

Introduction to SLGP Package

Athénaïs Gautier

This vignette provides a practical introduction to Spatial Logistic Gaussian Process (SLGP) modeling, demonstrating its implementation and application..

1 Dataset

We illustrate the model's capabilities using the Boston Housing dataset Harrison and Rubinfeld [1978], a well-known benchmark in statistical modeling and regression analysis.

For this vignette, we focus on modeling the distribution of median home values (`medv`) as a function of the proportion of pre-1940 owner-occupied units (`age`). This example highlights the ability of SLGPs to capture complex, spatially dependent distributions in data that exhibit heterogeneity and multi-modality.

```
library(dplyr)
# Load the dataset (available in MASS package)
if (!requireNamespace("MASS", quietly = TRUE)) install.packages("MASS")
data("Boston", package = "MASS")
df <- Boston %>%
  mutate(age_bin = cut(age, breaks = seq(0, 100, by = 10), include.lowest = FALSE)) %>%
  group_by(age_bin) %>%
  mutate(age_bin = paste0(age_bin, "\nn=", n())) %>%
  ungroup() %>%
  mutate(age_bin = factor(age_bin,
                          levels = sort(unique(age_bin), decreasing = FALSE))) %>%
  data.frame()

range_response <- c(0, 50) # Can use range(df$medv), or user defined range as we do here
range_x <- c(0, 100) # Can use range(df$age), or user defined range as we do here
```

We represent the data to visualise the relationship between `medv` and `age`.

```
library(ggplot2)
library(ggpubr)
#> Warning: le package 'ggpubr' a été compilé avec la version R 4.4.2
library(viridis)
#> Le chargement a nécessité le package : viridisLite
# Scatterplot: med vs. age
scatter_plot <- ggplot(df, aes(x = age, y = medv)) +
  geom_point(alpha = 0.5, color = "navy") +
  labs(x = "Proportion of owner-occupied units\nbuilt prior to 1940 [AGE, %]",
       y = "Median value of owner-occupied homes [MEDV, k$]",
       title = "Median value vs. age of homes") +
  theme_bw() +
  coord_cartesian(xlim=range_x,
                 ylim=range_response)
```

```
# Histogram: Distribution of med by age bin
hist_plot <- ggplot(df, aes(x = medv)) +
  geom_histogram(mapping=aes(y=after_stat(density)),
    position = "identity", breaks = seq(0, 50, 2.5),
    fill="darkgrey", col="grey50", lwd=0.2, alpha=0.7) +
  geom_rug(sides = "b", color = "navy", alpha = 0.5)+
  facet_wrap(~ age_bin, scales = "free_y", nrow=2) +
  labs(x = "Median value of owner-occupied homes [MEDV, k$]",
    y = "Probability density",
    title = "Histogram of median housing values by AGE group") +
  theme_bw()+
  coord_cartesian(xlim=range_response,
    ylim=c(0, 0.25))
ggarrange(scatter_plot, hist_plot, ncol = 2, nrow = 1,
  widths = c(0.3, 0.7))
```

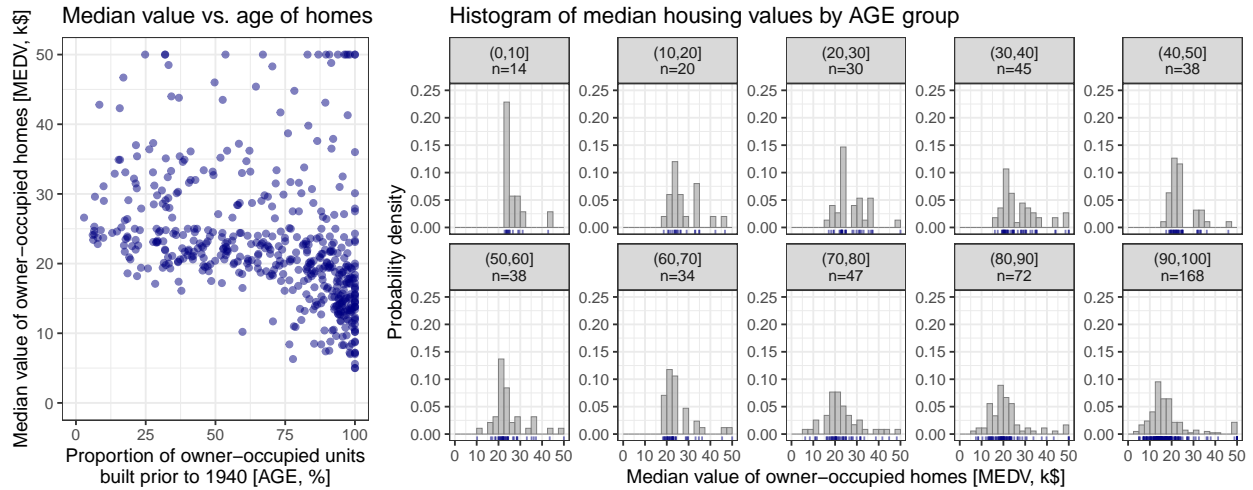


Figure 1: A visual representation of the dependency of the median value of owner-occupied homes on proportion of owner-occupied units constructed before 1940 in the Boston Housing dataset.

```
ggsave("./Figures/scatter.pdf", width=10, height=3.5)
```

We see that there is a general trend where older homes tend to have lower values, with exceptions likely due to survivor bias: older homes that persist tend to be of higher structural quality. This dataset provides an test case for SLGP modeling, offering a compact, one-dimensional covariate space, heterogeneously distributed data, and shifting distributional shapes.

2 SLGP model specifications

To model the distributional changes observed in the Boston Housing dataset, we now introduce the Spatial Logistic Gaussian Process (SLGP) model. SLGPs provide a flexible non-parametric framework for modeling spatially dependent probability densities. By transforming a Gaussian Process (GP) through exponentiation and normalization, SLGPs ensure positivity and integration to one, making them well-suited for density estimation. In this section, we specify the SLGP prior, and visualise its behaviour.

2.1 Prior

The prior in an SLGP represents our initial beliefs about the structure of the data before incorporating observations. It defines a distribution over possible density functions, capturing spatial dependencies while allowing sufficient flexibility. The following code chunk sets up an SLGP prior over $\text{medv} \sim \text{age}$, where medv is modeled as a function of age :

```
library(SLGP)

modelPrior <- slgp(medv~age, # Use a formula to specify predictors VS response
  # Can use medv~. for all variables,
  # Or medv ~ age + var2 + var3 for more variables
  data=df,
  method="none", #Maximum a posteriori estimation scheme
  basisFunctionsUsed = "RFF",
  interpolateBasisFun="WNN", # Will Accelerate inference
  hyperparams = list(lengthscale=c(0.15, 0.15),
    # Applied to normalised data
    # So 0.15 is 15% of the range of values
    sigma2=1),
  # Will be re-selected with sigmaEstimationMethod
  sigmaEstimationMethod = "heuristic",
  # Set to heuristic for numerical stability
  predictorsLower= c(range_x[1]),
  predictorsUpper= c(range_x[2]),
  responseRange= range_response,
  opts_BasisFun = list(nFreq=200,
    MatParam=5/2),
  seed=1)
```

Here:

- The model uses Random Fourier Features (RFF) for the finite-rank latent GP.
- The lengthscale (15% of the normalized range) controls the smoothness of variation. This is selected following our proposed heuristic.
- The heuristic sigma estimation ensures numerical stability.

2.1.1 Looking at several draws of the prior

To understand how the SLGP prior behaves, we generate and visualize random draws from the prior distribution over probability densities of medv at different age values.

```
library(tidyr)
nrep <- 3
set.seed(8)
p <- ncol(modelPrior@coefficients)
modelPrior@coefficients <- matrix(rnorm(n=nrep*p), nrow=nrep)

dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2], 5),
  seq(range_response[1], range_response[2], 101)))
colnames(dfGrid) <- c("age", "medv")
predPrior <- predictSLGP_newNode(SLGPmodel=modelPrior,
  newNodes = dfGrid)
colnames(predPrior) <- c("age", "medv", paste0("Draw from the prior n°", seq(nrep)))
```

```

predPrior <- predPrior%>%
  pivot_longer(-c("age", "medv"))
scale_factor <- 200
ggplot() +
  labs(y = "Proportion of owner-occupied units built prior to 1940 [AGE, %]",
       x = "Median value of owner-occupied homes [MEDV, k$]",
       title = "Samples from the SLGP Prior for the pdfs of MEDV at AGE,
               visualised across slices") +
  theme_bw() +
  geom_ribbon(data=predPrior,
            mapping=aes(x=medv, ymax=scale_factor*value+age,
                       ymin=age, group=-age, fill=age),
            col="grey", alpha=0.9) +
  # geom_point(data=df,
  #            mapping=aes(x = medv, y = age), alpha = 0.5, color = "navy") +
  scale_fill_viridis(option = "plasma",
                    guide = guide_colorbar(nrow = 1,
                                           title = "Indexing variable:
                                           Proportion of owner-occupied units built
                                           prior to 1940",
                                           barheight = unit(2, units = "mm"),
                                           barwidth = unit(55, units = "mm"),
                                           title.position = 'top',
                                           label.position = "bottom",
                                           title.hjust = 0.5)) +

  theme(legend.position = "bottom") +
  coord_flip() +
  facet_grid(.~name)

```

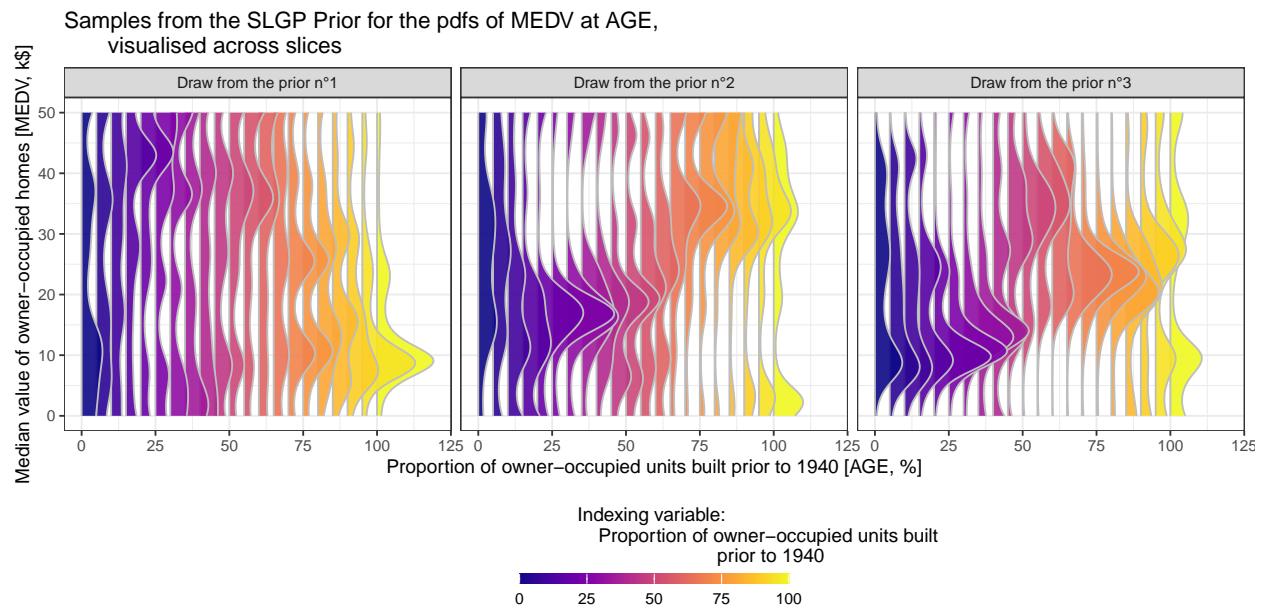


Figure 2: Samples from the SLGP Prior for the pdfs of MEDV at AGE, visualised across slices.

```
ggsave(paste0("./Figures/ribbonsPrior", ".pdf"), width=10, height=5)
```

This figure illustrates how the SLGP prior encodes a distribution over densities for different age values. The wide variability indicates the flexibility of SLGPs.

2.1.2 Assessing whether the flexibility matches that of the data

To assess how well the SLGP prior aligns with the actual data distribution, we can also compare it to histograms of `medv` at selected `age` values. This visualisation helps us evaluate whether the prior has enough flexibility to represent the observed variability in the data.

```
selected_values <- c(20, 50, 95)
gap <- 5
df_filtered <- df %>%
  mutate(interval=findInterval(age, c(0,
                                     selected_values[1]-gap,
                                     selected_values[1]+gap,
                                     selected_values[2]-gap,
                                     selected_values[2]+gap,
                                     selected_values[3]-gap,
                                     selected_values[3]+gap)))%>%
  filter(interval %in% c(2, 4, 6))%>%
  group_by(interval)%>%
  mutate(category = paste0("Age close to ", c("", selected_values[1],
                                              "", selected_values[2],
                                              "", selected_values[3])[interval],
                          "\nn=", n()))
names <- sort(unique(df_filtered$category))
dfGrid <- data.frame(expand.grid(selected_values,
                                seq(range_response[1], range_response[2], 101)))
colnames(dfGrid) <- c("age", "medv")
predPrior <- predictSLGP_newNode(SLGPmodel=modelPrior,
                                newNodes = dfGrid)
colnames(predPrior) <- c("age", "medv", paste0("Draw from the prior n°", seq(nrep)))
predPrior <- predPrior%>%
  pivot_longer(-c("age", "medv"))
predPrior$category <- ifelse(predPrior$age==selected_values[1], names[1],
                             ifelse(predPrior$age==selected_values[2], names[2], names[3]))

ggplot(mapping=aes(x = medv)) +
  geom_histogram(df_filtered,
                mapping=aes(y=after_stat(density)),
                position = "identity", breaks = seq(0, 50, 2.5),
                fill="darkgrey", col="grey50", lwd=0.2, alpha=0.7) +
  geom_rug(data=df_filtered, sides = "b", color = "navy", alpha = 0.5)+
  geom_line(data=predPrior, mapping=aes(y=value, group=name),
            color = "black", lwd=0.1, alpha=0.5)+
  geom_line(data=predPrior, mapping=aes(y=value, group=name, col=name), lwd=1.1)+
  facet_wrap(~ category, scales = "free_y", nrow=1) +
  labs(x = "Median value of owner-occupied homes [k$]",
       y = "Probability density",
       title = "Histogram of median value at bins centered at several 'age' values,
               with width 5\nversus draws from a SLGP prior") +
  theme_bw()+
```

```

theme(legend.position="bottom",
      legend.direction = "horizontal",
      legend.title = element_blank())+
coord_cartesian(xlim=range_response,
                ylim=c(0, 0.25))

```



Figure 3: Samples from the SLGP Prior versus histograms of median value at bins centered at several 'age' values

```

ggsave(paste0("./Figures/histPrior", ".pdf"), width=10, height=5)

```

Since this is a prior distribution, it does not yet incorporate information from the actual data. Therefore, we should not expect it to match the empirical histograms. However, the prior samples display a reasonable level of variability and structure, suggesting that the model would be well-suited for density estimation after incorporating data through posterior inference.

2.1.3 Other priors

To explore how different covariance structures influence the SLGP prior, we compare three additional kernel choices:

- Exponential Kernel: A special case of the Matérn class ($\nu = 1/2$) that results in less smooth realizations.
- Matérn 3/2 Kernel: A compromise between flexibility and smoothness.
- Gaussian Kernel: The limiting case, leading to infinitely differentiable functions.

```

# With exponential kernel
modelPrior2 <- slgp(medv~age,
  data=df,
  method="none",
  basisFunctionsUsed = "RFF",
  interpolateBasisFun="WNN",
  hyperparams = list(lengthscale=c(0.15, 0.15),
    sigma2=1),

```

```

        sigmaEstimationMethod = "heuristic",
        predictorsLower= c(range_x[1]),
        predictorsUpper= c(range_x[2]),
        responseRange= range_response,
        opts_BasisFun = list(nFreq=200,
                             MatParam=1/2),

        seed=1)
# With Matérn 3/2 kernel
modelPrior3 <- slgp(medv~age,
                   data=df,
                   method="none",
                   basisFunctionsUsed = "RFF",
                   interpolateBasisFun="WNN",
                   hyperparams = list(lengthscale=c(0.15, 0.15),
                                       sigma2=1),
                   sigmaEstimationMethod = "heuristic",
                   predictorsLower= c(range_x[1]),
                   predictorsUpper= c(range_x[2]),
                   responseRange= range_response,
                   opts_BasisFun = list(nFreq=200,
                                       MatParam=3/2),

                   seed=1)
# with Gaussian Kernel
modelPrior4 <- slgp(medv~age,
                   data=df,
                   method="none",
                   basisFunctionsUsed = "RFF",
                   interpolateBasisFun="WNN",
                   hyperparams = list(lengthscale=c(0.15, 0.15),
                                       sigma2=1),
                   sigmaEstimationMethod = "heuristic",
                   predictorsLower= c(range_x[1]),
                   predictorsUpper= c(range_x[2]),
                   responseRange= range_response,
                   opts_BasisFun = list(nFreq=200,
                                       MatParam=Inf),

                   seed=1)

```

The figure below visualizes samples from the priors corresponding to each kernel. While the overall behavior remains similar, the choice of kernel influences the smoothness of the density estimates. The exponential kernel exhibits the most variability, while the Gaussian kernel enforces stronger smoothness constraints.

```

nrep <- 1
set.seed(1)
coef <- matrix(rnorm(n=nrep*p), nrow=nrep)
modelPrior2@coefficients <- coef
modelPrior3@coefficients <- coef
modelPrior4@coefficients <- coef

dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2], 5),
                                   seq(range_response[1], range_response[2], 101)))
colnames(dfGrid) <- c("age", "medv")

predPrior <- rbind(predictSLGP_newNode(SLGPmodel=modelPrior2,

```

```

                                newNodes = dfGrid),
predictSLGP_newNode(SLGPmodel=modelPrior3,
                    newNodes = dfGrid),
predictSLGP_newNode(SLGPmodel=modelPrior4,
                    newNodes = dfGrid))
predPrior$Kernel <- c(sapply(seq(3), function(i){rep(i, nrow(dfGrid))}))
colnames(predPrior) <- c("age", "medv", "value", "Kernel")

predPrior$Kernel <- factor(c("Exponential", "Matérn 3/2", "Gaussian")[predPrior$Kernel],
                          levels=c("Exponential", "Matérn 3/2", "Gaussian"))

scale_factor <- 200
ggplot() +
  labs(y = "Proportion of owner-occupied units built prior to 1940 [AGE, %]",
       x = "Median value of owner-occupied homes [MEDV, k$]",
       title = "Samples from SLGP Priors with varying smoothnesses for the pdfs of MEDV at AGE, visuali")
  theme_bw()+
  geom_ribbon(data=predPrior,
            mapping=aes(x=medv, ymax=scale_factor*value+age,
                       ymin=age, group=-age, fill=age),
            col="grey", alpha=0.9)+
  scale_fill_viridis(option = "plasma",
                    guide = guide_colorbar(nrow = 1,
                                           title = "Indexing variable:
Proportion of owner-occupied units built
prior to 1940",
                                           barheight = unit(2, units = "mm"),
                                           barwidth = unit(55, units = "mm"),
                                           title.position = 'top',
                                           label.position = "bottom",
                                           title.hjust = 0.5))+

  theme(legend.position = "bottom")+
  coord_flip()+
  facet_grid(.~Kernel)

```

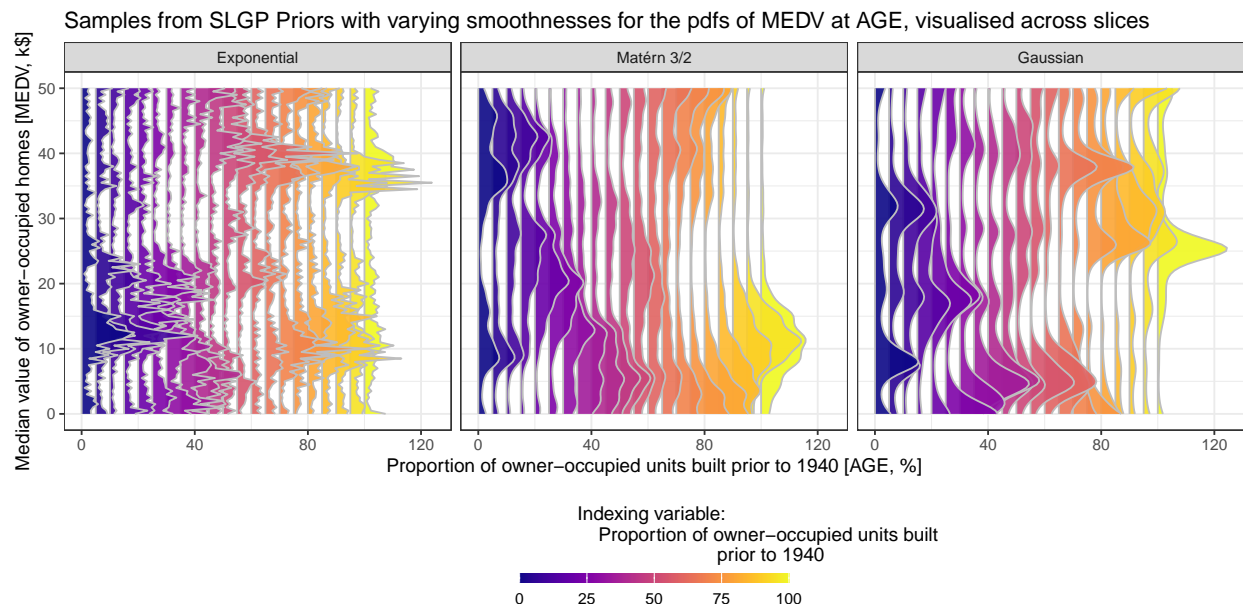



Figure 4: Samples from the other SLGP Priors for the pdfs of MEDV at AGE, visualised across slices.

```
ggsave(paste0("./Figures/ribbonsPriorOthers", ".pdf"), width=10, height=5)
```

2.2 Estimation: Maximum a posteriori estimate

For a fast and computationally efficient estimation, we propose using MAP estimation. MAP delivers a single point estimate by maximizing the posterior distribution. It is the fastest estimation scheme we propose, however MAP does not facilitate uncertainty quantification because it yields a non-probabilistic estimate of the underlying density field, focusing instead on identifying the mode of the posterior distribution.

2.2.1 Performing the estimation

We demonstrate three equivalent ways to train an SLGP model using our package:

- Direct initialization: Computes the basis functions and performs the full estimation from scratch.
- Retraining from another model: Reuses pre-computed basis functions from an existing SLGP model and updates only the coefficients for efficiency.
- Explicit reuse of prior components: Allows manually specifying pre-computed basis functions, offering greater control over the initialization.

```
modelMAP <- slgp(medv~age, # Use a formula to specify predictors VS response
  # Can use medv~. for all variables,
  # Or medv ~ age + var2 + var3 for more variables
  data=df,
  method="MAP", #Maximum a posteriori estimation scheme
  basisFunctionsUsed = "RFF",
  interpolateBasisFun="WNN", # Accelerate inference
  hyperparams = list(lengthscale=c(0.15, 0.15),
    # Applied to normalised data
    # So 0.15 is 15% of the range of values
    sigma2=1),
  # Will be re-selected with sigmaEstimationMethod
```

```

sigmaEstimationMethod = "heuristic",
# Set to heuristic for numerical stability
predictorsLower= c(range_x[1]),
predictorsUpper= c(range_x[2]),
responseRange= range_response,
opts_BasisFun = list(nFreq=200,
                     MatParam=5/2),

seed=1)

# Or equivalent, re-use the same basis functions
# and hyper parameters as in the prior we saw

modelMAP <- retrainSLGP(SLGPmodel=modelPrior,
                       newdata = df,
                       method="MAP")

# Or equivalent, more explicit in the re-using of the elements
# From the SLGP prior

modelMAP <- slgp(medv~age,
                 data=df,
                 method="MAP", #Maximum a posteriori estimation scheme
                 basisFunctionsUsed = "RFF",
                 interpolateBasisFun="WNN", # Accelerate inference
                 hyperparams = modelPrior@hyperparams,
                 sigmaEstimationMethod = "none", # Already selected in the prior
                 predictorsLower= c(range_x[1]),
                 predictorsUpper= c(range_x[2]),
                 responseRange= range_response,
                 opts_BasisFun = modelPrior@opts_BasisFun,
                 BasisFunParam = modelPrior@BasisFunParam,
                 seed=1)

```

2.2.2 Visualising the results

The estimated conditional density function is displayed using two different representations:

- A colormap showing the predicted density of medv as a function of age.
- A ribbon plot, which slices the conditional density at selected age values to provide a clearer view of the estimated distributions.

```

dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2],, 101),
                                seq(range_response[1], range_response[2],, 101)))
colnames(dfGrid) <- c("age", "medv")
pred <- predictSLGP_newNode(SLGPmodel=modelMAP,
                           newNodes = dfGrid)

ggplot() +
  labs(y = "Proportion of owner-occupied units\nbuilt prior to 1940 [%]",
       x = "Median value of owner-occupied homes [k$]",
       title = "Median value vs. age of homes") +
  theme_bw()+
  geom_raster(data=pred,
             mapping=aes(x=age, y=medv, fill=pdf_1))+
  geom_point(data=df,

```

```

mapping=aes(x=age, y=medv), alpha = 0.5,
pch="x", col="grey")+
scale_fill_viridis(option = "viridis",
  guide = guide_colorbar(nrow = 1,
    title = "Probability density of med at age",
    barheight = unit(2, units = "mm"),
    barwidth = unit(55, units = "mm"),
    title.position = 'top',
    label.position = "bottom",
    title.hjust = 0.5))+
theme(legend.position = "bottom")

```

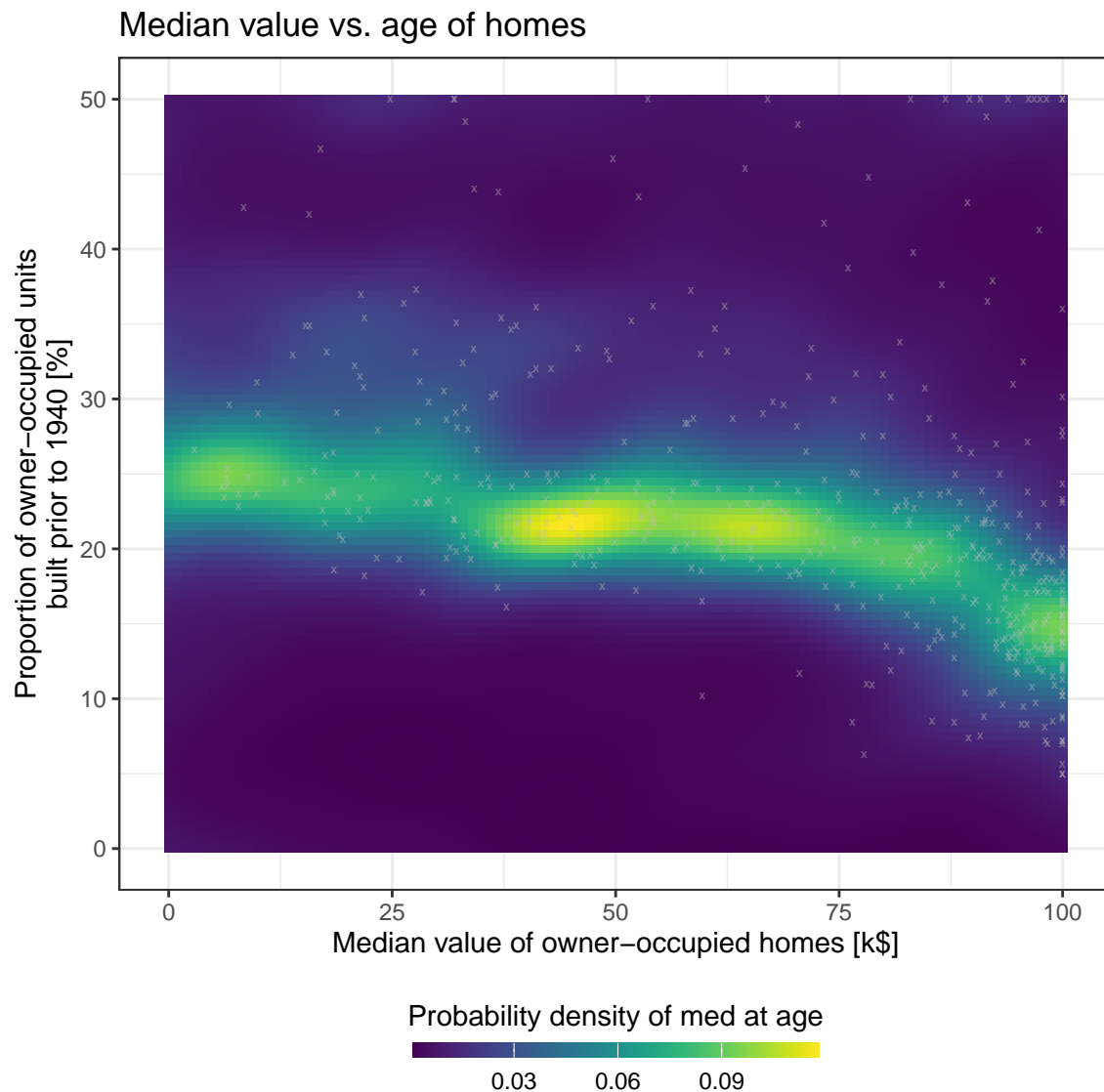


Figure 5: Predictive probability density of medv at age, as predicted by a SLGP.

```

library(viridis)
dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2], 5),
                                   seq(range_response[1], range_response[2],, 101)))
colnames(dfGrid) <- c("age", "medv")
pred <- predictSLGP_newNode(SLGPmodel=modelMAP,
                           newNodes = dfGrid)

scale_factor <- 100
ggplot() +
  labs(y = "Proportion of owner-occupied units\nbuilt prior to 1940 [%]",
       x = "Median value of owner-occupied homes [k$]",
       title = "Median value vs. age of homes") +
  theme_bw()+
  geom_ribbon(data=pred,
            mapping=aes(x=medv, ymax=scale_factor*pdf_1+age,
                       ymin=age, group=-age, fill=age),
            col="grey", alpha=0.9)+
  geom_point(data=df,
            mapping=aes(x = medv, y = age), alpha = 0.5, color = "navy")+
  scale_fill_viridis(option = "plasma",
                    guide = guide_colorbar(nrow = 1,
                                           title = "Indexing variable:
Proportion of owner-occupied units built
prior to 1940",
                                           barheight = unit(2, units = "mm"),
                                           barwidth = unit(55, units = "mm"),
                                           title.position = 'top',
                                           label.position = "bottom",
                                           title.hjust = 0.5))+
  theme(legend.position = "bottom")+
  coord_flip()

```

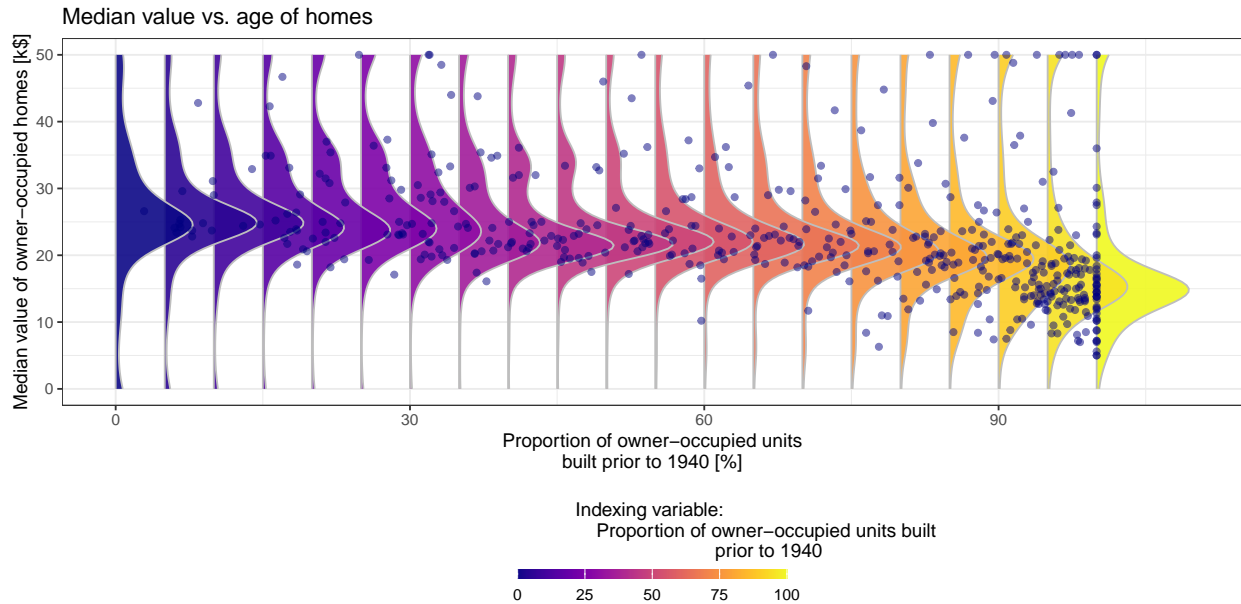


Figure 6: Predictive probability density of medv at age, seen over slices.

```
ggsave(paste0("./Figures/ribbonsMAP", ".pdf"), width=10, height=5)
```

The figure reveals how the SLGP captures the structure in the estimated conditional density of `medv` given `age`. It effectively adapts to variations in skewness as the distribution of home prices shifts across different property ages. Notably, the model also captures the survivor bias we noted earlier by assigning a small but notable probability mass to older units with high values.

2.2.3 Comparing the estimation to data

```
selected_values <- c(20, 50, 95)
gap <- 5
df_filtered <- df %>%
  mutate(interval=findInterval(age, c(0,
    selected_values[1]-gap,
    selected_values[1]+gap,
    selected_values[2]-gap,
    selected_values[2]+gap,
    selected_values[3]-gap,
    selected_values[3]+gap)))%>%
  filter(interval %in% c(2, 4, 6))%>%
  group_by(interval)%>%
  mutate(category = paste0("Age close to ", c("", selected_values[1],
    "", selected_values[2],
    "", selected_values[3])[interval],
    "\nn=", n()))
names <- sort(unique(df_filtered$category))
dfGrid <- data.frame(expand.grid(selected_values,
  seq(range_response[1], range_response[2], 101)))
colnames(dfGrid) <- c("age", "medv")
predMAP <- predictSLGP_newNode(SLGPmodel=modelMAP,
```

```

newNodes = dfGrid)
colnames(predMAP) <- c("age", "medv", "MAP estimator")
predMAP <- predMAP%>%
  pivot_longer(-c("age", "medv"))
predMAP$category <- ifelse(predMAP$age==selected_values[1], names[1],
  ifelse(predMAP$age==selected_values[2], names[2], names[3]))

ggplot(mapping=aes(x = medv)) +
  geom_histogram(df_filtered,
    mapping=aes(y=after_stat(density)),
    position = "identity", breaks = seq(0, 50, 2.5),
    fill="darkgrey", col="grey50", lwd=0.2, alpha=0.7) +
  geom_rug(data=df_filtered, sides = "b", color = "navy", alpha = 0.5)+
  geom_line(data=predMAP, mapping=aes(y=value, group=name),
    color = "black", lwd=0.1, alpha=0.5)+
  geom_line(data=predMAP, mapping=aes(y=value, group=name, col=name), lwd=1.1)+
  facet_wrap(~ category, scales = "free_y", nrow=1) +
  labs(x = "Median value of owner-occupied homes [k$]",
    y = "Probability density",
    title = "Histogram of median value at bins centered at several 'age' values, with width 5\nversus draws from a SLGP prior")
  theme_bw()+
  theme(legend.position="bottom",
    legend.direction = "horizontal",
    legend.title = element_blank())+
  coord_cartesian(xlim=range_response,
    ylim=c(0, 0.25))

```

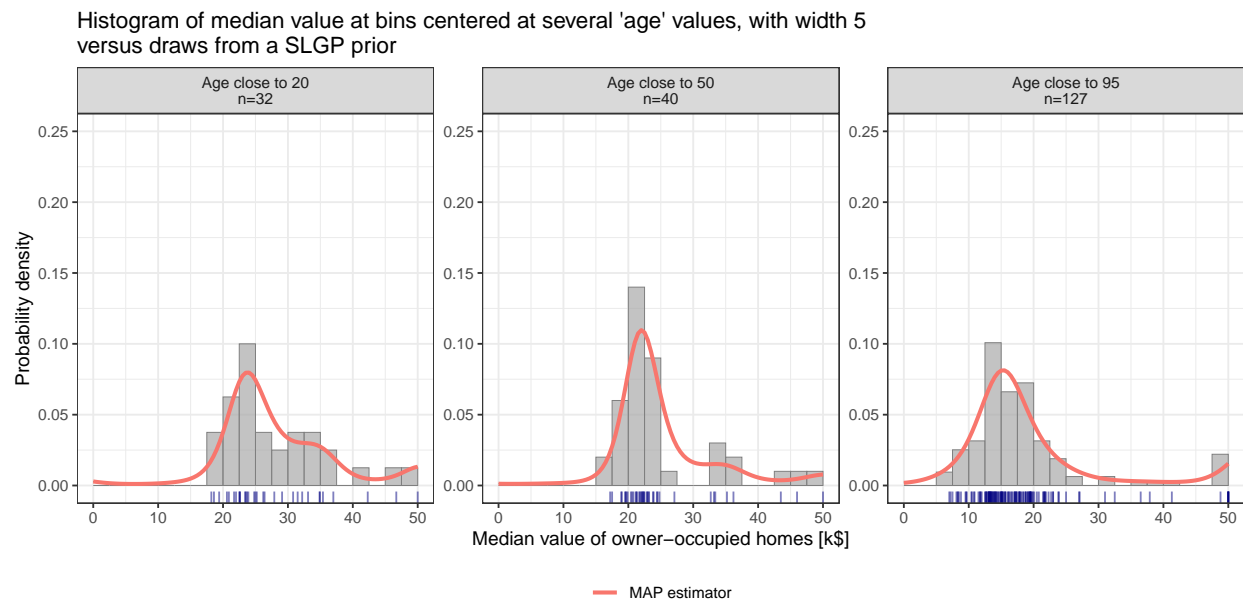


Figure 7: SLGP MAP versus histograms of median value at bins centered at several 'age' values

```

ggsave(paste0("./Figures/histMAP", ".pdf"), width=10, height=5)

```

This other figure directly compares the SLGP MAP estimate to histograms of `medv` at selected `age` values,

illustrating how well the model aligns with the observed data. Despite the small number of replicates, the SLGP estimate follows the empirical distribution, adapting to changes in shape across different age groups. The survivor bias is again visible, as the model assigns some probability mass to higher home values for older properties.

2.2.4 Influence of the lengthscale selected

To assess the impact of lengthscale selection on SLGP estimation, we follow the MAP-based strategy described in the paper. We define a prior over the lengthscales, perform posterior evaluations over a regular grid, and compare optimization results under two different data usage regimes:

- Full dataset: model is trained and evaluated on the full data.
- Partial training: model is trained on 25% of the data and evaluated on the remaining 75%.

Concretely, we specify an Inverse-Gamma prior on each lengthscale parameter.

```
set.seed(1)
newOrder <- sample(seq(nrow(df)))

dinvgamma <- function(x, alpha=4.5, beta=0.35) {
  ifelse(x<=0, 0, (beta^alpha / gamma(alpha)) * x^(-alpha - 1) * exp(-beta / x))
}
plot(dinvgamma, from=0, to=1)
if(file.exists("OptimisingLen.RData")){
  load(file="OptimisingLen.RData")
}else{
  lengthscale_grid <- seq(0.025, 0.5, 0.025)
  df_res <- data.frame(expand.grid(lengthscale_grid, lengthscale_grid))
  colnames(df_res) <- c("l_age", "l_medv")
  df_res$logPostData <- NaN
  df_res$logPostData2 <- NaN
}

df_res$logPrior <- log(dinvgamma(df_res$l_age))+log(dinvgamma(df_res$l_medv))
for(i in which(round(df_res$l_age*100)%5==0&
  round(df_res$l_medv*100)%5==0)){
  l1 <- df_res$l_age[i]
  l2 <- df_res$l_medv[i]
  if(is.na(df_res$logPostData[i])){

    modelMAPtemp <- retrainSLGP(SLGPmodel=modelMAP,
                                newdata = df,
                                method="MAP",
                                hyperparams=list(sigma2=1,
                                                  lengthscale=c(l1, l2)),
                                sigmaEstimationMethod="heuristic")
    df_res$logPostData[i] <- modelMAPtemp$logPostData
    if(sum(!is.na(df_res$logPostData))%10==0){
      save(df_res, file="OptimisingLen.RData")
    }
  }
}
if(is.na(df_res$logPostData2[i])){
```

```

modelMAPtemp <- retrainSLGP(SLGPmodel=modelMAP,
                           newdata = df[newOrder[1:126],
                                           c("age", "medv")],
                           method="MAP",
                           hyperparams=list(sigma2=1,
                                              lengthscale=c(11, 12)),
                           sigmaEstimationMethod="heuristic")
temp <- predictSLGP_newNode(modelMAPtemp,
                           newNodes = df[newOrder[-c(1:126)],
                                           c("age", "medv") ])
df_res$logPostData2[i] <- sum(log(temp$pdf_1))
if(sum(!is.na(df_res$logPostData2))%10==0){
  save(df_res, file="OptimisingLen.RData")
}
}
}
save(df_res, file="OptimisingLen.RData")

```

The following figure displays the resulting posterior landscapes.

```

load(file="OptimisingLen.RData")

min1 <- df_res[which.min(-df_res$logPostData-df_res$logPrior), ]
min2 <- df_res[which.min(-df_res$logPostData2-df_res$logPrior), ]
plot1 <- df_res %>%
  dplyr::filter(!is.na(logPostData))%>%
  ggplot(mapping=aes(x=l_age, y=l_medv))+
  geom_tile(mapping=aes(fill=-logPostData-logPrior))+
  geom_contour(mapping=aes(z=-logPostData-logPrior), col="black", bins=20)+
  geom_point(data=min1, col="red")+
  theme_bw()+
  scale_fill_viridis(direction=-1)+
  labs(fill="log Posterior")+
  theme(legend.position = "bottom",
        legend.direction="horizontal")+
  ggtitle("Trained and evaluated on full dataset.")

plot2 <- df_res %>%
  dplyr::filter(!is.na(logPostData2))%>%
  ggplot(mapping=aes(x=l_age, y=l_medv))+
  geom_tile(mapping=aes(fill=-logPostData2-logPrior))+
  geom_contour(mapping=aes(z=-logPostData2-logPrior), col="black", bins=20)+
  geom_point(data=min2, col="red")+
  theme_bw()+
  scale_fill_viridis(direction=-1)+
  labs(fill="log Posterior")+
  theme(legend.position = "bottom",
        legend.direction="horizontal")+
  ggtitle("Trained on 1/4 of dataset, evaluated on the rest.")

library(ggpubr)

ggarrange(plot1, plot2)

```

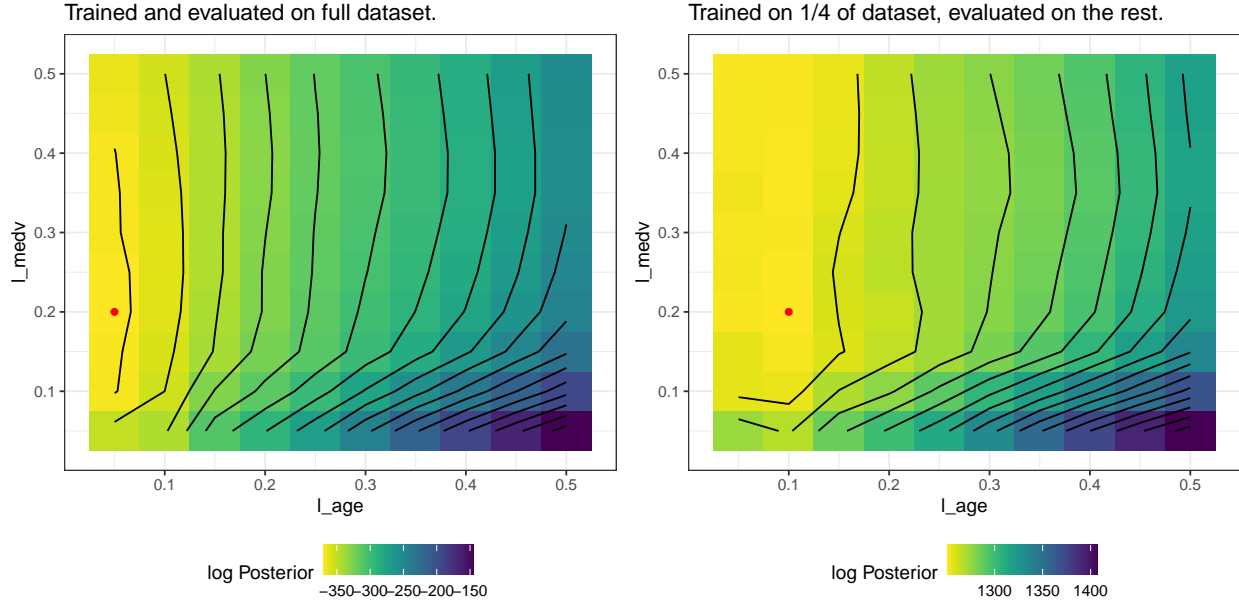



Figure 8: Posterior log-density (up to a constant) over the lengthscale grid (age, medv), under two training strategies. Left: trained and evaluated on the full dataset. Right trained on 25% of the data, evaluated on the remaining 75%. The red dot indicates the optimal lengthscales in each setting.

We observe that:

- Both approaches yield similar optimal lengthscales, showing the stability of the selection.
- These landscapes are fairly consistent with our proposed heuristic of setting all lengthscales around 10-15% of the ranges.

Now, for a qualitative interpretation, we fit and compare SLGP models with three representative choices of lengthscales:

- Short lengthscales (0.01, 0.01) — inducing highly localized priors,
- Long lengthscales (0.5, 0.5) — yielding overly smooth priors,
- Data-driven MAP estimate from the partial training scheme.

```
# Or equivalent, re-use the same basis functions
# and hyper parameters as in the prior we saw

modelMAP2 <- retrainSLGP(SLGPmodel=modelPrior,
  newdata = df,
  method="MAP",
  hyperparams=list(sigma2=1,
    lengthscale=c(0.01, 0.01)),
  sigmaEstimationMethod="heuristic")
modelMAP3 <- retrainSLGP(SLGPmodel=modelPrior,
  newdata = df,
  method="MAP",
  hyperparams=list(sigma2=1,
    lengthscale=c(0.5, 0.5)),
  sigmaEstimationMethod="heuristic")
modelMAP4 <- retrainSLGP(SLGPmodel=modelPrior,
```

```

newdata = df,
method="MAP",
hyperparams=list(sigma2=1,
                  lengthscale=c(min2$l_medv, min2$l_age)),
sigmaEstimationMethod="heuristic")
gc()
#>           used (Mb) gc trigger (Mb) max used (Mb)
#> Ncells 2281950 121.9   4691755 250.6 4691755 250.6
#> Vcells 6766005  51.7   23662044 180.6 28201788 215.2

```

This results in the following fitted densities:

```

dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2], 5),
                                   seq(range_response[1], range_response[2], 101)))
colnames(dfGrid) <- c("age", "medv")
pred1 <- predictSLGP_newNode(SLGPmodel=modelMAP,
                             newNodes = dfGrid)
pred1$Len <- "Heuristic lengthscale"
pred2 <- predictSLGP_newNode(SLGPmodel=modelMAP2,
                             newNodes = dfGrid)
pred2$Len <- "Short lengthscale"
pred3 <- predictSLGP_newNode(SLGPmodel=modelMAP3,
                             newNodes = dfGrid)
pred3$Len <- "Long lengthscale"
pred4 <- predictSLGP_newNode(SLGPmodel=modelMAP4,
                             newNodes = dfGrid)
pred4$Len <- "Optimised lengthscale"
pred <- rbind(pred1, pred2, pred3, pred4)
scale_factor <- 100
ggplot() +
  labs(y = "Proportion of owner-occupied units\nbuilt prior to 1940 [%]",
       x = "Median value of owner-occupied homes [k$]",
       title = "Median value vs. age of homes") +
  theme_bw() +
  geom_ribbon(data=pred,
            mapping=aes(x=medv, ymax=scale_factor*pdf_1+age,
                       ymin=age, group=-age, fill=age),
            col="grey", alpha=0.9) +
  geom_point(data=df,
            mapping=aes(x = medv, y = age), alpha = 0.5, color = "navy") +
  scale_fill_viridis(option = "plasma",
                    guide = guide_colorbar(nrow = 1,
                                           title = "Indexing variable:
                                           Proportion of owner-occupied units built
                                           prior to 1940",
                                           barheight = unit(2, units = "mm"),
                                           barwidth = unit(55, units = "mm"),
                                           title.position = 'top',
                                           label.position = "bottom",
                                           title.hjust = 0.5)) +
  theme(legend.position = "bottom") +
  facet_grid(Len~.) +
  coord_flip()

```

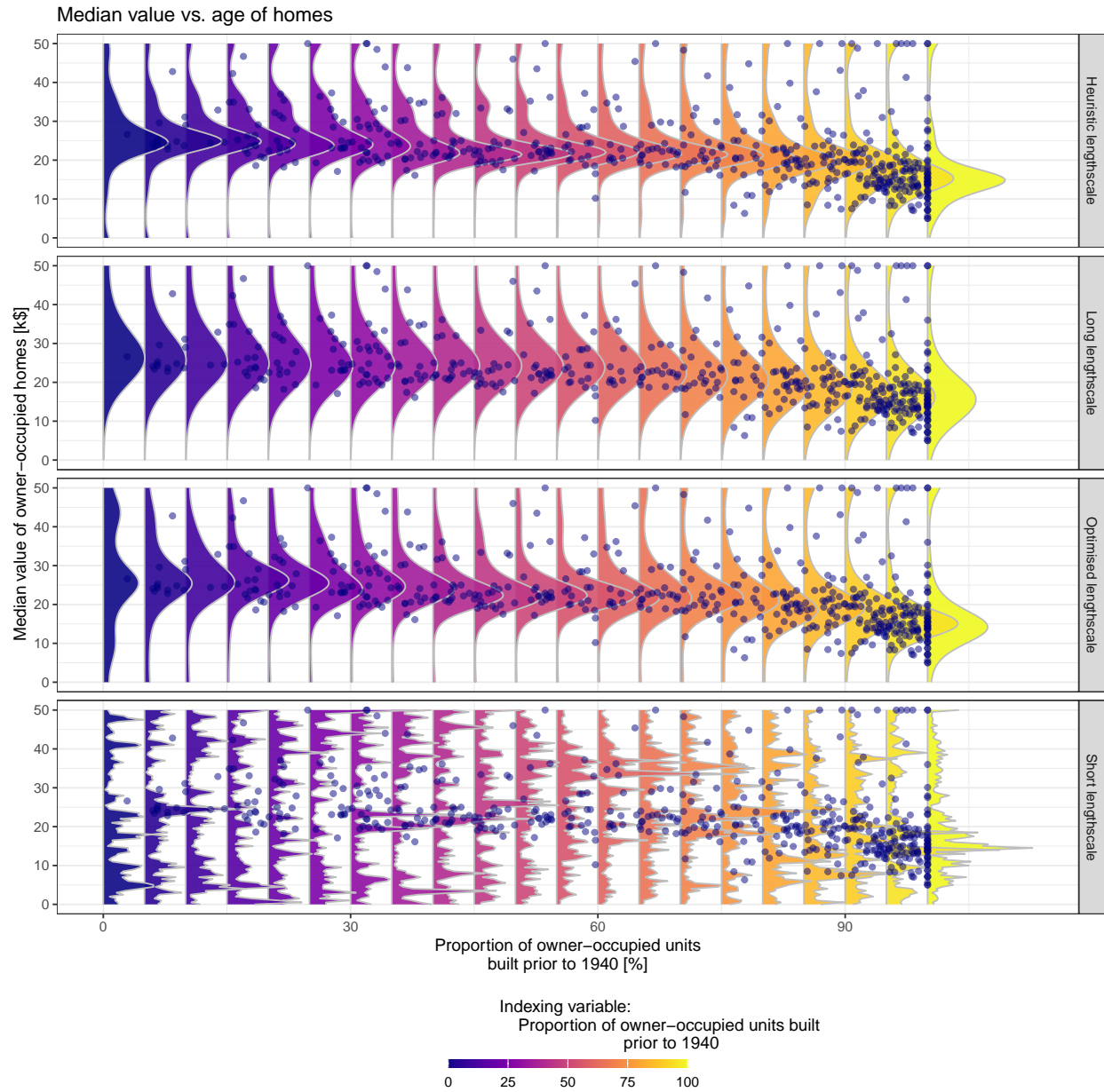


Figure 9: Comparison of fitted densities under different lengthscale settings

```
dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2], 5),
                                   seq(range_response[1], range_response[2], 101)))
colnames(dfGrid) <- c("age", "medv")

dfGrid <- data.frame(expand.grid(selected_values,
                                   seq(range_response[1], range_response[2], 101)))
colnames(dfGrid) <- c("age", "medv")
pred1 <- predictSLGP_newNode(SLGPmodel=modelMAP,
                             newNodes = dfGrid)
pred1$Len <- "Heuristic lengthscale"
pred2 <- predictSLGP_newNode(SLGPmodel=modelMAP2,
```

```

                                newNodes = dfGrid)
pred2$Len <- "Short lengthscale"
pred3 <- predictSLGP_newNode(SLGPmodel=modelMAP3,
                                newNodes = dfGrid)
pred3$Len <- "Long lengthscale"
pred4 <- predictSLGP_newNode(SLGPmodel=modelMAP4,
                                newNodes = dfGrid)
pred4$Len <- "Optimised lengthscale"
pred <- rbind(pred1, pred2, pred3, pred4)
colnames(pred) <- c("age", "medv", "MAP estimator", "Len")
pred <- pred%>%
  pivot_longer(-c("age", "medv", "Len"))
pred$category <- ifelse(pred$age==selected_values[1], names[1],
                        ifelse(pred$age==selected_values[2], names[2], names[3]))

ggplot(mapping=aes(x = medv)) +
  geom_histogram(df_filtered,
                mapping=aes(y=after_stat(density)),
                position = "identity", breaks = seq(0, 50, 2.5),
                fill="darkgrey", col="grey50", lwd=0.2, alpha=0.7) +
  geom_rug(data=df_filtered, sides = "b", color = "navy", alpha = 0.5)+
  geom_line(data=pred, mapping=aes(y=value, group=name),
            color = "black", lwd=0.1, alpha=0.5)+
  geom_line(data=pred, mapping=aes(y=value, group=name, col=name), lwd=1.1)+
  facet_grid(Len ~ category, scales = "free_y") +
  labs(x = "Median value of owner-occupied homes [k$]",
       y = "Probability density",
       title = "Histogram of median value at bins centered at several 'age' values, with width 5\nversus")
  theme_bw()+
  theme(legend.position="bottom",
        legend.direction = "horizontal",
        legend.title = element_blank())+
  coord_cartesian(xlim=range_response,
                  ylim=c(0, 0.25))

```

Histogram of median value at bins centered at several 'age' values, with width 5 versus draws from a SLGP prior

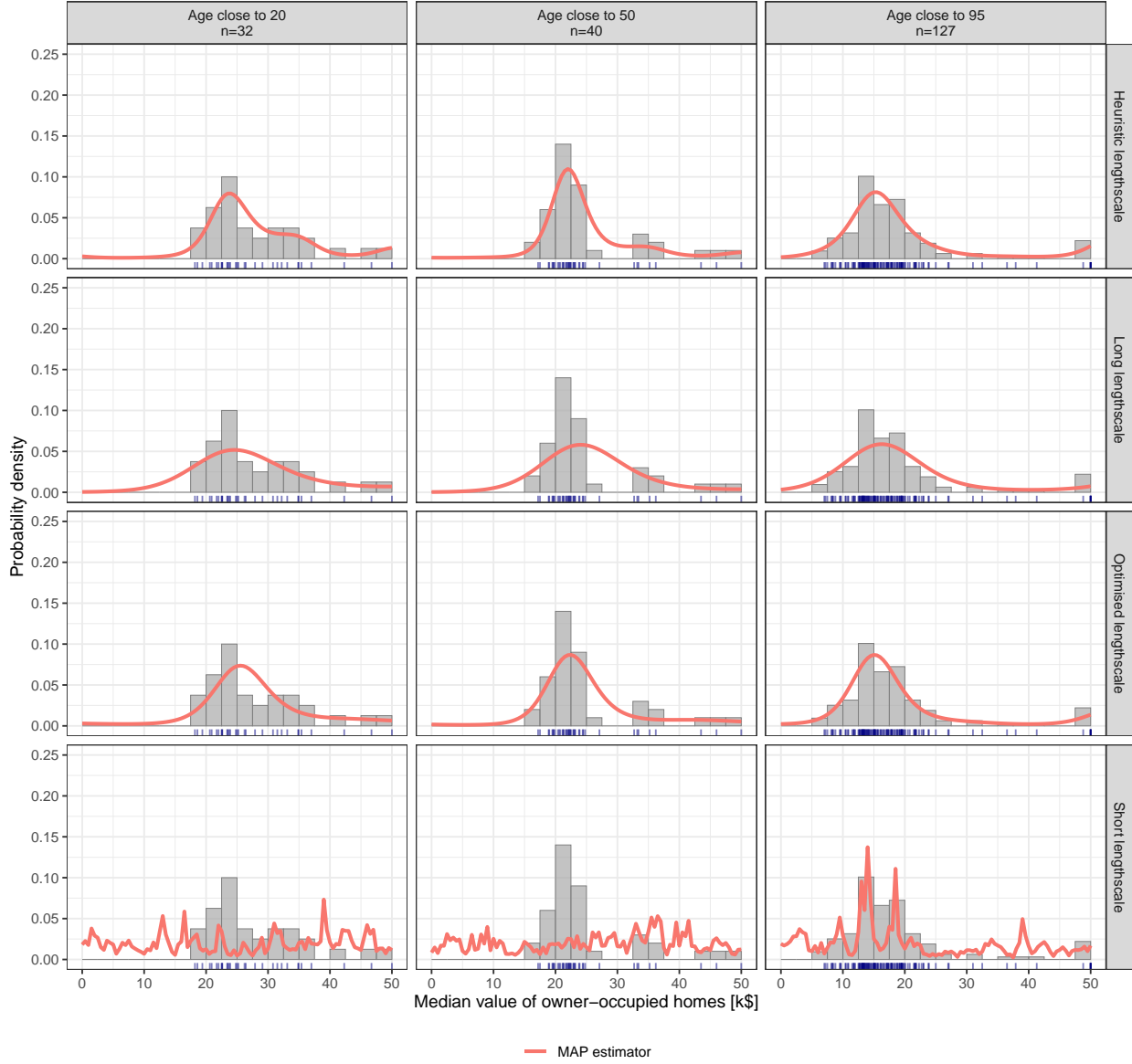


Figure 10: Comparison of fitted densities under different lengthscale settings

We observe that when the lengthscale is too big, while the model captures the general support of the distribution, it fails to capture finer details, such as multi-modalities in certain distributions. This limitation likely arises from using a large lengthscale, which oversmooths small-scale variations. On the opposite using too small a length scale restricts the model's spatial aggregation, limiting the reach of spatial information to a very local scope.

Using a heuristic as before, or the optimised version yield a compromise between the previously observed limitations. For simplicity, we will continue with the lengthscales yielded by the heuristic, set to 15% of the ranges.

2.3 Laplace approximation estimate

By integrating the MAP approach with Laplace approximation, we refine our estimation strategy by approximating the posterior distribution with a multivariate Gaussian. This method strikes a balance between the full Bayesian inference of MCMC and the computational efficiency of MAP estimation. By leveraging both the gradient and Hessian of the posterior, it captures essential curvature information, providing a more informed approximation of the posterior landscape.

As for MAP, we could have use all three implementations: training from scratch, re-training a model or manually specifying components. We go for the second.

```
# Or equivalent, re-use the same basis functions  
# and hyper parameters as in the prior we saw
```

```
modelLaplace <- retrainSLGP(SLGPmodel=modelPrior,  
                           newdata = df,  
                           method="Laplace")
```

Unlike MAP estimation, which provides only a point estimate, the Laplace approximation allows us to visualize uncertainty. To highlight this aspect, we compare multiple posterior draws from the approximation against histogram data, emphasizing how the estimated conditional density fluctuates. The figure below illustrates this by overlaying samples from the posterior approximation with binned data, showcasing the range of possible densities rather than focusing solely on the MAP estimate.

```
# Define the three selected values  
selected_values <- c(20, 50, 95)  
gap <- 5  
  
dfGrid <- data.frame(expand.grid(seq(3),  
                                seq(range_response[1], range_response[2], 101)))  
colnames(dfGrid) <- c("ID", "medv")  
dfGrid$age <- selected_values[dfGrid$ID]  
pred <- predictSLGP_newNode(SLGPmodel=modelLaplace,  
                           newNodes = dfGrid)  
pred$meanpdf <- rowMeans(pred[, -c(1:3)])  
  
library(tidyr)  
# Filter the data: keep values within ±5 of the selected ones  
df_filtered <- df %>%  
  mutate(interval=findInterval(age, c(0,  
                                selected_values[1]-gap,  
                                selected_values[1]+gap,  
                                selected_values[2]-gap,  
                                selected_values[2]+gap,  
                                selected_values[3]-gap,  
                                selected_values[3]+gap)))%>%  
  filter(interval %in% c(2, 4, 6))%>%  
  group_by(interval)%>%  
  mutate(category = paste0("Age close to ", c("", selected_values[1],  
                                              "", selected_values[2],  
                                              "", selected_values[3])[interval],  
                          "\nn=", n()))  
names <- sort(unique(df_filtered$category))  
pred$category <- names[pred$ID]  
  
set.seed(1)
```

```

selected_cols <- sample(seq(1000), size=10, replace=FALSE)
df_plot <- pred %>%
  dplyr::select(c("age", "medv", "category",
                 paste0("pdf_", selected_cols)))%>%
  pivot_longer(-c("age", "medv", "category"))

ggplot(mapping=aes(x = medv)) +
  geom_histogram(df_filtered,
                mapping=aes(y=after_stat(density)),
                position = "identity", breaks = seq(0, 50, 2.5),
                fill="darkgrey", col="grey50", lwd=0.2, alpha=0.7) +
  geom_rug(data=df_filtered, sides = "b", color = "navy", alpha = 0.5)+
  geom_line(data=df_plot, mapping=aes(y=value, group=name),
            color = "black", lwd=0.1, alpha=0.5)+
  geom_line(data=pred, mapping=aes(y=meanpdf, group=category), color = "red")+
  facet_wrap(~ category, scales = "free_y", nrow=1) +
  labs(x = "Median value of owner-occupied homes [k$]",
       y = "Probability density",
       title = "Histogram of median value at bins centered at several 'age' values, with width 5\nversus SLGP draws from a laplace approximation (black curves) and average (red curve)
  theme_bw()+
  coord_cartesian(xlim=range_response,
                  ylim=c(0, 0.25))

```

Histogram of median value at bins centered at several 'age' values, with width 5 versus SLGP draws from a laplace approximation (black curves) and average (red curve)

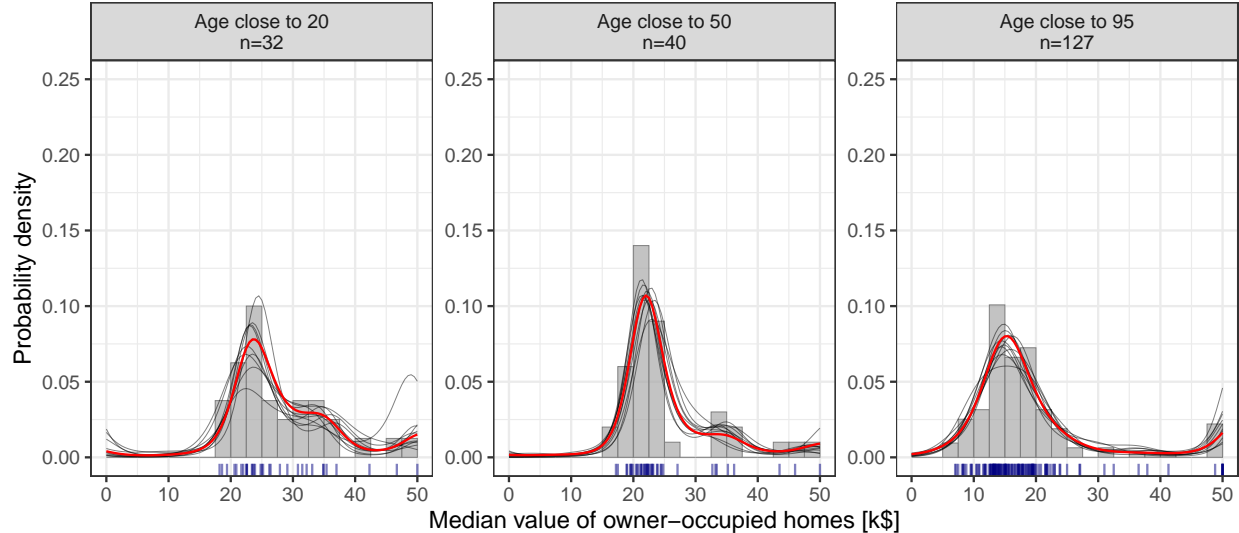


Figure 11: Predictive probability density (and draws from a Laplace approximation) of medv at age, seen over 3 slices.

2.4 MCMC estimate

This method allows us to fully explore the posterior distribution by drawing samples from it. Unlike MAP or Laplace approximation, which provide a single point estimate or a Gaussian approximation, MCMC enables exact Bayesian inference of the underlying density field, given the observed data. The main drawback of this approach being its higher computational cost

As for MAP and Laplace, we could have used all three implementations: training from scratch, re-training a model or manually specifying components. We go for the second again.

```
if(!file.exists("modelMCMC.Rdata")){
  modelMCMC <- retrainSLGP(SLGPmodel=modelPrior,
                           newdata = df,
                           method="MCMC",
                           opts = list(stan_chains=2, stan_iter=1000))
  save(modelMCMC, file="modelMCMC.Rdata")
}else{
  load("modelMCMC.Rdata")
}
```

MCMC provides a rich representation of uncertainty in the estimated density. Instead of showing a single estimate, we compare multiple posterior draws against histogram data, similarly to what we did for the Laplace approximation.

```
# Define the three selected values
pred <- predictSLGP_newNode(SLGPmodel=modelMCMC,
                            newNodes = dfGrid)
pred$meanpdf <- rowMeans(pred[, -c(1:3)])
pred$category <- names[pred$ID]

df_plot <- pred %>%
  dplyr::select(c("age", "medv", "category",
                 paste0("pdf_", selected_cols)))%>%
  pivot_longer(-c("age", "medv", "category"))

ggplot(mapping=aes(x = medv)) +
  geom_histogram(df_filtered,
                mapping=aes(y=after_stat(density)),
                position = "identity", breaks = seq(0, 50, 2.5),
                fill="darkgrey", col="grey50", lwd=0.2, alpha=0.7) +
  geom_rug(data=df_filtered, sides = "b", color = "navy", alpha = 0.5)+
  geom_line(data=df_plot, mapping=aes(y=value, group=name),
            color = "black", lwd=0.1, alpha=0.5)+
  geom_line(data=pred, mapping=aes(y=meanpdf, group=category), color = "red")+
  facet_wrap(~ category, scales = "free_y", nrow=1) +
  labs(x = "Median value of owner-occupied homes [k$]",
       y = "Probability density",
       title = "Histogram of median value at bins centered at several 'age' values, with width 5\\nversus")
  theme_bw()+
  coord_cartesian(xlim=range_response,
                  ylim=c(0, 0.25))
```


Histogram of median value at bins centered at several 'age' values, with width 5 versus SLGP draws from a MCMC (black curves) and average (red curve)

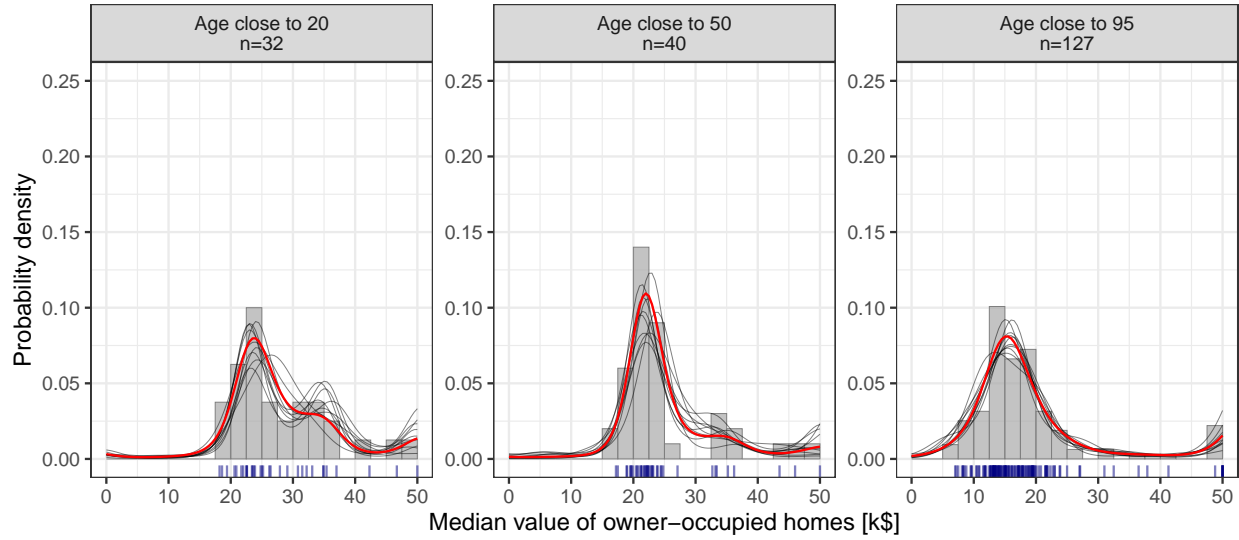


Figure 12: Predictive probability density (and draws from a MCMC) of medv at age, seen over 3 slices.

3 By-products of the estimation

One of the advantages of the SLGP framework is that it predicts entire probability density functions (PDFs) over space. This opens the door to a wide range of nonlinear inferences on the estimated field. In particular, we can compute and visualize functionals of the predicted densities, such as moments (mean, variance, skewness, etc.) or quantiles.

In our current implementation, we provide predictions for both centered and uncentered moments of the estimated PDFs at each location. These derived quantities can themselves be interpreted as spatial fields and are thus informative summaries of the underlying random process.

Importantly, when using probabilistic inference schemes like Laplace approximation or MCMC, the uncertainty in the SLGP predictions naturally propagates to these functionals. As a result, we can also quantify uncertainty on the functionals. For example, we compute credible intervals for the mean or variance field. This is illustrated in the upcoming figure, where we use the MCMC-trained SLGP model to compute moments.

```
dfX <- data.frame(age=seq(range_x[1], range_x[2], 1))

predMean <- predictSLGP_moments(SLGPmodel=modelMCMC,
                                newNodes = dfX,
                                power=c(1),
                                centered=FALSE)
predVar <- predictSLGP_moments(SLGPmodel=modelMCMC,
                                newNodes = dfX,
                                power=c(2, 3, 4),
                                centered=TRUE)

pred <- rbind(predMean, predVar)
pred <- pred %>%
  pivot_longer(~c("age", "power"))%>%
  mutate(value=ifelse(power==2, sqrt(value), value))%>%
  pivot_wider(values_from = value,
              names_from = power)%>%
```

```

mutate(`3`=`3`/`2`^2,
       `4`=`4`/`2`^4)%>%
pivot_longer(-c("age", "name"), names_to = "power")%>%
data.frame()
#> Warning: There was 1 warning in `mutate()`.
#> i In argument: `value = ifelse(power == 2, sqrt(value), value)`.
#> Caused by warning in `sqrt()`:
#> ! Production de NaN

pred$power <- factor(c("Expected value",
                      "Standard deviation",
                      "Skewness", "Kurtosis")[as.numeric(pred$power)],
                    levels=c("Expected value", "Standard deviation",
                              "Skewness", "Kurtosis"))

df_plot <- pred %>%
  group_by(age, power)%>%
  summarise(q10 = quantile(value, probs=c(0.1)),
            q50 = quantile(value, probs=c(0.5)),
            q90 = quantile(value, probs=c(0.9)),
            mean = mean(value), .groups="keep")%>%
  ungroup()

ggplot(df_plot, mapping=aes(x = age)) +
  geom_ribbon(mapping = aes(ymin=q10, ymax=q90),
            alpha = 0.25, lty=2, col="black", fill="cornflowerblue")+
  geom_line(mapping=aes(y=q50))+
  facet_grid(.~power)+
  labs(x = "Proportion of owner-occupied units\nbuilt prior to 1940 [AGE, %]",
       y = "Median value of owner-occupied homes [MEDV, k$]") +
  theme_bw()+
  coord_cartesian(xlim=range_x,
                  ylim=range_response)

```

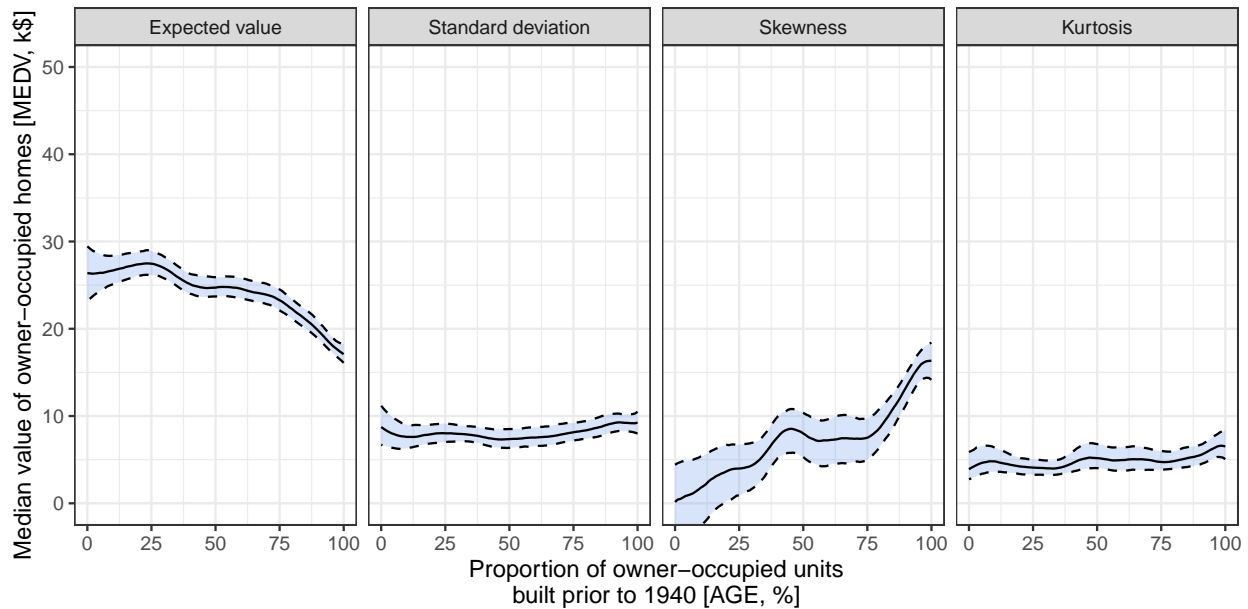


Figure 13: Simultaneous prediction of the fields moments (and associated uncertainty) using a SLGP model

We also support the joint prediction of quantiles at arbitrary levels. Because quantiles are derived directly from the estimated densities, they are guaranteed to be consistent and non-crossing - a property that cannot always be ensured in standard quantile regression. This makes them reliable tools for summarizing distributional shape (e.g., asymmetry, spread) at each location.

```
# Define the three selected values

probsL <- c(5, 25, 50, 75, 95)/100

pred <- predictSLGP_quantiles(SLGPmodel=modelMCMC,
                             newNodes = dfX,
                             probs=probsL)

df_plot <- pred %>%
  pivot_longer(-c("age", "probs")) %>%
  group_by(age, probs) %>%
  summarise(q10 = quantile(value, probs=c(0.1)),
            q50 = quantile(value, probs=c(0.5)),
            q90 = quantile(value, probs=c(0.9)),
            mean = mean(value), .groups="keep") %>%
  ungroup() %>%
  mutate(probs=factor(paste0("Quantile: ", 100*probs, "%"),
                     levels=paste0("Quantile: ", 100*probsL, "%")))

ggplot(df_plot, mapping=aes(x = age, col=probs, fill=probs, group=probs)) +
  geom_ribbon(mapping = aes(ymin=q10, ymax=q90),
            alpha = 0.25, lty=2) +
  geom_line(mapping=aes(y=q50)) +
  labs(x = "Proportion of owner-occupied units\nbuilt prior to 1940 [AGE, %]",
```

```

y = "Median value of owner-occupied homes [MEDV, k$]",
fill = "Quantile levels",
col = "Quantile levels") +
theme_bw() +
coord_cartesian(xlim=range_x,
                ylim=range_response)

```

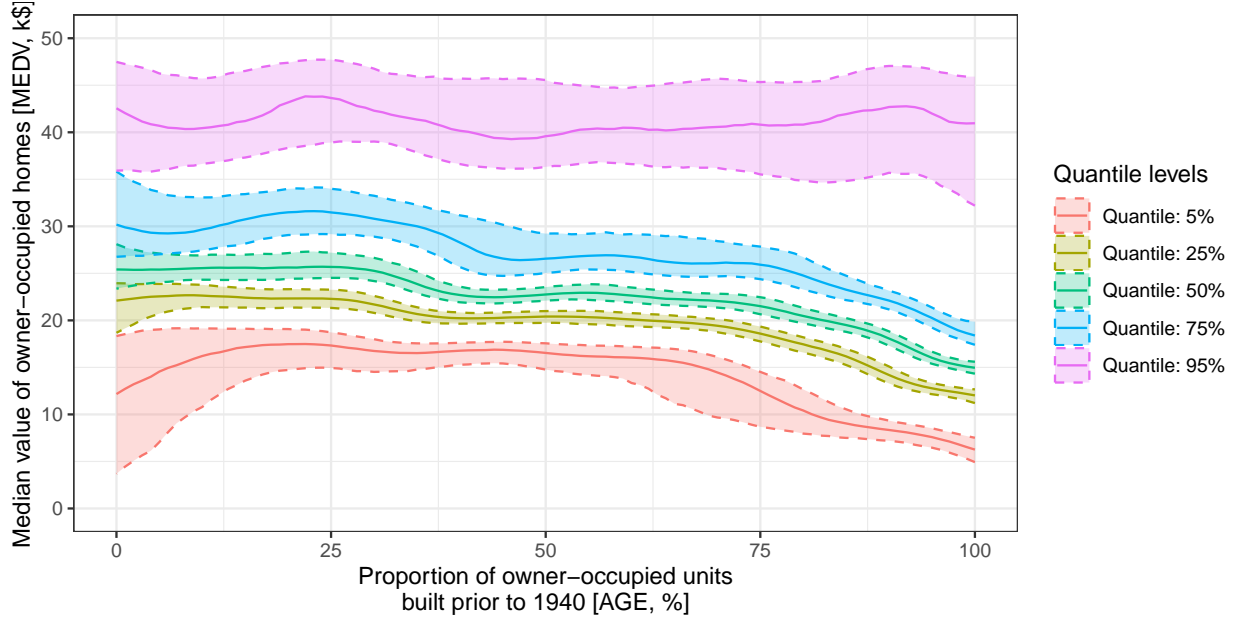


Figure 14: Simultaneous quantile prediction (and associated uncertainty) using a SLGP model

4 Impact of various parameters on the quality of the estimation

To assess the quality of SLGP-based density field estimation, we first define a reference field using the MAP estimate, which serves as the ground truth. We then generate synthetic data from this reference field and compare different estimation setups to evaluate their impact on accuracy.

For each tested configuration, we compute the estimation error, comparing the inferred density field to the reference field. This allows us to measure the effect of different modeling choices. We investigate the impact of the following factors:

- Incorrect hyperparameters: Using suboptimal lengthscale to quantify the effect of poor prior specification.
- Incorrect smoothness assumption: Comparing inference under a Matérn kernel versus an Exponential kernel.
- Data quantity: Evaluating performance when using fewer or more data points, reflecting real-world data availability constraints.
- Number of basis functions: Exploring the trade-off between approximation accuracy and computational efficiency when reducing or increasing the number of basis functions. This analysis provides practical insights into how SLGP estimation behaves under different conditions, guiding model selection for real applications.

In order to quantify the prediction error for different configurations, we define an Integrated Hellinger distance to measure dissimilarity between two probability density valued fields $f(x, \cdot)$ and $f'(x, \cdot)$:

$$d_{IH}^2(f, f') = \frac{1}{2} \int_D \int_T \left(\sqrt{f(\mathbf{v}, u)} - \sqrt{f'(\mathbf{v}, u)} \right)^2 du d\mathbf{v}$$

We first generate 10 different samples from the reference field.

```
len_list <- c(0.01, 0.05, 0.15, 0.5)
n_list <- c(5, 25, 100,
           250, 500, 1000,
           2500, 5000, 10000,
           25000, 50000, 100000)
nFreq_list <- c(10, 50, 100, 200, 500)
matpar_list <- c(1/2, 5/2, Inf)
rep_list <- seq(10)

for(i in seq(max(rep_list))){
  title <- paste0("./res/samp_", i, ".Rdata")
  samp <- data.frame()
  if(!(file.exists(title))){
    for(slice in seq(10)){
      cat("Rep", i, "slice", slice, "/10\n")
      set.seed(i*1000*slice)
      newX <- data.frame(age=runif(max(n_list)/10, range_x[1], range_x[2]))
      temp <- sampleSLGP(modelMAP,
                        newX = newX,
                        n=1,
                        interpolateBasisFun = "WNN")
      samp <- rbind(samp, temp)
    }
    save(samp, file=title)
    gc()
  }
}
```

And then run the density estimations with different parameters on these samples.

```
dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2], 101),
                                seq(range_response[1], range_response[2], 101)))
colnames(dfGrid) <- c("age", "medv")
predRef <- predictSLGP_newNode(SLGPmodel=modelMAP,
                              newNodes = dfGrid)

for(i in seq(max(rep_list))){
  set.seed(i)
  title <- paste0("./res/samp_", i, ".Rdata")
  load(file=title)
  for(matpar in matpar_list){
    for(nFreq in nFreq_list){
      title2 <- paste0("./res/SLGP_Mat_", matpar*2,
                      "half_nFreq_", nFreq,
                      "_rep_", i, ".Rdata")
      if(!(file.exists(title2))){
        modelCurrent <- slgp(medv~age,
```

```

        data=samp[1:5, ],
        method="none",
        basisFunctionsUsed = "RFF",
        interpolateBasisFun="WNN",
        sigmaEstimationMethod = "heuristic",
        predictorsLower= c(range_x[1]),
        predictorsUpper= c(range_x[2]),
        responseRange= range_response,
        opts_BasisFun = list(nFreq=nFreq,
                             MatParam=matpar),

        seed=i)
    save(modelCurrent, file=title2)
    cat("Created and saved model ", title2, "\n")
  }else{
    load(title2)
  }
  for(len in len_list){
    title3 <- paste0("./res/Mat_", matpar*2,
                     "half_nFreq_", nFreq,
                     "_len_", len*100, "%",
                     "_rep_", i, ".Rdata")
    modelCurrent@hyperparams$lengthscale <- c(len, len)
    if(!file.exists(title3) | i==1 & nFreq==100){
      dH_list <- rep(NA, length(n_list))
      for(j in seq_along(n_list)){
        cat(".")
        n <- n_list[j]
        modelCurrent <- retrainSLGP(SLGPmodel=modelCurrent,
                                     newdata = samp[1:n, ],
                                     method="MAP")
        predTemp <- predictSLGP_newNode(SLGPmodel=modelCurrent,
                                         newNodes = dfGrid)
        dH <- sqrt(mean((sqrt(predRef$pdf_1)-sqrt(predTemp$pdf_1))^2)*
                    diff(range_x)*diff(range_response))
        dH_list[j] <- dH
        if(i==1 & nFreq==100 & n==100){
          mod1 <- modelCurrent
        }
        if(i==1 & nFreq==100 & n==10000){
          mod2 <- modelCurrent
        }
      }
      if(i==1 & nFreq==100){
        save(dH_list, mod1, mod2, file=title3)
      }else{
        save(dH_list, file=title3)
      }
      cat("Did ", title3, "\n")
      gc()
    }
  }
}
}
}

```

```
}
```

we concatenate the results and are ready to visualise them.

```
df_res <- data.frame()

for(i in seq(max(rep_list))){
  for(matpar in matpar_list){
    for(nFreq in nFreq_list){
      for(len in len_list){
        title3 <- paste0("./res/Mat_", matpar*2,
                          "half_nFreq_", nFreq,
                          "_len_", len*100, "%",
                          "_rep_", i, ".Rdata")
        if(file.exists(title3)){
          load(file=title3)
          temp <- data.frame(matpar=matpar,
                             nFreq=nFreq,
                             len=len,
                             n=n_list,
                             dH=dH_list,
                             rep=i)
          df_res <- rbind(df_res, temp)
        }
      }
    }
  }
}
```

When the lengthscale is too small, the estimation fails to generalize effectively, leading to poor density reconstruction.

```
df_res_plot <- df_res %>%
  group_by(matpar, nFreq, len, n)%>%
  summarise(meandH=mean(dH),
            q10=quantile(dH, probs=0.1),
            q90=quantile(dH, probs=0.9), .groups="keep") %>%
  ungroup()%>%
  mutate(len=paste0("Lengthscale: ", 100*len,
                    "%\n of the range"))%>%
  mutate(len=factor(len, levels= paste0("Lengthscale: ",
                                         c(1, 5, 15, 50),
                                         "%\n of the range")))%>%
  mutate(matpar=ifelse(is.infinite(matpar),
                       "RFF of Gaussian kernel",
                       matpar))%>%
  mutate(matpar=ifelse(matpar==0.5,
                       "RFF of exponential kernel",
                       matpar))%>%
  mutate(matpar=ifelse(matpar==2.5,
                       "RFF of Matérn 5/2 kernel",
                       matpar))%>%
  mutate(matpar=factor(matpar, levels= paste0("RFF of ",
                                              c("exponential",
                                              "Matérn 5/2",
```

```

    "Gaussian"),
    " kernel"))))%>%

data.frame()
df_res_plot %>%
  filter(as.numeric(len) == 1 ) %>%
  ggplot(aes(x=n, group=nFreq,
             col=as.factor(nFreq), fill=as.factor(nFreq)))+
  geom_ribbon(mapping=aes(ymin = q10, ymax=q90), alpha=0.1)+
  geom_line(mapping= aes(y=meandH))+
  theme_bw()+
  facet_grid(len~matpar)+
  scale_x_log10()+
  scale_y_log10()+
  xlab("Size of the training sample")+
  ylab("Integrated square Hellinger distance")+
  labs(fill = "Number of frequencies",
       col= "Number of frequencies")+
  theme(legend.direction = "horizontal", legend.position = "bottom")

```

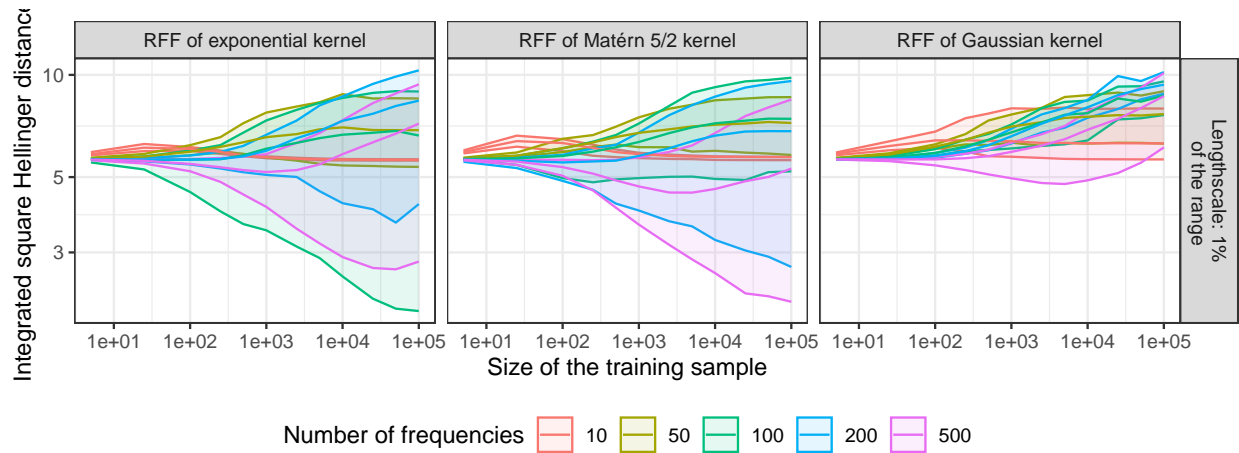


Figure 15: Integrated square Hellinger distance between reference field and estimated field, when the lengthscale is too small

Indeed, with such low lengthscale, the model struggles to share information across different regions, the predictive uncertainty remains high even as the amount of data increases. This is reflected in the Integrated Hellinger distance, which does not decrease significantly as more observations are incorporated.

For other, more reasonable values of the lengthscales, we observe more satisfying behaviours:

```

df_res_plot %>%
  filter(as.numeric(len) != 1 ) %>%
  ggplot(aes(x=n, group=nFreq,
             col=as.factor(nFreq), fill=as.factor(nFreq)))+
  geom_ribbon(mapping=aes(ymin = q10, ymax=q90), alpha=0.1)+
  geom_line(mapping= aes(y=meandH))+
  theme_bw()+
  facet_grid(len~matpar)+
  scale_x_log10()+
  scale_y_log10()+

```



```

xlab("Size of the training sample")+
ylab("Integrated square Hellinger distance")+
labs(fill = "Number of frequencies",
     col= "Number of frequencies")+
theme(legend.direction = "horizontal", legend.position = "bottom")

```

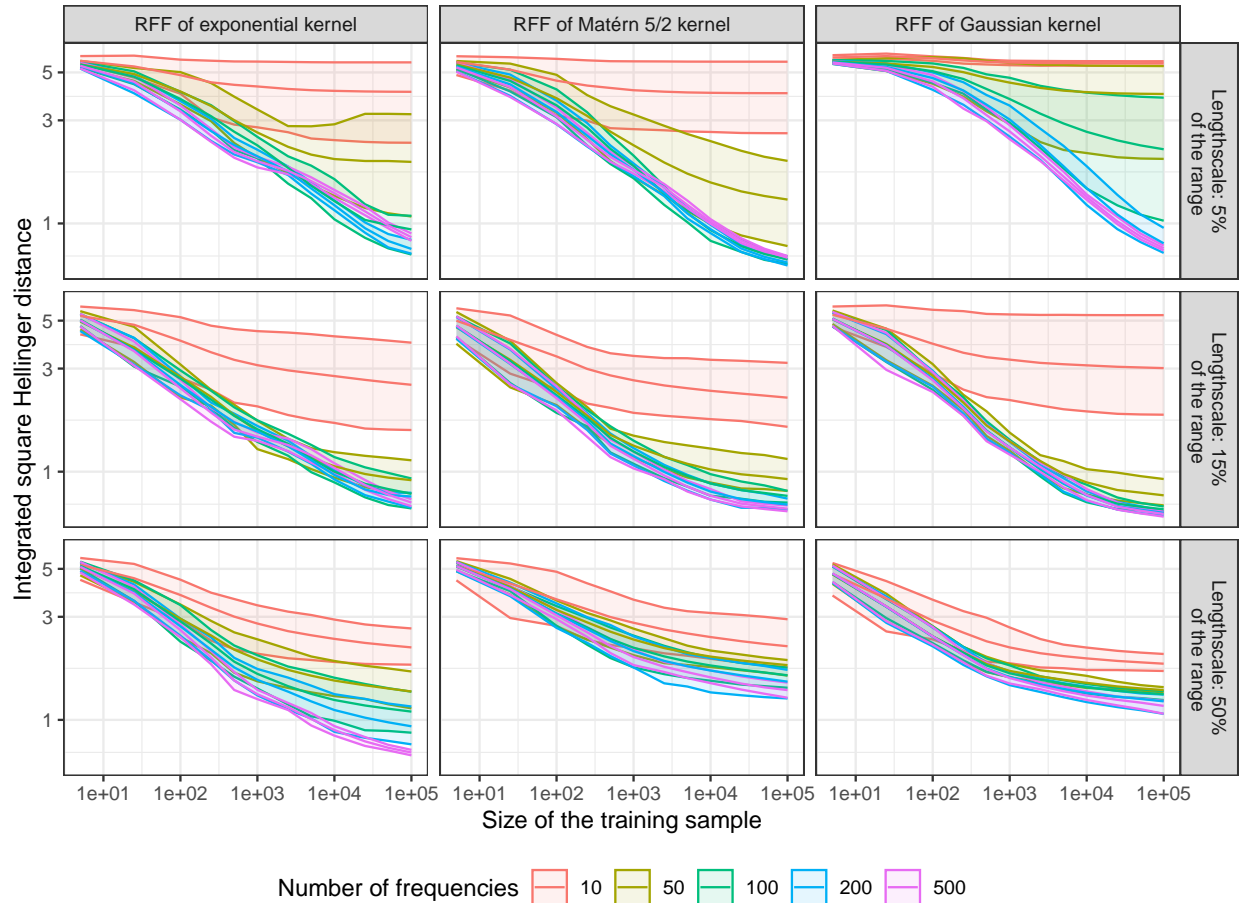


Figure 16: Integrated square Hellinger distance between reference field and estimated field, when the lengthscale is reasonably small, correct, and reasonably big

```

df_res_plot %>%
  filter(as.numeric(len) != 1) %>%
  filter(n >= 100) %>%
  ggplot(aes(x=n, group=nFreq,
             col=as.factor(nFreq), fill=as.factor(nFreq)))+
  geom_ribbon(mapping=aes(ymin = q10, ymax=q90), alpha=0.1)+
  geom_line(mapping= aes(y=meandH))+
  theme_bw()+
  facet_grid(len~matpar)+
  coord_cartesian(ylim=c(min(df_res$dH), 2))+
  scale_x_log10()+
  scale_y_log10()+
  xlab("Size of the training sample")+

```

```
ylab("Integrated square Hellinger distance")+
labs(fill = "Number of frequencies",
     col= "Number of frequencies")+
theme(legend.direction = "horizontal", legend.position = "bottom")
```

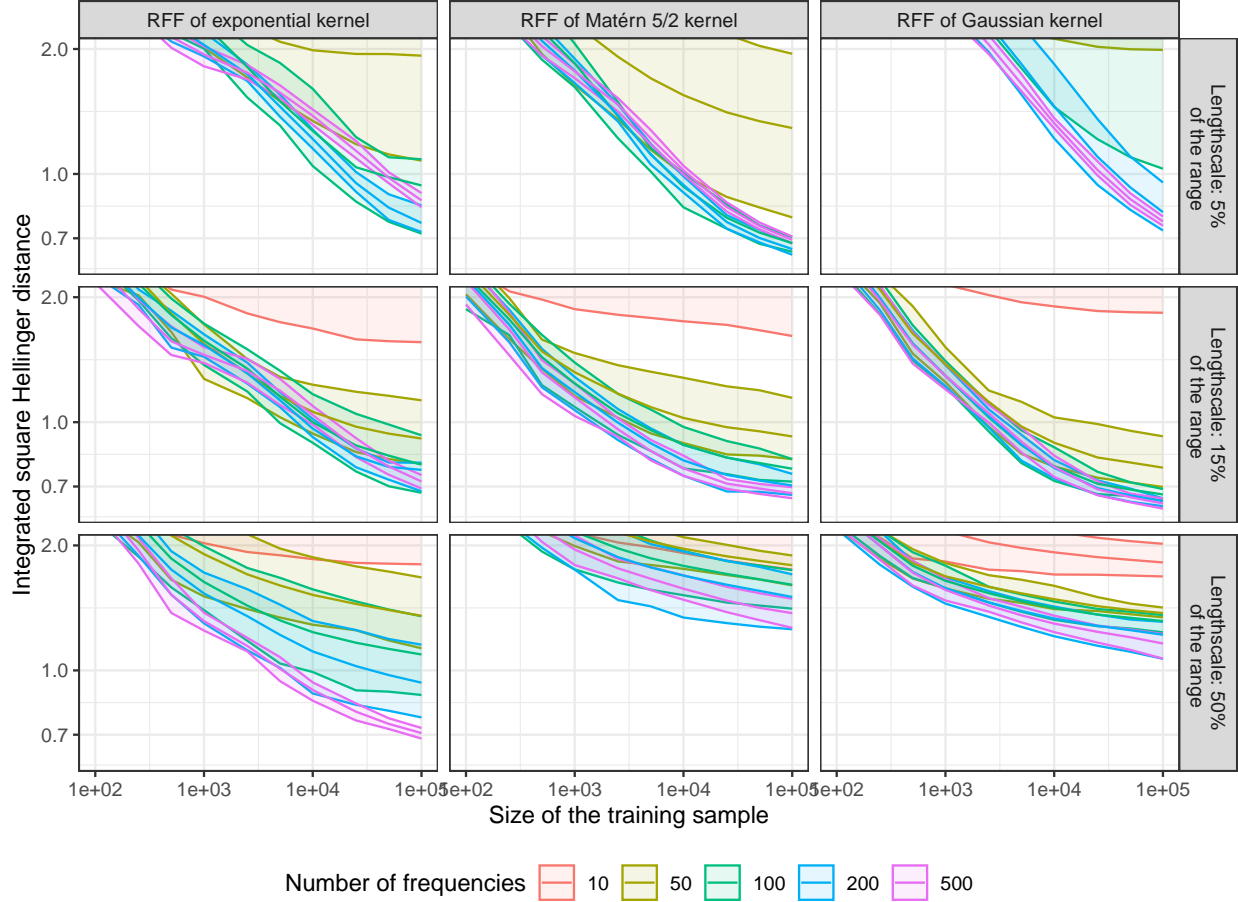


Figure 17: Integrated square Hellinger distance between reference field and estimated field, when the lengthscale is reasonably small, correct, and reasonably big. (Same results, different scale)

Indeed, one notices that higher number of frequencies (i.e. of basis functions) increases the performances of the mode, until the well-specified setting (200 frequencies) is reached.

As observed qualitatively in the MAP section, models that are too smooth (wrong kernel assumptions and/or wrong lengthscale), struggle to capture fine-scale elements.

References

David Harrison and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, March 1978. ISSN 0095-0696. doi: 10.1016/0095-0696(78)90006-2. URL <https://www.sciencedirect.com/science/article/pii/0095069678900062>.