          )
        print(mapa_t_valor_drogas)
        ggsave(paste0("mapa_t_valor_",var_indep_na_formula, "_gwr.png"), plot = mapa_t_valor_drogas, wi
        message(paste0("Mapa dos valores-t locais de '", var_independente_nome_amigavel, "' salvo como
      }, error = function(e_plot) {
        cat(paste("Erro ao gerar ou salvar o mapa dos valores-t de '", var_independente_nome_amigavel,
      })
    } else {
      warning(paste0("Coluna do valor-t '", t_val_col_name_plot, "' não encontrada nos resultados. O ma
    }

    # Mapa de significância opcional (precisaria ser ajustado para usar t_val_col_name_plot e formula_d
    # ...

} else {
    message("Nenhum resultado GWR válido (resultados_gwr_poligonos_sf) para plotar. Verifique as etapas
}
```

```
## Iniciando a criação dos mapas de resultados GWR...

## Primeiras linhas dos resultados GWR combinados com os polígonos (atributos):
##   id_celula_grade n_roubos n_drogas log1p_n_roubos log1p_n_drogas   Intercept
## 1               1        0        0      0.0000000       0.000000 -0.04735255
## 2               2        0        0      0.0000000       0.000000 -0.02552189
## 3               3        1       10      0.6931472       2.397895 -0.03699986
## 4               4        0        5      0.0000000       1.791759 -0.01846921
## 5               5        0        9      0.0000000       2.302585 -0.06323534
## 6               6        4       26      1.6094379       3.295837 -0.02155795
```

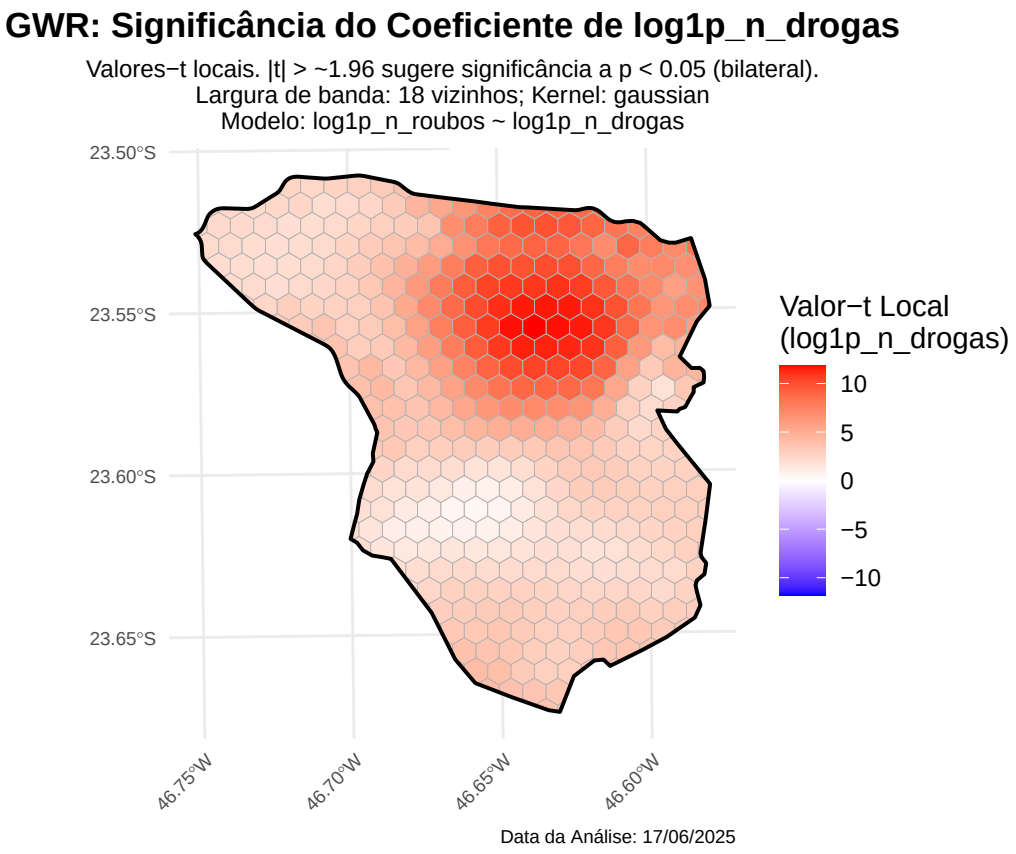
```
##   Intercept_TV coef_log1p_n_drogas log1p_n_drogas_TV  Local_R2         y
## 1  -0.15952030           0.1944990          2.268392 0.4621949 0.0000000
## 2  -0.08860688           0.1937619          2.356343 0.4119160 0.0000000
## 3  -0.12338008           0.1972008          2.258865 0.4369848 0.6931472
## 4  -0.06663292           0.1902470          2.434022 0.3945564 0.0000000
## 5  -0.19811937           0.2045261          2.132905 0.4846298 0.0000000
## 6  -0.07466145           0.1936369          2.329266 0.4061033 1.6094379
##          yhat    residual
## 1 -0.04735255  0.04735255
## 2 -0.02552189  0.02552189
## 3  0.43586710  0.25728008
## 4  0.32240757 -0.32240757
## 5  0.40770352 -0.40770352
## 6  0.61663758  0.99280033

##
## Nomes das colunas nos resultados finais (polígonos):
##  id_celula_grade, n_roubos, n_drogas, log1p_n_roubos, log1p_n_drogas, Intercept, Intercept_TV, coef_l

## Criando mapa para os coeficientes locais da variável 'log1p_n_drogas' (coluna 'coef_log1p_n_drogas')
```




```
## Mapa dos coeficientes locais de 'log1p_n_drogas' salvo como mapa_coef_log1p_n_drogas_gwr.png.

## Criando mapa para o R² Local (coluna 'Local_R2')...
```




```
## Mapa do R² local salvo como mapa_r2_local_gwr_log1p_n_drogas.png.

## Criando mapa para os valores-t locais do coeficiente de 'log1p_n_drogas' (coluna 'log1p_n_drogas_TV')
```




```
## Mapa dos valores-t locais de 'log1p_n_drogas' salvo como mapa_t_valor_log1p_n_drogas_gwr.png.
```

```
message("\n--- Análise GWR (modelo com variáveis log1p) com grade hexagonal e visualizações concluídas
```

```
##
## --- Análise GWR (modelo com variáveis log1p) com grade hexagonal e visualizações concluídas ---
```

```
# ```

# (O script continua com os Blocos 10 e 11)
```

```
# Bloco Final: Inspeção dos Nomes das Colunas Finais
if (!is.null(resultados_gwr_poligonos_sf) && nrow(resultados_gwr_poligonos_sf) > 0) {
    cat("\n\n--- Nomes Finais das Colunas no Objeto 'resultados_gwr_poligonos_sf' ---\n")
    print(names(resultados_gwr_poligonos_sf))
    cat("\n")
} else {
    cat("\n'resultados_gwr_poligonos_sf' é NULL ou vazio. Verifique as etapas anteriores.\n")
}
```

```
##
##
## --- Nomes Finais das Colunas no Objeto 'resultados_gwr_poligonos_sf' ---
##  [1] "id_celula_grade"     "n_roubos"            "n_drogas"
```

```
##  [4] "log1p_n_roubos"      "log1p_n_drogas"      "Intercept"
##  [7] "Intercept_TV"        "coef_log1p_n_drogas" "log1p_n_drogas_TV"
## [10] "Local_R2"            "y"                   "yhat"
## [13] "residual"            "geometry"            "Local_R2_plot"
```

```
# Bloco 10: Resumo dos Resultados GWR e Análise de Resíduos
cat("\n--- Bloco 10: Resumo dos Resultados GWR e Análise de Resíduos ---\n")
```

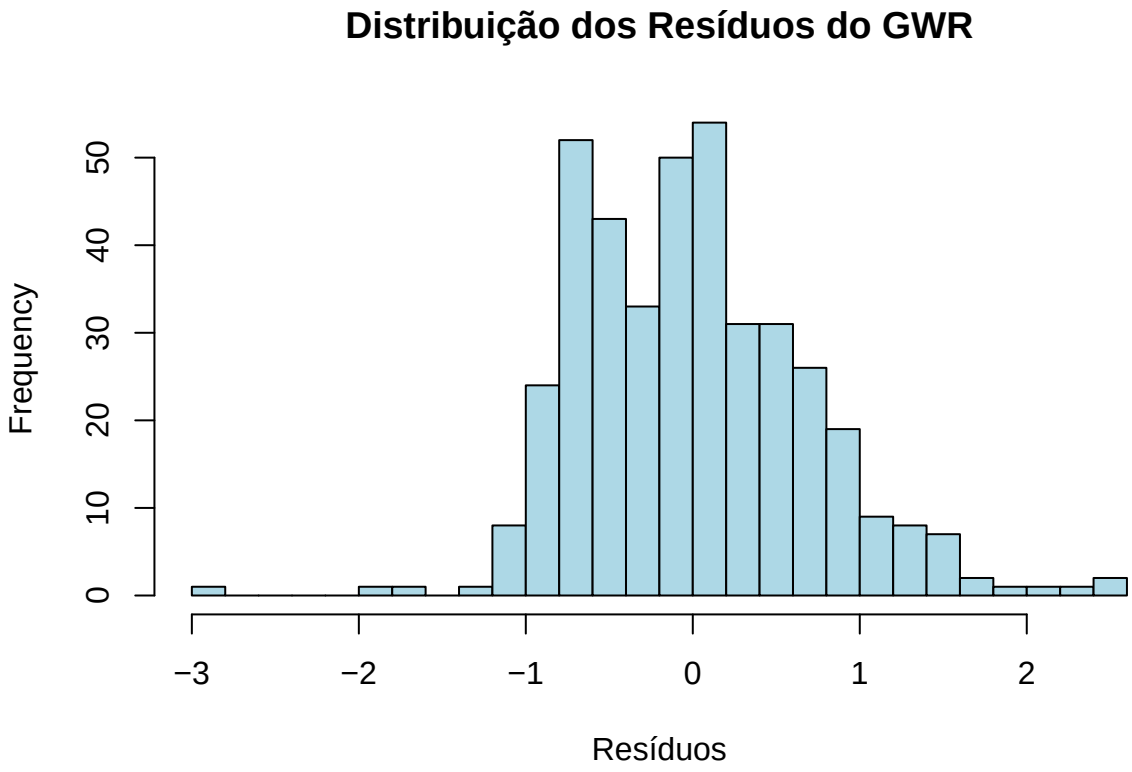
```
##
## --- Bloco 10: Resumo dos Resultados GWR e Análise de Resíduos ---
```

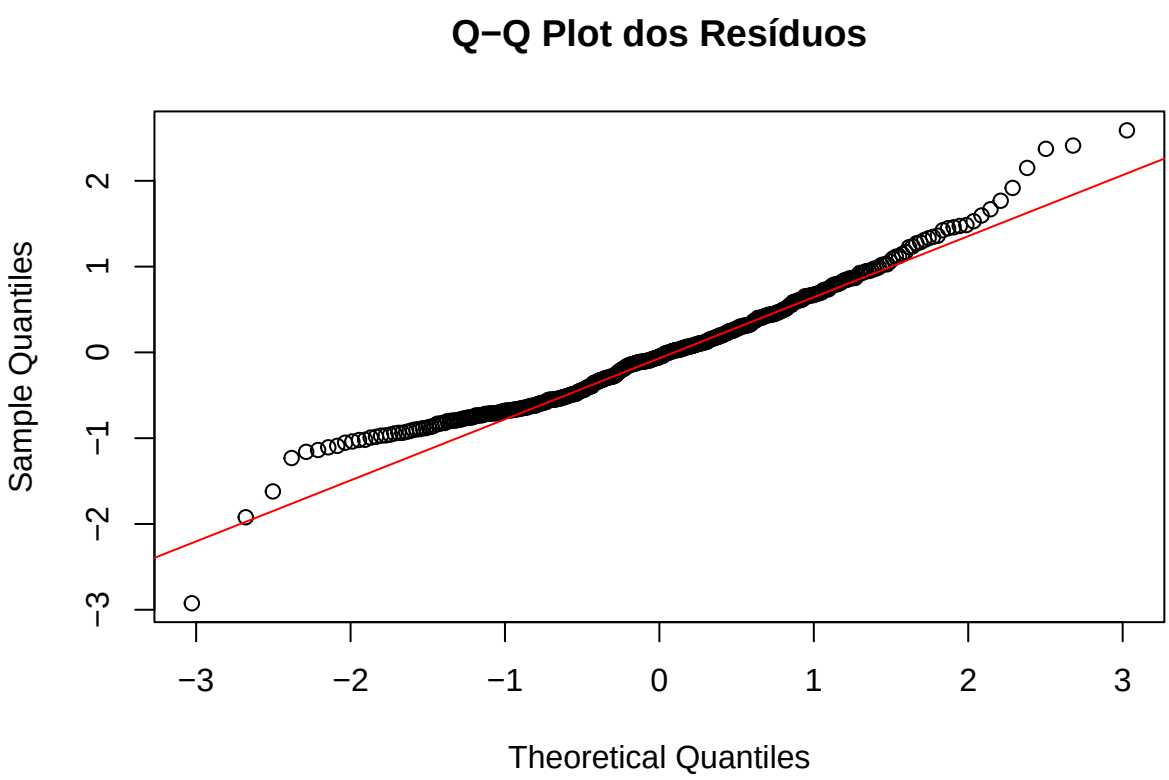
```
if (!is.null(resultados_gwr_poligonos_sf) && nrow(resultados_gwr_poligonos_sf) > 0) {

    # SEÇÃO 1: Resumo do R² Local
    if ("Local_R2" %in% names(resultados_gwr_poligonos_sf)) {
        cat("\n--- SEÇÃO 1: Resumo do R² Local ---\n")
        r2_summary <- summary(resultados_gwr_poligonos_sf$Local_R2)
        print(r2_summary)
        cat("\n")
    } else {
        cat("Coluna 'Local_R2' não encontrada. Verifique os resultados do GWR.\n")
    }

    # SEÇÃO 2: Resumo dos Coeficientes
    coef_col_name <- "coef_n_drogas"
    if (coef_col_name %in% names(resultados_gwr_poligonos_sf)) {
        cat(paste0("\n--- SEÇÃO 2: Resumo dos Coeficientes '", coef_col_name, "' ---\n"))
        coef_summary <- summary(resultados_gwr_poligonos_sf[[coef_col_name]])
        print(coef_summary)
        cat("\n")
    } else {
        cat(paste0("Coluna de coeficiente '", coef_col_name, "' não encontrada.\n"))
    }

    # SEÇÃO 3: Resumo dos Valores-t
    t_val_col_name <- "n_drogas_TV"
    if (t_val_col_name %in% names(resultados_gwr_poligonos_sf)) {
        cat(paste0("\n--- SEÇÃO 3: Resumo dos Valores-t '", t_val_col_name, "' ---\n"))
        t_val_summary <- summary(resultados_gwr_poligonos_sf[[t_val_col_name]])
        print(t_val_summary)
        cat("\n")
    } else {
        cat(paste0("Coluna de valor-t '", t_val_col_name, "' não encontrada.\n"))
    }

    # SEÇÃO 4: Análise de Resíduos
    if ("residual" %in% names(resultados_gwr_poligonos_sf)) {
        cat("\n--- SEÇÃO 4: Análise de Resíduos ---\n")
        residual_summary <- summary(resultados_gwr_poligonos_sf$residual)
        print(residual_summary)
        cat("\n")

        # Histograma dos resíduos
```

```
        hist(resultados_gwr_poligonos_sf$residual,
             main = "Distribuição dos Resíduos do GWR",
             xlab = "Resíduos",
             col = "lightblue",
             breaks = 30)

        # Q-Q plot dos resíduos
        qqnorm(resultados_gwr_poligonos_sf$residual,
               main = "Q-Q Plot dos Resíduos")
        qqline(resultados_gwr_poligonos_sf$residual,
               col = "red")
    } else {
        cat("Coluna 'residual' não encontrada. Análise de resíduos não pode ser realizada.\n")
    }

    # SEÇÃO 5: Teste de Autocorrelação Espacial dos Resíduos (I de Moran)
    if ("residual" %in% names(resultados_gwr_poligonos_sf) && nrow(resultados_gwr_poligonos_sf) > 0) {
        cat("\n--- SEÇÃO 5: Teste de Autocorrelação Espacial dos Resíduos ---\n")

        tryCatch({
            # Criar matriz de pesos espaciais
            coords_residuos <- st_coordinates(st_centroid(resultados_gwr_poligonos_sf))
            nb_residuos <- knn2nb(knearneigh(coords_residuos, k = 8))
            listw_residuos <- nb2listw(nb_residuos, style = "W")

            # Teste I de Moran
            moran_test <- moran.test(resultados_gwr_poligonos_sf$residual, listw_residuos)
            cat("\nTeste I de Moran para os resíduos:\n")
            print(moran_test)

            if (moran_test$p.value < 0.05) {
                cat("\nAVISO: Autocorrelação espacial significativa detectada nos resíduos (p < 0.05).\n
                cat("Isso pode indicar que o modelo GWR não capturou completamente a estrutura espacial
            } else {
                cat("\nNenhuma autocorrelação espacial significativa nos resíduos (p >= 0.05).\n")
                cat("O modelo GWR parece ter capturado adequadamente a estrutura espacial.\n")
            }
        }, error = function(e) {
            cat("\nErro ao calcular o I de Moran para os resíduos.\n")
        })
    } else {
        cat("Coluna 'residual' não encontrada em 'resultados_gwr_poligonos_sf' ou o objeto está vazio.\n
        cat("A análise de autocorrelação espacial dos resíduos não pode ser realizada.\n")
    }

}
```

```
##
## --- SEÇÃO 1: Resumo do R² Local ---
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1788  0.2551  0.3980  0.4171  0.5398  0.8013
##
## Coluna de coeficiente 'coef_n_drogas' não encontrada.
```

```
## Coluna de valor-t 'n_drogas_TV' não encontrada.
##
## --- SEÇÃO 4: Análise de Resíduos ---
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.924125 -0.548303 -0.048512 -0.004367  0.411694  2.588215
```






```
##
## --- SEÇÃO 5: Teste de Autocorrelação Espacial dos Resíduos ---

## Warning: st_centroid assumes attributes are constant over geometries

##
## Teste I de Moran para os resíduos:
##
##  Moran I test under randomisation
##
## data:  resultados_gwr_poligonos_sf$residual
## weights: listw_residuos
##
## Moran I statistic standard deviate = 3.2095, p-value = 0.0006649
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.074480965      -0.002469136       0.000574839
##
##
## AVISO: Autocorrelação espacial significativa detectada nos resíduos (p < 0.05).
## Isso pode indicar que o modelo GWR não capturou completamente a estrutura espacial.
```

```r
cat("\n--- Fim do Bloco 10: Resumo GWR (com variáveis transformadas log1p e Análise de Resíduos) ---\n")
```

```
##
## --- Fim do Bloco 10: Resumo GWR (com variáveis transformadas log1p e Análise de Resíduos) ---
```

```r
# --- Bloco 10 Final: Resumo Completo dos Resultados e Diagnósticos do Modelo GWR ---
# Consolida e refina a análise final, incluindo diagnósticos e interpretações aprofundadas.
cat("\n\n--- BL. 10: RESUMO COMPLETO DOS RESULTADOS E DIAGNÓSTICOS DO MODELO GWR ---")
```

```
##
##
## --- BL. 10: RESUMO COMPLETO DOS RESULTADOS E DIAGNÓSTICOS DO MODELO GWR ---
```

```r
cat("\n--------------------------------------------------------------------------\n")
```

```
##
## --------------------------------------------------------------------------
```

```r
# Verifica se os objetos necessários existem e não estão vazios antes de prosseguir
if (!exists("gwr_resultado_lista") || is.null(gwr_resultado_lista) ||
    !exists("resultados_gwr_poligonos_sf") || is.null(resultados_gwr_poligonos_sf) ||
    nrow(resultados_gwr_poligonos_sf) == 0) {

    cat("\nERRO: Os resultados do GWR ('gwr_resultado_lista' ou 'resultados_gwr_poligonos_sf') não foram
    cat("Impossível gerar o relatório de análise final. Verifique se os blocos anteriores (carregamento

} else { # Se os objetos existem e não estão vazios, procede com a análise

    cat("\n## 1. Informações Gerais do Modelo GWR ##")
    cat("\n-------------------------------------\n")
    # Usa a fórmula real do script (definida anteriormente, e.g., formula_gwr <- log1p_n_roubos ~ log1p_
    cat(paste0("Fórmula do modelo: ", deparse(formula_gwr), "\n"))
    cat(paste0("Tipo de kernel: ", gwr_resultado_lista$GW.arguments$kernel, "\n"))

    # Tenta extrair o método de seleção da largura de banda de forma segura
    bw_approach <- tryCatch({ gwr_resultado_lista$GW.arguments$approach }, error = function(e) "Não espe
    cat(paste0("Método de seleção de largura de banda: ", bw_approach, "\n"))

    cat(paste0("Largura de banda ótima (Adaptativa): ", round(gwr_resultado_lista$GW.arguments$bw, 2),
    cat(paste0("Número de observações (pontos de regressão): ", nrow(gwr_resultado_lista$SDF), "\n"))

    # Tenta extrair o CRS final projetado de forma segura
    final_crs_epsg <- tryCatch({ st_crs(resultados_gwr_poligonos_sf)$epsg }, error = function(e) NA_inte
    final_crs_epsg_str <- if (is.na(final_crs_epsg)) "Não disponível" else paste0("EPSG:", final_crs_eps
    cat(paste0("CRS Final Projetado Utilizado: ", final_crs_epsg_str, "\n"))

    cat("\n## 2. Diagnósticos Globais do Modelo GWR ##")
    cat("\n-----------------------------------------\n")

    # Extrai os diagnósticos do objeto GWR
    diagnostics <- gwr_resultado_lista$GW.diagnostic
```

```r
# Imprime diagnósticos essenciais usando a função robusta
print_diagnostic_robust(diagnostics$AIC, "AIC", "Métrica de qualidade de ajuste, penaliza complexida
print_diagnostic_robust(diagnostics$AICc, "AICc", "AIC corrigido para amostras pequenas. Mais confia
print_diagnostic_robust(diagnostics$BIC, "BIC", "Similar ao AIC/AICc, penaliza mais a complexidade.

# ENP (Effective Number of Parameters) - Tenta extrair de diferentes locais possíveis ou usa NULL s
enp_val <- tryCatch({ diagnostics$ENP }, error = function(e) NULL) # Tenta do slot principal
if (is.null(enp_val) && !is.null(gwr_resultado_lista$GW.arguments$ENP)) { # Tenta se estiver nos ar
    enp_val <- gwr_resultado_lista$GW.arguments$ENP
}
print_diagnostic_robust(enp_val, "ENP (Effective Number of Parameters)", "Mede a complexidade 'efeti

# EDF (Effective Degrees of Freedom) - Geralmente disponível em diagnostics
print_diagnostic_robust(diagnostics$edf, "EDF (Effective Degrees of Freedom)", "Graus de liberdade

# Sigma (Residual standard error) - Tenta extrair de diferentes locais ou calcula do resíduo se nec
sigma_val <- tryCatch({ diagnostics$sigma }, error = function(e) NULL) # Tenta do slot principal
if (is.null(sigma_val) || !is.finite(sigma_val)) { # Se não encontrou ou é inválido, tenta calcular
    sigma_val_alt <- tryCatch({ sd(resultados_gwr_poligonos_sf$residual, na.rm = TRUE) }, error = fu
    if (!is.null(sigma_val_alt) && is.finite(sigma_val_alt)) {
        sigma_val <- sigma_val_alt
        print_diagnostic_robust(sigma_val, "Sigma (Est. Desvio Padrão Resíduos - Calculado do SDF)"
    } else {
        print_diagnostic_robust(NULL, "Sigma (Est. Desvio Padrão Resíduos)", "Dispersão típica dos
    }
} else {
     print_diagnostic_robust(sigma_val, "Sigma (Est. Desvio Padrão Resíduos)", "Dispersão típica dos
}

# RSS (Residual Sum of Squares) - Geralmente disponível em diagnostics
print_diagnostic_robust(diagnostics$RSS, "RSS (Residual Sum of Squares)", "Soma dos resíduos quadrad

# R2 Global (Calculado manualmente na escala log1p, como feito no script original)
# Verifica se as colunas 'y' (observado) e 'yhat' (previsto) existem no SDF do GWR
if ("y" %in% names(gwr_resultado_lista$SDF) && "yhat" %in% names(gwr_resultado_lista$SDF)) {
     y_obs <- gwr_resultado_lista$SDF$y
     y_hat <- gwr_resultado_lista$SDF$yhat
     # Remove NAs para calcular TSS e RSS
     valid_indices <- !is.na(y_obs) & !is.na(y_hat)
     y_obs_valid <- y_obs[valid_indices]
     y_hat_valid <- y_hat[valid_indices]

     if (length(y_obs_valid) > 1) { # Certifica-se de que há dados suficientes após remover NAs
         tss <- sum((y_obs_valid - mean(y_obs_valid))^2)
         rss_calc <- sum((y_obs_valid - y_hat_valid)^2)

         # Evita divisão por zero se TSS for zero (ocorre se todos os y_obs forem iguais)
         if (tss > 1e-9) { # Usa uma pequena tolerância para zero
             r2_global_calc <- 1 - (rss_calc / tss)
             cat("\nR2 Global (Calculado - escala transformada):\n")
             print_diagnostic_robust(r2_global_calc, "   R2 Global", "Proporção da variância da VD

             # Cálculo R2 Ajustado (requer EDF ou ENP válidos)
```

```
            r2_ajustado_calc <- NA_real_ # Inicializa com NA
            n_obs_valid <- length(y_obs_valid)
            k_eff <- if (!is.null(diagnostics$edf) && is.finite(diagnostics$edf)) {
                        diagnostics$edf
                    } else if (!is.null(enp_val) && is.finite(enp_val)) {
                        enp_val
                    } else {
                        NULL # Nenhum valor efetivo de parâmetros encontrado
                    }

            if (!is.null(k_eff) && n_obs_valid > k_eff + 1) { # Certifica-se de que tem observaçõe
                 r2_ajustado_calc <- 1 - ((1 - r2_global_calc) * (n_obs_valid - 1) / (n_obs_valid -
                 print_diagnostic_robust(r2_ajustado_calc, "   R2 Ajustado", "R2 global penalizado
            } else {
                 cat("   R2 Ajustado: Não calculado (EDF/ENP não disponível ou insuficiente observa
            }
        } else {
            cat("   R2 Global: Não calculável (Todos os valores observados de 'y' são constantes).'
            cat("   R2 Ajustado: Não calculável.\n")
        }
    } else {
        cat("R2 Global (Calculado): Dados insuficientes ou com muitos NAs após filtragem para cálcu
        cat("R2 Ajustado: Não calculável.\n")
    }
} else {
    cat("R2 Global (Calculado): Colunas 'y' ou 'yhat' não encontradas no SDF do GWR.\n")
    cat("R2 Ajustado: Não calculável.\n")
}

cat("\n## 3. Resumo dos Coeficientes Locais e Significância (na escala log1p) ##")
cat("\n--------------------------------------------------------------------------\n")

# Nomes das colunas de interesse (Ajustados para log1p - devem coincidir com os nomes criados no Bl
var_indep_formula <- all.vars(formula_gwr)[2] # Ex: "log1p_n_drogas"
coef_col_name <- paste0("coef_", var_indep_formula) # Ex: "coef_log1p_n_drogas"
t_val_col_name <- paste0(var_indep_formula, "_TV") # Ex: "log1p_n_drogas_TV"
intercept_col_name <- "Intercept" # Nome padrão para o intercepto
intercept_tv_col_name <- "Intercept_TV" # Nome padrão para o t-valor do intercepto

# Resumo do Intercepto Local
if (intercept_col_name %in% names(resultados_gwr_poligonos_sf)) {
    cat(paste0("Resumo Estatístico para '", intercept_col_name, " (Intercepto Local)':\n"))
    print(summary(resultados_gwr_poligonos_sf[[intercept_col_name]]))
    cat("  -> Representa o valor esperado da variável dependente (log1p_n_roubos) quando todas as va
} else {
    cat(paste0("Coluna de coeficiente '", intercept_col_name, "' não encontrada nos resultados finai
}

# Resumo do Valor-t do Intercepto
 if (intercept_tv_col_name %in% names(resultados_gwr_poligonos_sf)) {
    cat(paste0("\nResumo Estatístico para '", intercept_tv_col_name, " (Valor-t Intercepto Local)':'
    print(summary(resultados_gwr_poligonos_sf[[intercept_tv_col_name]]))
    cat("  -> Indica a significância estatística do Intercepto local em cada ponto. |t| > ~1.96 sug
```

```
} else {
    cat(paste0("\nColuna de valor-t '", intercept_tv_col_name, "' não encontrada nos resultados fina
}

# Resumo do Coeficiente Local da Variável Independente Principal (log1p_n_drogas)
if (coef_col_name %in% names(resultados_gwr_poligonos_sf)) {
    cat(paste0("\nResumo Estatístico para '", coef_col_name, " (Coeficiente Local de ", var_indep_fo
    print(summary(resultados_gwr_poligonos_sf[[coef_col_name]]))
    cat(paste0("  -> Mede a força e a direção da relação entre '", var_indep_formula, "' e 'log1p_n_
    cat("     **A VARIAÇÃO NESTES VALORES (Min. a Max.) É A EVIDÊNCIA CHAVE DA NÃO ESTACIONARIEDADE
    cat("     Isso justifica o uso do GWR: o efeito da variável preditora não é o mesmo em toda a á

    # Interpretação detalhada dos Coeficientes
    mean_coef <- mean(resultados_gwr_poligonos_sf[[coef_col_name]], na.rm = TRUE)
    sd_coef <- sd(resultados_gwr_poligonos_sf[[coef_col_name]], na.rm = TRUE)
    cat(paste0("\n     - Média dos coeficientes locais: ", round(mean_coef, 4), "\n"))
    cat(paste0("     - Desvio Padrão dos coeficientes locais: ", round(sd_coef, 4), "\n"))
    total_valid_coef <- sum(!is.na(resultados_gwr_poligonos_sf[[coef_col_name]]))
    if (total_valid_coef > 0) {
        prop_positivo <- sum(resultados_gwr_poligonos_sf[[coef_col_name]] > 0, na.rm = TRUE) / tota
        prop_negativo <- sum(resultados_gwr_poligonos_sf[[coef_col_name]] < 0, na.rm = TRUE) / tota
        cat(paste0("     - Proporção de coeficientes positivos: ", round(prop_positivo, 1), " %\n"
        cat(paste0("     - Proporção de coeficientes negativos: ", round(prop_negativo, 1), " %\n"
    } else {
        cat("     Nenhum coeficiente válido para análise de proporção.\n")
    }

} else {
    cat(paste0("Coluna de coeficiente '", coef_col_name, "' não encontrada nos resultados finais (es
}

# Resumo dos Valores-t Locais da Variável Independente Principal (log1p_n_drogas_TV)
if (t_val_col_name %in% names(resultados_gwr_poligonos_sf)) {
    cat(paste0("\nResumo Estatístico para '", t_val_col_name, " (Valor-t Local de ", var_indep_formu
    print(summary(resultados_gwr_poligonos_sf[[t_val_col_name]]))
    cat(paste0("  -> Indica a significância estatística do coeficiente local de '", var_indep_formul
    cat("     Valores |t| maiores sugerem maior confiança de que o coeficiente local é diferente de
    cat("     |t| > ~1.96 sugere significância estatística a p < 0.05 (bicaudal, aproximação).\n")

    # Análise de Significância Detalhada baseada em limites de valor-t
    total_valid_t <- sum(!is.na(resultados_gwr_poligonos_sf[[t_val_col_name]]))
    if (total_valid_t > 0) {
        sig_5_count <- sum(abs(resultados_gwr_poligonos_sf[[t_val_col_name]]) > 1.96, na.rm = TRUE)
        sig_1_count <- sum(abs(resultados_gwr_poligonos_sf[[t_val_col_name]]) > 2.58, na.rm = TRUE)
        cat(paste0("\n     - Número de localidades significantes a 5% (|t| > 1.96): ", sig_5_count
        cat(paste0("     - Número de localidades significantes a 1% (|t| > 2.58): ", sig_1_count, "
        cat(paste0("     - Número de localidades NÃO significantes a 5%: ", total_valid_t - sig_5_c
    } else {
        cat("     Nenhum valor-t válido para análise de significância.\n")
    }

} else {
    cat(paste0("\nColuna de valor-t '", t_val_col_name, "' não encontrada nos resultados finais (esp
```

```r
}

# Resumo do R2 Local
r2_col_name <- "Local_R2"
if (r2_col_name %in% names(resultados_gwr_poligonos_sf)) {
    cat(paste0("\n## 4. Resumo do R2 Local (Poder Explicativo Variável) ##\n"))
    cat("-----------------------------------------------------\n")
    cat(paste0("Resumo Estatístico para '", r2_col_name, "':\n"))
    print(summary(resultados_gwr_poligonos_sf[[r2_col_name]]))
    cat("  -> Indica a proporção da variação local na variável dependente (log1p_n_roubos) que é exp
    cat("     Uma alta variação (Min. a Max.) no R2 Local confirma que o poder explicativo do modelo

    # Interpretação detalhada do R2 Local
    mean_r2 <- mean(resultados_gwr_poligonos_sf[[r2_col_name]], na.rm = TRUE)
    sd_r2 <- sd(resultados_gwr_poligonos_sf[[r2_col_name]], na.rm = TRUE)
     cat(paste0("\n     - Média do R2 Local: ", round(mean_r2, 3), "\n"))
     cat(paste0("     - Desvio Padrão do R2 Local: ", round(sd_r2, 3), "\n"))
     # Coeficiente de Variação do R2 Local (se a média > uma pequena tolerância para evitar divisão
     if (mean_r2 > 1e-9) {
        cv_r2 <- (sd_r2 / mean_r2) * 100
        cat(paste0("     - Coeficiente de Variação do R2 Local: ", round(cv_r2, 1), " % (Um CV alto
     } else {
         cat("     - Coeficiente de Variação do R2 Local: Não aplicável (Média do R2 é zero ou negat
     }

    # Distribuição do R2 Local por classes predefinidas
    r2_classes <- cut(resultados_gwr_poligonos_sf[[r2_col_name]],
                      breaks = c(-Inf, 0.1, 0.3, 0.5, 0.7, 1), # Classes: <0.1, 0.1-0.3, 0.3-0.5, 0
                      labels = c("<0.1 (Muito Baixo)", "0.1-0.3 (Baixo)", "0.3-0.5 (Moderado)",
                                 "0.5-0.7 (Alto)", ">0.7 (Muito Alto)"),
                      include.lowest = TRUE, right = TRUE) # include.lowest=TRUE inclui o 0, right=
    cat("\n     Distribuição do R2 Local por classes:\n")
    print(table(r2_classes, useNA = "ifany")) # useNA = "ifany" para incluir contagem de NAs, se ho
    cat("\n")

} else {
    cat(paste0("Coluna '", r2_col_name, "' não encontrada nos resultados finais (esperado nome: ", r
}

cat("\n## 5. Análise de Resíduos ##")
cat("\n---------------------------\n")

residual_col_name <- "residual"
if (residual_col_name %in% names(resultados_gwr_poligonos_sf)) {
    cat(paste0("Resumo Estatístico para '", residual_col_name, "':\n"))
    print(summary(resultados_gwr_poligonos_sf[[residual_col_name]]))
    cat("  -> As diferenças entre os valores observados (log1p_n_roubos) e os valores previstos (yha
    cat("     Resíduos indicam a porção da variável dependente que não foi explicada pelo modelo.\n

    # Estatísticas Adicionais dos Resíduos (Média, DP, Assimetria, Curtose)
    mean_res <- mean(resultados_gwr_poligonos_sf[[residual_col_name]], na.rm = TRUE)
    sd_res <- sd(resultados_gwr_poligonos_sf[[residual_col_name]], na.rm = TRUE)
    cat(paste0("\n     - Média dos resíduos: ", round(mean_res, 6), " (Idealmente próximo de zero)\n
```

```r
cat(paste0("     - Desvio Padrão dos resíduos: ", round(sd_res, 4), "\n"))

# Verifica se o pacote e1071 está carregado para Assimetria e Curtose
if (requireNamespace("e1071", quietly = TRUE)) {
    skew_res <- e1071::skewness(resultados_gwr_poligonos_sf[[residual_col_name]], na.rm = TRUE)
    kurt_res <- e1071::kurtosis(resultados_gwr_poligonos_sf[[residual_col_name]], na.rm = TRUE)
    cat(paste0("     - Assimetria (Skewness): ", round(skew_res, 4), " (Mede a simetria da dist
    cat(paste0("     - Curtose (Kurtosis): ", round(kurt_res, 4), " (Mede o 'achatamento'/'cauc
} else {
    cat("     Instale o pacote 'e1071' para calcular Assimetria e Curtose: install.packages('e
}

# Teste de Normalidade dos Resíduos (Shapiro-Wilk)
# Testa apenas se houver dados residuais válidos
valid_residuals <- resultados_gwr_poligonos_sf[[residual_col_name]][!is.na(resultados_gwr_poligo
if (length(valid_residuals) > 3) { # Shapiro-Wilk requer pelo menos 4 dados não NA
    shapiro_test <- tryCatch({
        # Shapiro-Wilk limitado a 5000 observações para performance. Amostra se necessário.
        res_sample <- if (length(valid_residuals) > 5000) {
                          sample(valid_residuals, 5000, replace = FALSE)
                      } else {
                          valid_residuals
                      }
        shapiro.test(res_sample)
    }, error = function(e) {
        cat("\n     Erro ao executar o Teste de Shapiro-Wilk: ", e$message, "\n")
        NULL # Retorna NULL em caso de erro
    })

    if (!is.null(shapiro_test)) {
        cat("\n     Teste de Normalidade dos Resíduos (Shapiro-Wilk):\n")
        cat(paste0("     - Estatística W: ", round(shapiro_test$statistic, 4), "\n"))
        cat(paste0("     - p-valor: ", format(shapiro_test$p.value, scientific = TRUE), "\n"))
        if (shapiro_test$p.value < 0.05) {
            cat("     - Conclusão: Resíduos NÃO seguem distribuição normal (p < 0.05).\n")
            cat("       -> **Implicação:** A não normalidade dos resíduos é uma violação de um
            cat("       -> Isso pode afetar a validade dos testes de significância baseados no
            cat("       -> **Sugestão:** Considere usar Generalized Geographically Weighted Reg
        } else {
            cat("     - Conclusão: Não há evidência contra a normalidade dos resíduos (p >= 0.0
        }
    } else {
         cat("\n     Teste de Shapiro-Wilk não pôde ser realizado.\n")
    }
} else {
    cat("\n     Dados residuais insuficientes ou inválidos para realizar o Teste de Normalidade
}

# Plotagem de Resíduos (Histograma e Q-Q Plot) - GERA GRÁFICOS, NÃO TEXTO. Comentei a chamada d
# Para ver os gráficos, remova o '#' das linhas abaixo no seu script R.
# cat("\n     -> Verifique o Histograma e o Q-Q Plot dos Resíduos (gerados separadamente) para
# hist(resultados_gwr_poligonos_sf[[residual_col_name]], main = "Distribuição dos Resíduos do G
# qqnorm(resultados_gwr_poligonos_sf[[residual_col_name]], main = "Q-Q Plot dos Resíduos")
```

```r
    # qqline(resultados_gwr_poligonos_sf[[residual_col_name]], col = "red")

} else {
    cat(paste0("Coluna '", residual_col_name, "' não encontrada nos resultados finais. Análise de re
}

cat("\n## 6. Autocorrelação Espacial dos Resíduos (Teste I de Moran) ##")
cat("\n------------------------------------------------------------\n")

# Verifica se a coluna de resíduos existe e se há observações suficientes para o Moran
if (residual_col_name %in% names(resultados_gwr_poligonos_sf) && nrow(resultados_gwr_poligonos_sf) >
    tryCatch({
        # --- Criação da Matriz de Vizinhança e Lista de Pesos ---
        # Utiliza vizinhança por adjacência (rainha), mais apropriada para polígonos.
        # É CRUCIAL que resultados_gwr_poligonos_sf seja um objeto espacial (sf ou sp).
        if (!inherits(resultados_gwr_poligonos_sf, "sf") && !inherits(resultados_gwr_poligonos_sf, "
            stop("O objeto de resultados ('resultados_gwr_poligonos_sf') não é um objeto espacial v
        }

        # Converte para objeto 'sp' se for 'sf', pois 'poly2nb' ainda pode preferir 'sp'
        resultados_sp <- as(resultados_gwr_poligonos_sf, "Spatial")

        # Cria a matriz de vizinhança por adjacência (queen = TRUE considera cantos)
        nb_residuos <- poly2nb(resultados_sp, queen = TRUE)

        # Verifica se há polígonos isolados (sem vizinhos) e lida com eles (zero.policy=TRUE)
        # Uma contagem de vizinhos 0 indica um polígono isolado.
        isolated <- which(card(nb_residuos) == 0)
        if (length(isolated) > 0) {
            cat(paste0("     AVISO: Foram encontrados ", length(isolated), " polígono(s) isolado(s)
            cat("     Estes polígonos serão excluídos do cálculo da lista de pesos (zero.policy=TRU
        }

        # Converte para lista de pesos espaciais (style="W" - row-standardized é padrão e comum)
        listw_residuos <- nb2listw(nb_residuos, style = "W", zero.policy = TRUE) # zero.policy=TRUE

        # --- Executa o Teste I de Moran ---
        # 'alternative = "greater"' testa autocorrelação espacial positiva (mais comum em resíduos
        moran_test <- moran.test(resultados_gwr_poligonos_sf[[residual_col_name]], listw_residuos, a

        cat("\n     Teste I de Moran para os resíduos (baseado em vizinhança por adjacência):\n")
        cat(paste0("       - Estatística I de Moran: ", round(moran_test$estimate[1], 4), "\n"))
        cat(paste0("       - Expectativa sob H0 (aleatoriedade espacial): ", round(moran_test$estima
        cat(paste0("       - Variância: ", round(moran_test$estimate[3], 6), "\n"))
        cat(paste0("       - Desvio padrão: ", round(sqrt(moran_test$estimate[3]), 4), "\n"))
        cat(paste0("       - p-valor (alternativa: maior que a expectativa): ", format(moran_test$p

        # --- Interpretação do Resultado do Teste de Moran ---
        if (moran_test$p.value < 0.05) {
            cat("\n     --> **AVISO CRÍTICO:** Foi detectada AUTOCORRELAÇÃO ESPACIAL POSITIVA e esta
            cat("         Isso é um indicador forte de que o modelo GWR atual NÃO capturou completam
            cat("         Resíduos em localidades próximas tendem a ser semelhantes (altos resíduos
            cat("         **Possíveis causas e soluções:**\n")
```

```
                cat("         - **Variáveis Omitidas:** Fatores espaciais importantes que influenciam os
                cat("         - **Forma Funcional:** A relação entre as variáveis pode ser mais complexa
                cat("         - **Necessidade de Modelagem de Erro Espacial:** Pode ser apropriado usar
            } else {
                cat("\n     --> Nenhuma autocorrelação espacial significativa detectada nos resíduos (p
                cat("         Isso sugere que o modelo GWR parece ter capturado a maior parte da estrutu
            }

        }, error = function(e) {
            # Reporta erros específicos que podem ocorrer no cálculo do Moran
            cat("\nERRO INESPERADO ao calcular ou reportar o Teste I de Moran para os resíduos:\n")
            cat(paste("     Mensagem de erro:", e$message, "\n"))
            cat("     Verifique se o objeto espacial final 'resultados_gwr_poligonos_sf' é válido, se o
        })
    } else {
        # Informa por que o Moran não foi calculado
        cat("\nNão foi possível calcular o Teste I de Moran para os resíduos.\n")
        if (!(residual_col_name %in% names(resultados_gwr_poligonos_sf))) {
            cat(paste0("  - A coluna '", residual_col_name, "' (resíduos) não foi encontrada nos resulta
        }
        if (nrow(resultados_gwr_poligonos_sf) < 30) {
             cat(paste0("  - Número insuficiente de observações (", nrow(resultados_gwr_poligonos_sf),
        }
    }




} # Fim do if(exists)
```

```
##
## ## 1. Informações Gerais do Modelo GWR ##
## -------------------------------------
## Fórmula do modelo: log1p_n_roubos ~ log1p_n_drogas
## Tipo de kernel: gaussian
## Método de seleção de largura de banda:
## Largura de banda ótima (Adaptativa): 18 vizinhos
## Número de observações (pontos de regressão): 406
## CRS Final Projetado Utilizado: EPSG:31983
##
## ## 2. Diagnósticos Globais do Modelo GWR ##
## ----------------------------------------
## AIC: 881.229
##  -> Métrica de qualidade de ajuste, penaliza complexidade. Menor é melhor.
## AICc: 911.9409
##  -> AIC corrigido para amostras pequenas. Mais confiável em GWR.
## BIC: 600.4164
##  -> Similar ao AIC/AICc, penaliza mais a complexidade. Favorece modelos mais simples.
## ENP (Effective Number of Parameters): Não disponível ou não numérico/finito (Valor: NULL).
##  -> Mede a complexidade 'efetiva' do modelo GWR. (Não calculado/Reportado)
## EDF (Effective Degrees of Freedom): 370.3623
##  -> Graus de liberdade efetivos. Relacionado ao ENP.
## Sigma (Est. Desvio Padrão Resíduos - Calculado do SDF): 0.6954
```

```
##  -> Dispersão típica dos resíduos. Menor é melhor.
## RSS (Residual Sum of Squares): 195.8601
##  -> Soma dos resíduos quadrados (na escala transformada). Menor é melhor.
##
## R2 Global (Calculado - escala transformada):
##    R2 Global: 0.5059
##  -> Proporção da variância da VD (transformada) explicada pelo modelo.
##    R2 Ajustado: -4.7775
##  -> R2 global penalizado pela complexidade (EDF/ENP).
##
## ## 3. Resumo dos Coeficientes Locais e Significância (na escala log1p) ##
## ------------------------------------------------------------------
## Resumo Estatístico para 'Intercept (Intercepto Local)':
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.64426 -1.21255 -0.26722 -0.67367  0.08167  0.57886
##   -> Representa o valor esperado da variável dependente (log1p_n_roubos) quando todas as variáveis in
##
## Resumo Estatístico para 'Intercept_TV (Valor-t Intercepto Local)':
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -8.1480 -3.2234 -0.8166 -1.6113  0.2879  1.6091
##   -> Indica a significância estatística do Intercepto local em cada ponto. |t| > ~1.96 sugere signifi
##
## Resumo Estatístico para 'coef_log1p_n_drogas (Coeficiente Local de log1p_n_drogas)':
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.05208 0.19378 0.25260 0.35453 0.48359 0.99487
##   -> Mede a força e a direção da relação entre 'log1p_n_drogas' e 'log1p_n_roubos' em cada localidade
##      **A VARIAÇÃO NESTES VALORES (Min. a Max.) É A EVIDÊNCIA CHAVE DA NÃO ESTACIONARIEDADE ESPACIAL.*
##      Isso justifica o uso do GWR: o efeito da variável preditora não é o mesmo em toda a área de estu
##
##      - Média dos coeficientes locais: 0.3545
##      - Desvio Padrão dos coeficientes locais: 0.2314
##      - Proporção de coeficientes positivos: 100 %
##      - Proporção de coeficientes negativos: 0 %
##
## Resumo Estatístico para 'log1p_n_drogas_TV (Valor-t Local de log1p_n_drogas)':
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4926  2.4993  3.2024  4.4190  6.1797 11.8244
##   -> Indica a significância estatística do coeficiente local de 'log1p_n_drogas' em cada localidade.
##      Valores |t| maiores sugerem maior confiança de que o coeficiente local é diferente de zero.
##      |t| > ~1.96 sugere significância estatística a p < 0.05 (bicaudal, aproximação).
##
##      - Número de localidades significantes a 5% (|t| > 1.96): 368 (90.6 %)
##      - Número de localidades significantes a 1% (|t| > 2.58): 295 (72.7 %)
##      - Número de localidades NÃO significantes a 5%: 38 (9.4 %)
##
## ## 4. Resumo do R2 Local (Poder Explicativo Variável) ##
## ------------------------------------------------------
## Resumo Estatístico para 'Local_R2':
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1788  0.2551  0.3980  0.4171  0.5398  0.8013
##   -> Indica a proporção da variação local na variável dependente (log1p_n_roubos) que é explicada pel
##      Uma alta variação (Min. a Max.) no R2 Local confirma que o poder explicativo do modelo difere ne
##
##      - Média do R2 Local: 0.417
```

```
##      - Desvio Padrão do R2 Local: 0.173
##      - Coeficiente de Variação do R2 Local: 41.6 % (Um CV alto indica grande variabilidade na capaci
##
##      Distribuição do R2 Local por classes:
## r2_classes
## <0.1 (Muito Baixo)    0.1-0.3 (Baixo) 0.3-0.5 (Moderado)     0.5-0.7 (Alto)
##                  0                138                141                 97
##  >0.7 (Muito Alto)
##                 30
##
##
## ## 5. Análise de Resíduos ##
## ---------------------------
## Resumo Estatístico para 'residual':
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.924125 -0.548303 -0.048512 -0.004367  0.411694  2.588215
##   -> As diferenças entre os valores observados (log1p_n_roubos) e os valores previstos (yhat) pelo m
##      Resíduos indicam a porção da variável dependente que não foi explicada pelo modelo.
##
##      - Média dos resíduos: -0.004367 (Idealmente próximo de zero)
##      - Desvio Padrão dos resíduos: 0.6954
##      - Assimetria (Skewness): 0.4607 (Mede a simetria da distribuição. 0 indica simetria)
##      - Curtose (Kurtosis): 1.1451 (Mede o 'achatamento'/'caudas' da distribuição. 0 para Normal (Fis
##
##      Teste de Normalidade dos Resíduos (Shapiro-Wilk):
##      - Estatística W: 0.9717
##      - p-valor: 4.326061e-07
##      - Conclusão: Resíduos NÃO seguem distribuição normal (p < 0.05).
##        -> **Implicação:** A não normalidade dos resíduos é uma violação de um pressuposto do modelo
##        -> Isso pode afetar a validade dos testes de significância baseados no pressuposto de normali
##        -> **Sugestão:** Considere usar Generalized Geographically Weighted Regression (GGWR) com uma
##
## ## 6. Autocorrelação Espacial dos Resíduos (Teste I de Moran) ##
## ------------------------------------------------------------
##
##      Teste I de Moran para os resíduos (baseado em vizinhança por adjacência):
##        - Estatística I de Moran: 0.102
##        - Expectativa sob H0 (aleatoriedade espacial): -0.0025
##        - Variância: 0.000935
##        - Desvio padrão: 0.0306
##        - p-valor (alternativa: maior que a expectativa): 3.17255e-04
##
##      --> **AVISO CRÍTICO:** Foi detectada AUTOCORRELAÇÃO ESPACIAL POSITIVA e estatisticamente SIGNIF
##          Isso é um indicador forte de que o modelo GWR atual NÃO capturou completamente a estrutura
##          Resíduos em localidades próximas tendem a ser semelhantes (altos resíduos perto de altos re
##          **Possíveis causas e soluções:**
##          - **Variáveis Omitidas:** Fatores espaciais importantes que influenciam os roubos e/ou a re
##          - **Forma Funcional:** A relação entre as variáveis pode ser mais complexa do que a modelad
##          - **Necessidade de Modelagem de Erro Espacial:** Pode ser apropriado usar modelos que inclu
```

```
# --- Bloco 11: Salvar Resultados Finais em GeoPackage ---
cat("\n\n--- Bloco 11: Salvar Resultados Finais em GeoPackage ---\n")
```

```
##
```

```
##
## --- Bloco 11: Salvar Resultados Finais em GeoPackage ---

# Verificar se a biblioteca 'sf' está carregada, pois é necessária para st_write.
if (!requireNamespace("sf", quietly = TRUE)) {
    cat("AVISO: O pacote 'sf' não está instalado. Não é possível salvar em GeoPackage.\n")
    cat("Por favor, instale o pacote com: install.packages('sf')\n")
} else if (!exists("resultados_gwr_poligonos_sf") || is.null(resultados_gwr_poligonos_sf) || nrow(resul
    cat("AVISO: O objeto 'resultados_gwr_poligonos_sf' não foi encontrado ou está vazio.\n")
    cat("Nenhum resultado para salvar em GeoPackage. Verifique as etapas anteriores do script.\n")
} else {
    # Definir o caminho completo e o nome do arquivo GeoPackage de saída
    caminho_diretorio_saida <- "C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda/
    nome_base_arquivo <- "resultados_analise_gwr_final.gpkg" # Você pode alterar o nome do arquivo se d
    nome_arquivo_gpkg_completo <- file.path(caminho_diretorio_saida, nome_base_arquivo)

    nome_camada_gpkg <- "gwr_coeficientes_locais" # Nome da camada dentro do GeoPackage

    cat(paste("Tentando salvar os resultados GWR no arquivo GeoPackage em:", nome_arquivo_gpkg_completo
    cat(paste("Nome da camada a ser criada:", nome_camada_gpkg, "\n"))

    # Opcional: Verificar se o diretório de saída existe.
    # Se não existir, você pode querer criá-lo com dir.create(caminho_diretorio_saida, recursive = TRUE
    # ou simplesmente alertar o usuário. Por ora, vamos assumir que ele existe.
    if (!dir.exists(caminho_diretorio_saida)) {
        cat(paste("ALERTA: O diretório de saída especificado NÃO EXISTE:", caminho_diretorio_saida, "\n
        cat("Por favor, crie o diretório manualmente ou ajuste o caminho no script.\n")
        # Você pode optar por parar o script aqui se o diretório for crucial.
        # stop("Diretório de saída não encontrado.")
    }

    tryCatch({
        # Salvar o objeto sf (SpatialPolygonsDataFrame com resultados GWR) em um arquivo GeoPackage.
        sf::st_write(obj = resultados_gwr_poligonos_sf,
                     dsn = nome_arquivo_gpkg_completo, # Usar o caminho completo
                     layer = nome_camada_gpkg,
                     driver = "GPKG",        # Especifica o driver para GeoPackage
                     delete_layer = TRUE,    # Sobrescreve a camada se já existir
                                             # Use delete_dsn = TRUE para sobrescrever o arquivo inteir
                     quiet = FALSE)          # Mostra mensagens do processo

        cat(paste("\nResultados GWR salvos com sucesso em:", nome_arquivo_gpkg_completo, "como camada:"
        cat("Você pode abrir este arquivo em um software GIS (QGIS, ArcGIS, etc.) para visualizar os ma

    }, error = function(e) {
        cat("\nERRO AO SALVAR EM GEOPACKAGE:\n")
        cat(paste("Mensagem de erro:", e$message, "\n"))
        cat(paste("Verifique se o caminho do diretório está correto, se o diretório existe e se você te
        cat("Certifique-se também de que o pacote 'sf' está funcionando corretamente.\n")
    })
}
```

```
## Tentando salvar os resultados GWR no arquivo GeoPackage em: C:/Users/Rodrigo - H2R/OneDrive - Conhec
## Nome da camada a ser criada: gwr_coeficientes_locais
```

```
## Deleting layer ‘gwr_coeficientes_locais’ using driver ‘GPKG’
## Writing layer ‘gwr_coeficientes_locais’ to data source
##   ‘C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marketing Ltda/Documentos/mba/mba_arrumac
## Writing 406 features with 14 fields and geometry type Polygon.
##
## Resultados GWR salvos com sucesso em: C:/Users/Rodrigo - H2R/OneDrive - Conhecimento e Ação em Marke
## Você pode abrir este arquivo em um software GIS (QGIS, ArcGIS, etc.) para visualizar os mapas de coe
```

```
cat("\n--- Fim do Bloco 11: Salvar Resultados ---\n")
```

```
##
## --- Fim do Bloco 11: Salvar Resultados ---
```

```
# Dentro do Bloco 10, na seção "## 2. Diagnósticos Globais do Modelo GWR ##"

if (!is.null(gwr_resultado_lista$GW.diagnostic)) {
    diagnostics <- gwr_resultado_lista$GW.diagnostic

    # Função auxiliar para verificar e imprimir diagnósticos
    print_diagnostic <- function(value, name, description) {
        if (!is.null(value) && is.numeric(value) && is.finite(value)) { # is.finite() verifica NA, NaN,
            cat(paste0(name, ": ", round(value, 2), "\n"))
            cat(paste0("  -> ", description, "\n"))
        } else {
            cat(paste0(name, ": Não disponível ou não numérico (Valor: ", as.character(value), ").\n"))
        }
    }

    # AICc
    print_diagnostic(diagnostics$AICc, "AICc (Akaike Information Criterion Corrigido)", "O AICc é uma m

    # ENP
    print_diagnostic(diagnostics$ENP, "ENP (Effective Number of Parameters)", "O ENP reflete a complexi

    # EDF
    print_diagnostic(diagnostics$edf, "EDF (Effective Degrees of Freedom)", "Similar ao ENP, o EDF é us

    # Sigma
    print_diagnostic(diagnostics$sigma, "Sigma (Estimativa do Desvio Padrão dos Resíduos)", "O sigma re

    # RSS
    print_diagnostic(diagnostics$RSS, "RSS (Residual Sum of Squares)", "O RSS mede a soma das diferença

    # Cálculo do R² Global (manter como está, pois já tem uma verificação if)
    if ("y" %in% names(gwr_resultado_lista$SDF) && "yhat" %in% names(gwr_resultado_lista$SDF)) {
        # ... (código do R² global) ...
    } else {
        cat("R² Global (Calculado): Não foi possível calcular pois 'y' ou 'yhat' não estão presentes no
    }
} else {
    cat("Diagnósticos globais (gwr_resultado_lista$GW.diagnostic) não disponíveis ou a estrutura está v
}
```

```
## AICc (Akaike Information Criterion Corrigido): 911.94
```

```
##   -> O AICc é uma métrica de qualidade de ajuste que penaliza modelos com mais parâmetros...
## ENP (Effective Number of Parameters): Não disponível ou não numérico (Valor: ).
## EDF (Effective Degrees of Freedom): 370.36
##   -> Similar ao ENP, o EDF é usado em cálculos estatísticos e representa os graus de liberdade efeti
## Sigma (Estimativa do Desvio Padrão dos Resíduos): Não disponível ou não numérico (Valor: ).
## RSS (Residual Sum of Squares): 195.86
##   -> O RSS mede a soma das diferenças quadráticas entre os valores observados e previstos, indicando

## NULL
```