# Supplementary Materials for "Fitting log-Gaussian Cox processes using generalized additive model software" published in the American Statistician

Elliot Dovers, Jakub Stoklosa and David I. Warton

## S1 Full Simulation Results

In addition to the simulation results presented in the manuscript we explored broader simulation settings for the log-Gaussian Cox processes (LGCPs) generated, including:

- $\beta_0 = -4.5, -3.5, -2.5$ to produce smaller, moderate and larger point patterns on average (*i.e.* $E(N) = 266, 730, 1994$ respectively).
- $\rho = 10, 30, 50$, to produce latent fields that varied at finer, moderate and coarser spatial scales relative to the size of the domain (that spanned 100 units in each coordinate direction).
- $\rho_X = 10, 30, 50$, to produce a fixed effect field that varied at finer, moderate and coarser spatial scales relative to the domain. However, the results were consistent across these values and so we only report scenarios in which $\rho_X = 30$.

For our main comparison of fitting LGCPs with `mgcv` (as specified in Section 3) and `R-INLA`, results were broadly consistent across the scenarios above (Figures S1-S3). However, these scenarios provided some useful insights for the specific settings required for `mgcv` to be able to choose the basis dimension (as in Section 3.1) and estimate the spatial range parameter (as in Section 3.2).

Table S1 shows the full simulation results for the scenario presented in the main paper (*i.e.* $\beta_0 = -3.5$ and $\rho = \rho_X = 30$), for which Figure 1 shows the main comparison between `mgcv` and `R-INLA` under simple specifications. We additionally explored different settings for each software. For `mgcv` our key comparisons for fitting LGCP are: methods REML or UBRE via `method="REML"` or `method="GCV.Cp"` respectively, as well as basis functions as Gaussian Processes (GP) or thin plate regression splines (TPRS), via `bs="gp"` or `bs="tprs"`. These comparisons are shown under "Method Spec." and "Basis Type" respectively in Table S1. For `R-INLA` we compared the results using both the default settings for priors on hyperparameters (*i.e.* parameters of the latent field's covariance function, including $\rho$), as well as commonly used, penalized complexity priors (Fuglstad et al., 2019) — these are shown as "Default" and "P.C. Priors" respectively under "Method Spec." in Table S1 however, as these produced near identical results we focus on the latter throughout.

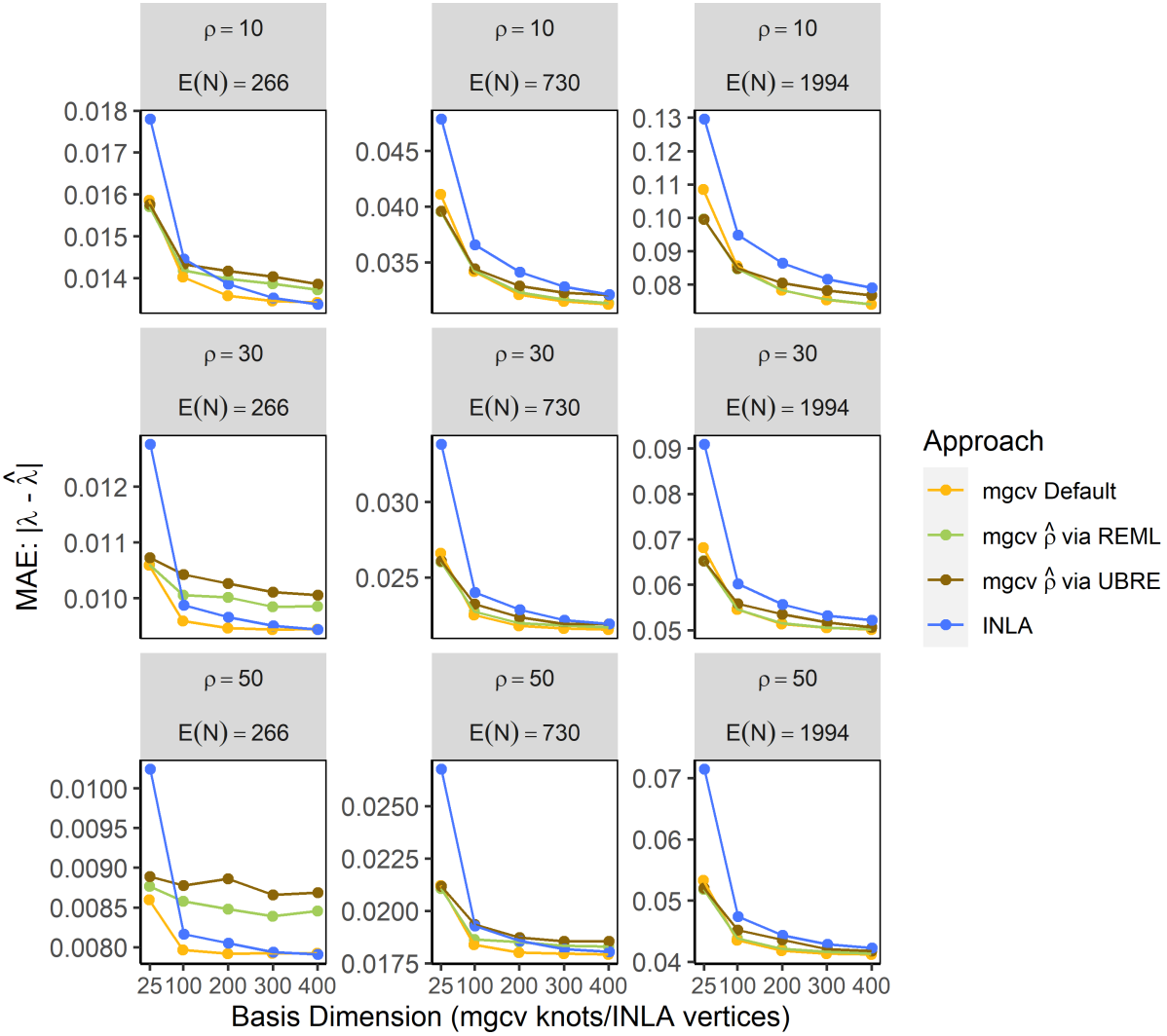


Figure S1: Performance of point estimation from the fitted LGCPs using various approaches, with a range of basis function configurations — represented by increasing basis function dimension for `mgcv`, and mesh vertices for `R-INLA`. Approaches compared were using `mgcv` under default settings presented in Section 3 (mgcv Default); `mgcv` with $\rho$ estimated via the REML and UBRE criteria (mgcv $\hat{\rho}$ via REML and UBRE resp.); as well as using `R-INLA` with penalized complexity priors. The y-axes are the mean absolute error (MAE) between the true intensity field $\lambda$ to that fitted by the model $\hat{\lambda}$ over the 10 000 point grid representing the domain, averaged over simulations. We found that, provided we choose enough basis functions, the models fitted by either approach typically do equally well at fitting the intensity field.

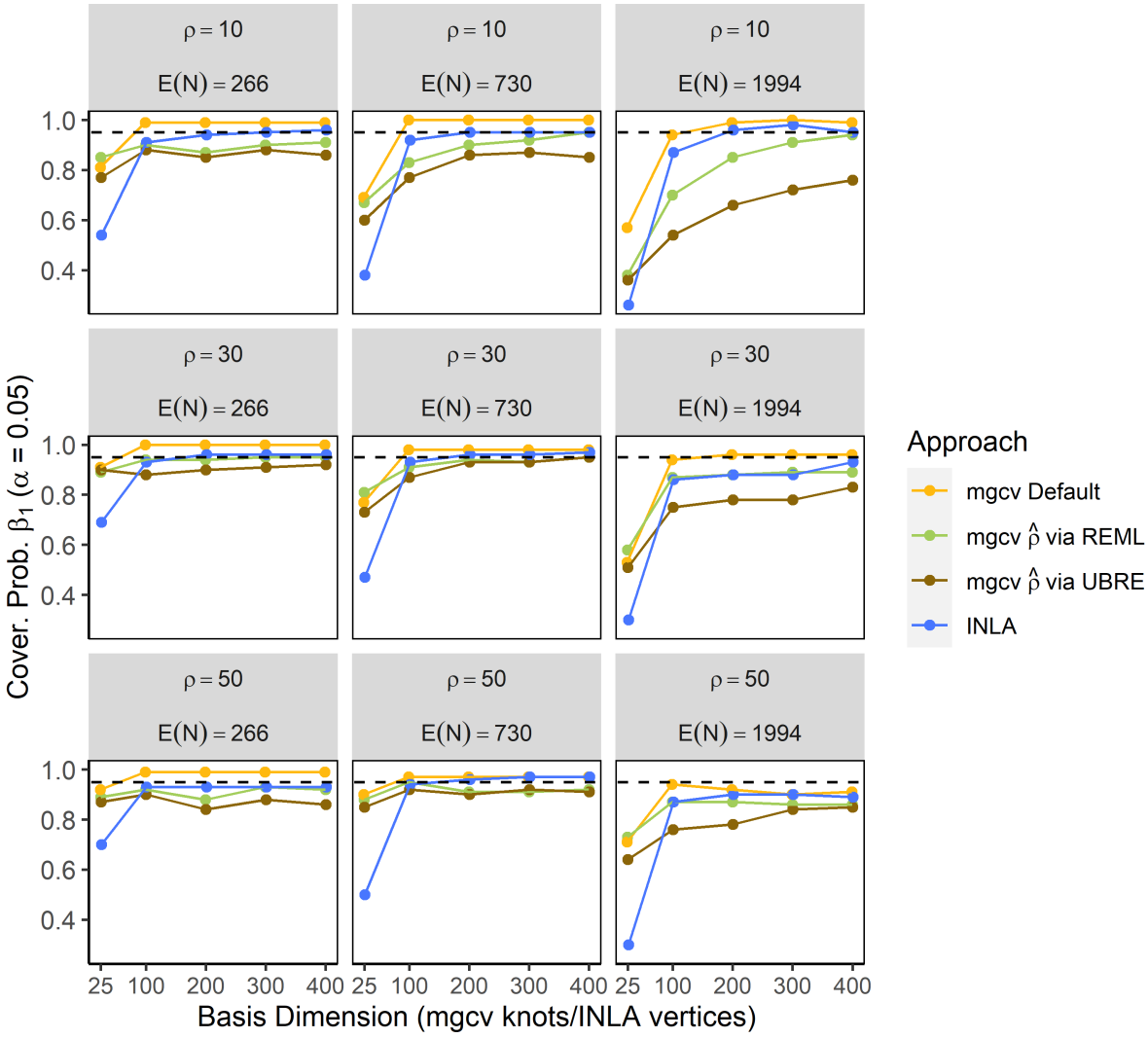


Figure S2: Performance of interval estimation from the fitted LGCPs using various approaches, with a range of basis function configurations — represented by increasing basis function dimension for `mgcv`, and mesh vertices for `R-INLA`. Approaches compared were using `mgcv` under default settings presented in Section 3 (mgcv Default); `mgcv` with $\rho$ estimated via the REML and UBRE criteria (mgcv $\hat{\rho}$ via REML and UBRE resp.); as well as using `R-INLA` with penalized complexity priors. The y-axes are coverage probabilities (Cover. Prob.) of 95% Wald confidence intervals constructed for $\beta_1$ (dashed line indicates nominal coverage).

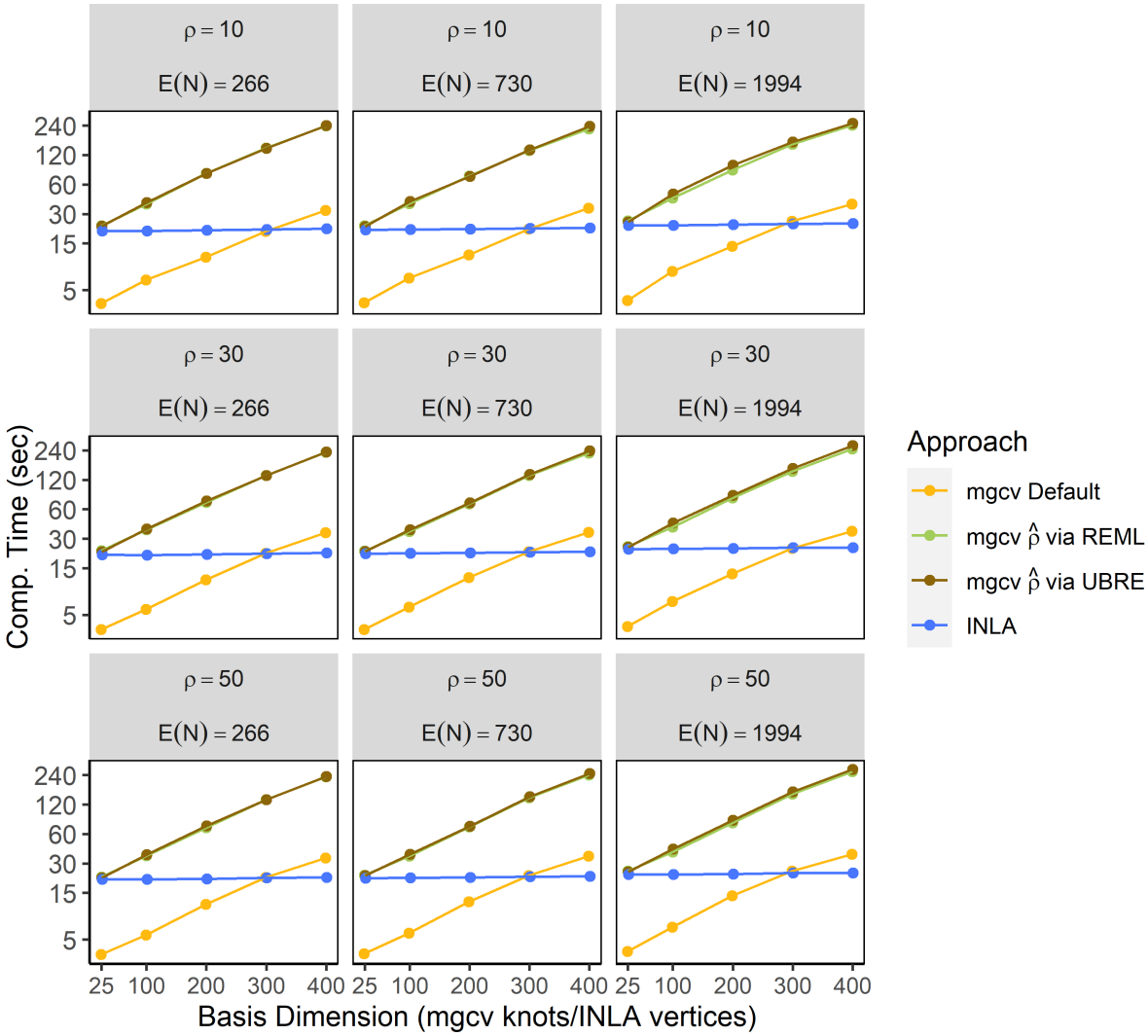


Figure S3: Average computation time of the various approaches to fit simulated LGCPs, with a range of basis function configurations — represented by increasing basis function dimension for `mgcv`, and mesh vertices for `R-INLA`. Approaches compared were using `mgcv` under default settings presented in Section 3 (mgcv Default); `mgcv` with $\rho$ estimated via the REML and UBRE criteria (mgcv $\hat{\rho}$ via REML and UBRE respectively); as well as using `R-INLA` with penalized complexity priors. We found that computation times for `mgcv` became increasingly long with the more basis functions used whereas computation times for `R-INLA` were relatively unaffected by choice of $k$. Moreover, computation times were far longer for `mgcv` when additionally estimating $\rho$ which requires the additional optimization outlined in Section 3.2.

| Software | Method Spec. | Basis Type | Range Crit. | $k$ | MAE $\hat{\lambda}$ | Coverage $\beta_1$ | Comp. Time | RMSE $\hat{\beta}_1$ | RMSE $\hat{\rho}$ |
|---|---|---|---|---|---|---|---|---|---|
| mgcv | REML | GP | | 25 | 0.027 | 89% | 4.597 | 0.301 | |
| mgcv | REML | GP | | 100 | 0.022 | 98% | 5.722 | 0.225 | |
| mgcv | REML | GP | | 200 | 0.022 | 98% | 11.681 | 0.225 | |
| mgcv | REML | GP | | 300 | 0.022 | 98% | 22.872 | 0.225 | |
| mgcv | REML | GP | | 400 | 0.022 | 98% | 38.788 | 0.226 | |
| mgcv | REML | GP | log-Lik. | 25 | 0.026 | 82% | 25.630 | 0.311 | 22.664 |
| mgcv | REML | GP | log-Lik. | 100 | 0.024 | 75% | 48.542 | 0.272 | 26.601 |
| mgcv | REML | GP | log-Lik. | 200 | 0.025 | 68% | 98.049 | 0.271 | 27.755 |
| mgcv | REML | GP | log-Lik. | 300 | 0.026 | 62% | 172.083 | 0.280 | 28.204 |
| mgcv | REML | GP | log-Lik. | 400 | 0.026 | 61% | 263.534 | 0.277 | 28.361 |
| mgcv | REML | GP | AIC | 25 | 0.026 | 82% | 24.768 | 0.300 | 21.865 |
| mgcv | REML | GP | AIC | 100 | 0.023 | 86% | 44.783 | 0.239 | 21.323 |
| mgcv | REML | GP | AIC | 200 | 0.022 | 91% | 85.011 | 0.224 | 19.103 |
| mgcv | REML | GP | AIC | 300 | 0.022 | 92% | 151.082 | 0.221 | 18.316 |
| mgcv | REML | GP | AIC | 400 | 0.022 | 95% | 261.103 | 0.219 | 18.239 |
| mgcv | REML | GP | REML/UBRE | 25 | 0.026 | 81% | 23.032 | 0.297 | 21.658 |
| mgcv | REML | GP | REML/UBRE | 100 | 0.023 | 91% | 35.881 | 0.227 | 17.639 |
| mgcv | REML | GP | REML/UBRE | 200 | 0.022 | 94% | 68.393 | 0.222 | 17.069 |
| mgcv | REML | GP | REML/UBRE | 300 | 0.022 | 93% | 133.130 | 0.223 | 17.321 |
| mgcv | REML | GP | REML/UBRE | 400 | 0.022 | 95% | 226.027 | 0.224 | 16.941 |
| mgcv | UBRE | GP | | 25 | 0.027 | 77% | 3.783 | 0.573 | |
| mgcv | UBRE | GP | | 100 | 0.023 | 98% | 6.218 | 0.227 | |
| mgcv | UBRE | GP | | 200 | 0.022 | 98% | 12.299 | 0.229 | |
| mgcv | UBRE | GP | | 300 | 0.022 | 98% | 22.289 | 0.229 | |
| mgcv | UBRE | GP | | 400 | 0.022 | 98% | 35.168 | 0.229 | |
| mgcv | UBRE | GP | log-Lik. | 25 | 0.026 | 73% | 20.805 | 0.499 | 22.245 |
| mgcv | UBRE | GP | log-Lik. | 100 | 0.024 | 76% | 47.052 | 0.270 | 26.567 |
| mgcv | UBRE | GP | log-Lik. | 200 | 0.025 | 66% | 91.258 | 0.272 | 27.932 |
| mgcv | UBRE | GP | log-Lik. | 300 | 0.026 | 60% | 171.658 | 0.278 | 28.293 |
| mgcv | UBRE | GP | log-Lik. | 400 | 0.026 | 61% | 268.804 | 0.276 | 28.375 |
| mgcv | UBRE | GP | AIC | 25 | 0.026 | 73% | 22.985 | 0.477 | 21.739 |
| mgcv | UBRE | GP | AIC | 100 | 0.023 | 87% | 43.500 | 0.240 | 21.350 |
| mgcv | UBRE | GP | AIC | 200 | 0.022 | 93% | 83.373 | 0.223 | 18.910 |
| mgcv | UBRE | GP | AIC | 300 | 0.022 | 93% | 153.553 | 0.224 | 18.329 |
| mgcv | UBRE | GP | AIC | 400 | 0.022 | 95% | 282.019 | 0.216 | 18.230 |
| mgcv | UBRE | GP | REML/UBRE | 25 | 0.026 | 73% | 22.726 | 0.477 | 21.739 |
| mgcv | UBRE | GP | REML/UBRE | 100 | 0.023 | 87% | 37.500 | 0.240 | 21.350 |
| mgcv | UBRE | GP | REML/UBRE | 200 | 0.022 | 93% | 70.391 | 0.223 | 18.910 |
| mgcv | UBRE | GP | REML/UBRE | 300 | 0.022 | 93% | 136.619 | 0.224 | 18.329 |
| mgcv | UBRE | GP | REML/UBRE | 400 | 0.022 | 95% | 238.066 | 0.216 | 18.230 |
| mgcv | UBRE | TPRS | | 25 | 0.027 | 85% | 1.790 | 0.540 | |
| mgcv | UBRE | TPRS | | 100 | 0.023 | 97% | 5.235 | 11.163 | |
| mgcv | UBRE | TPRS | | 200 | 0.022 | 98% | 11.473 | 0.291 | |
| mgcv | UBRE | TPRS | | 300 | 0.022 | 98% | 20.571 | 0.296 | |
| mgcv | UBRE | TPRS | | 400 | 0.022 | 98% | 34.055 | 0.292 | |
| INLA | P.C. Priors | SPDE Mesh | Default | 25 | 0.034 | 47% | 22.945 | 0.351 | 48.773 |
| INLA | P.C. Priors | SPDE Mesh | Default | 100 | 0.024 | 93% | 22.786 | 0.247 | 38.928 |
| INLA | P.C. Priors | SPDE Mesh | Default | 200 | 0.023 | 96% | 22.945 | 0.242 | 40.177 |
| INLA | P.C. Priors | SPDE Mesh | Default | 300 | 0.022 | 96% | 23.333 | 0.241 | 37.281 |
| INLA | P.C. Priors | SPDE Mesh | Default | 400 | 0.022 | 97% | 23.560 | 0.235 | 37.153 |
| INLA | Default | SPDE Mesh | Default | 25 | 0.034 | 45% | 22.133 | 0.352 | 57.823 |
| INLA | Default | SPDE Mesh | Default | 100 | 0.024 | 94% | 22.360 | 0.251 | 23.747 |
| INLA | Default | SPDE Mesh | Default | 200 | 0.023 | 95% | 22.679 | 0.247 | 22.327 |
| INLA | Default | SPDE Mesh | Default | 300 | 0.022 | 97% | 22.964 | 0.245 | 19.900 |
| INLA | Default | SPDE Mesh | Default | 400 | 0.022 | 97% | 23.321 | 0.238 | 19.781 |

Table S1: Simulation results comparing fitted LGCP using a range of basis dimensions, $k$: mean absolute error in the fitted intensity (MAE $\lambda$); Coverage probability of 95% Wald intervals for $\beta_1$ (Coverage $\beta_1$); Time taken to fit the model (Comp. Time); Root-mean squared error of $\hat{\beta}_1$ (RMSE $\hat{\beta}_1$); and root-mean squared error of $\hat{\rho}$ (RMSE $\hat{\rho}$). A variety settings were compared. 1. *Software* used. 2. *Method Spec.*: `mgcv` criterion, either restricted maximum likelihood (REML), unbiased risk estimator (UBRE), and prior type for `R-INLA`. 3. *Basis Type*: the basis approximation to $\xi$ — Gaussan process (GP), thin-plate regression splines (TPRS), and the stochastic partial differential equation (SPDE Mesh used for `R-INLA`) . 4. *Range Crit.*: the criterion used to estimate $\rho$.

## S1.1 Choice of `method` in `mgcv`

When using the `gam` function in `mgcv`, the `method` argument specifies the criterion that is minimized when fitting the model — like mixed modeling generally, a marginalized or otherwise penalized likelihood must be used due to the penalized smoother(s) acting as random effects. We trialled the default (when modeling a Poisson distribution) of `UBRE`, a generalization to the unbiased risk estimator of the expected mean square error of model predictions (instead using model deviance) and which approximates AIC (Wood, 2004). In the Poisson case, this is given by

$$\text{UBRE Criterion} = \frac{1}{n} D(\hat{\boldsymbol{b}}) + \frac{2}{n} \text{tr}\,(A) - 1 \tag{S:eq1}$$

where $D(\hat{\boldsymbol{b}})$ is the model deviance and $A$ is the influence matrix so that $\text{tr}\,(A)$ represents the effective degrees of freedom. We additionally trialled `method="REML"` which uses a restricted maximum likelihood criterion — the negative of the joint likelihood for all model parameters, marginalized over the smooth coefficients (Wood, 2011) — that is

$$\text{REML Criterion} = -\int L\,(\boldsymbol{\beta}, \boldsymbol{b})\,\mathrm{d}\boldsymbol{b} \tag{S:eq2}$$

The integral is computed using a Laplace approximation. We found that results were broadly consistent when using either criterion for fitting the simulated LGCPs via `mgcv`— there was some indication that using the REML criterion led to slightly better point and interval estimation of the additional fixed effect coefficient $\beta_1$ *only when using a small basis function dimension* (see columns "RMSE $\hat{\beta}_1$" and "Coverage $\beta_1$ in Table S1 for $k = 25$ and where "Range Crit." is blank).

Larger differences in results between the UBRE and REML criteria were observed when additionally estimating the spatial range parameter ($\rho$) of the latent field. In Table S1, column "Range Crit." indicates the criterion by which we optimize the fitted model for $\rho$ as described in Section 3.2. As this is a novel use of the software, we trialled several approaches to optimizing for $\hat{\rho}$: the log-likelihood ("log-Lik."); Akaike's Information Criterion (AIC); and the default criterion according to the method specified ("REML/UBRE") — note that a blank entry for column "Range Crit." in Table S1 indicates that $\rho$ is not estimated. Figure S2 shows that coverage probabilities of Wald confidence intervals (for $\beta_1$) are typically closer to the nominal level when using `method="REML"` — most notably when the latent field is varying at relatively short spatial scales ($\rho = 10$) and the data are numerous ($\beta_0 = -2.5$, which typically leads to about 2000 point events per pattern). Similarly, Figure S6 shows that the model tended to better estimate $\rho$ when using the REML criterion (*i.e.* in more of the simulation scenarios we explored).

## S1.2 Type of basis functions used in `mgcv`

We also explored the sensitivity of simulation results to different basis functions used when fitting LGCPs via `mgcv`. We compared the use of Gaussian processes (`bs="gp"`) to the default for bivariate smooths — thin-plate regression splines (TPRS) as used in Youngman and Economou (2017). Figure S4 shows the simulation results for $\beta_0 = -3.5$ and $\rho = \rho_X = 30$. We found almost no difference in estimating the true intensity field (A), coverage probability of Wald confidence intervals for $\beta_1$ (B, except at $k = 25$ which is an inadequate number of basis functions see Section S1.3), and computation times (C). Using Gaussian process basis functions has the additional benefit of being able to estimate the spatial range of the latent field, $\rho$, as in Section 3.2.

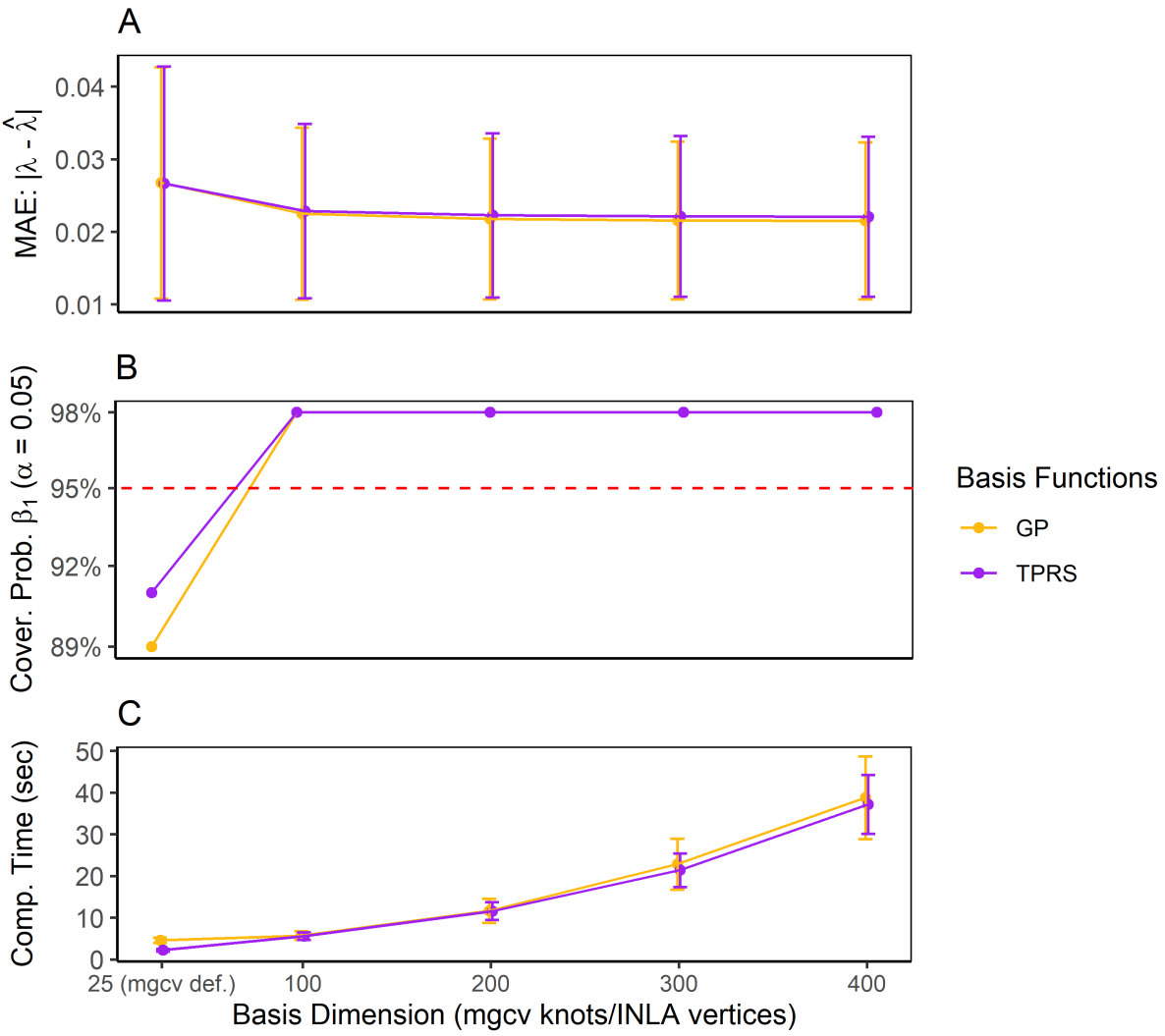


Figure S4: Performance of point and interval estimators from models fitted to simulated LGCP via `mgcv` using basis functions set as either Gaussian process smooths (GP) or Thin-Plate Regression Splines (TPRS) across an increasing basis function dimension. A: Mean absolute error between the true intensity field $\lambda$ to that fitted by the model $\hat{\lambda}$, estimated over a 10 000 point grid. B: Coverage probabilities of 95% Wald confidence intervals constructed on $\hat{\beta}_1$ (dashed line indicates nominal coverage). C: Computation time in seconds. We found that the basis function types typically do equally well at point and interval estimation, in similar computation times.

## S1.3 Choosing basis dimension

Our simulation results tended to show that the fitted model is consistent (in terms of point and interval estimation of the LGCP intensity $\lambda$ and fixed effect coefficient, $\beta_1$ respectively), provided a "large enough" basis dimension, $k$ is used. Figure S1 shows that the mean absolute error in estimating $\lambda$ typically levels off for $k = 200$ or more in all scenarios, however we found that interval estimation was more sensitive to the choice of $k$. Figure S2 shows that nominal coverage was reached at different values of $k$, depending on the scenario. We typically needed a larger $k$ to capture the latent field when it is operating at shorter spatial scales ($\rho = 10$) and when the expected number of points was greater ($E(N) = 1994$).

So how can we determine the adequacy of $k$ in real applications where we do not know the underlying truth? As per Wood (2017, section 5.9) we examined the effective degrees of freedom (edf), which is recommended to be "small" compared to $k$. Figure S5 shows the edf plotted against $k$ for the simulations scenarios examined. We found that a general rule of $k = 4 \times \text{edf}$ suggests a large $k$ when $\rho$ is small and $E(N)$ is large, and which broadly aligns with the results in Figure S2.

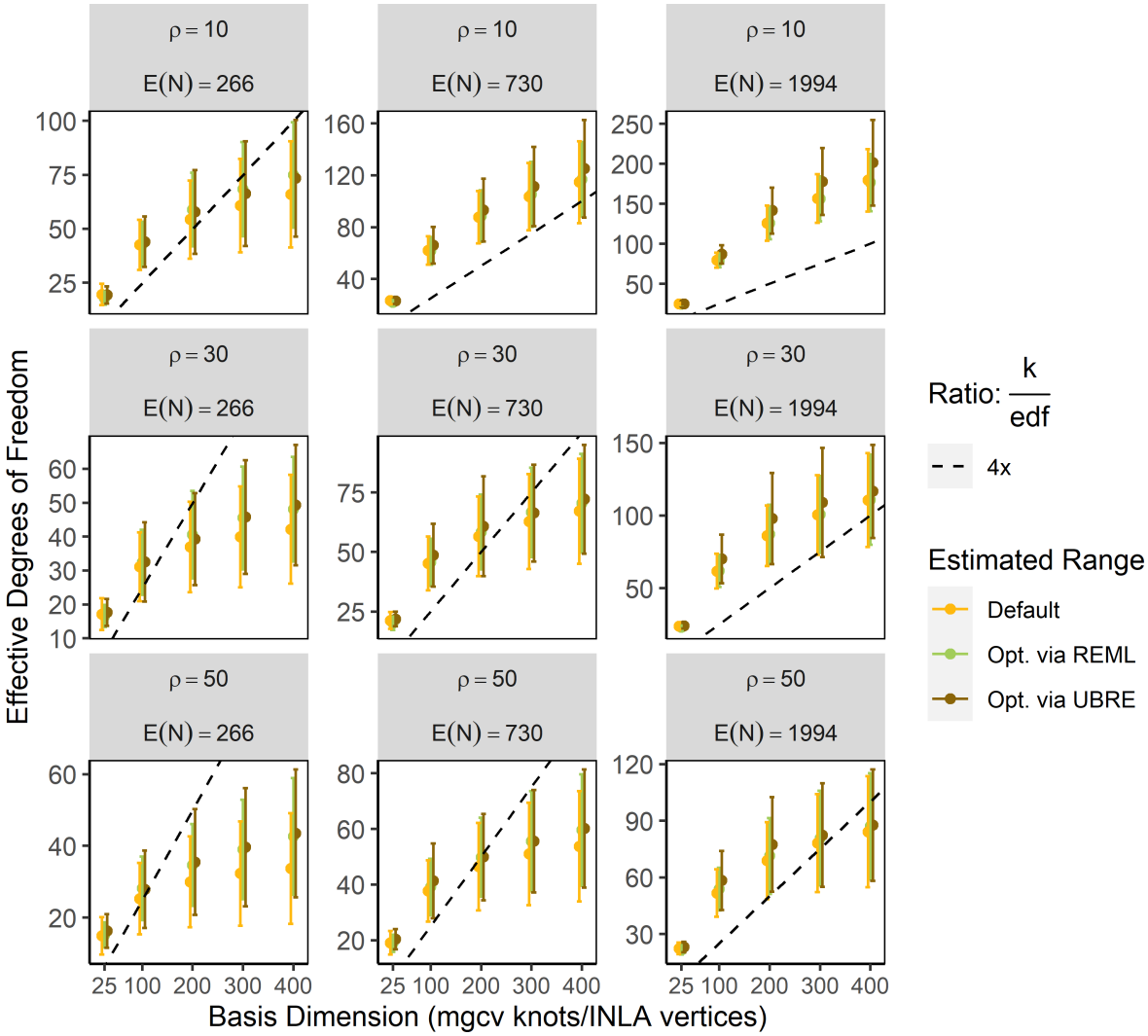


Figure S5: The effective degrees of freedom from the fitted LGCP simulations using `mgcv` (with $\rho$ either taking the default value or being estimated, as indicated in legend) across a range of basis function dimensions ($k = 25, 100, 200, 300, 400$). Panels indicate different values of spatial range of the latent field ($\rho$) and expected point pattern size ($E(N)$, *i.e.* the amount of data). The dashed line indicates our proposed stopping rule for an adequately large $k$, *i.e.* when $k$ is $4\times$ the effective degrees of freedom. Since both the spatial range and expected number of points determine the realized point pattern size, we found that larger $E(N)$ and smaller $\rho$ require larger $k$ to meet our proposed adequacy rule.

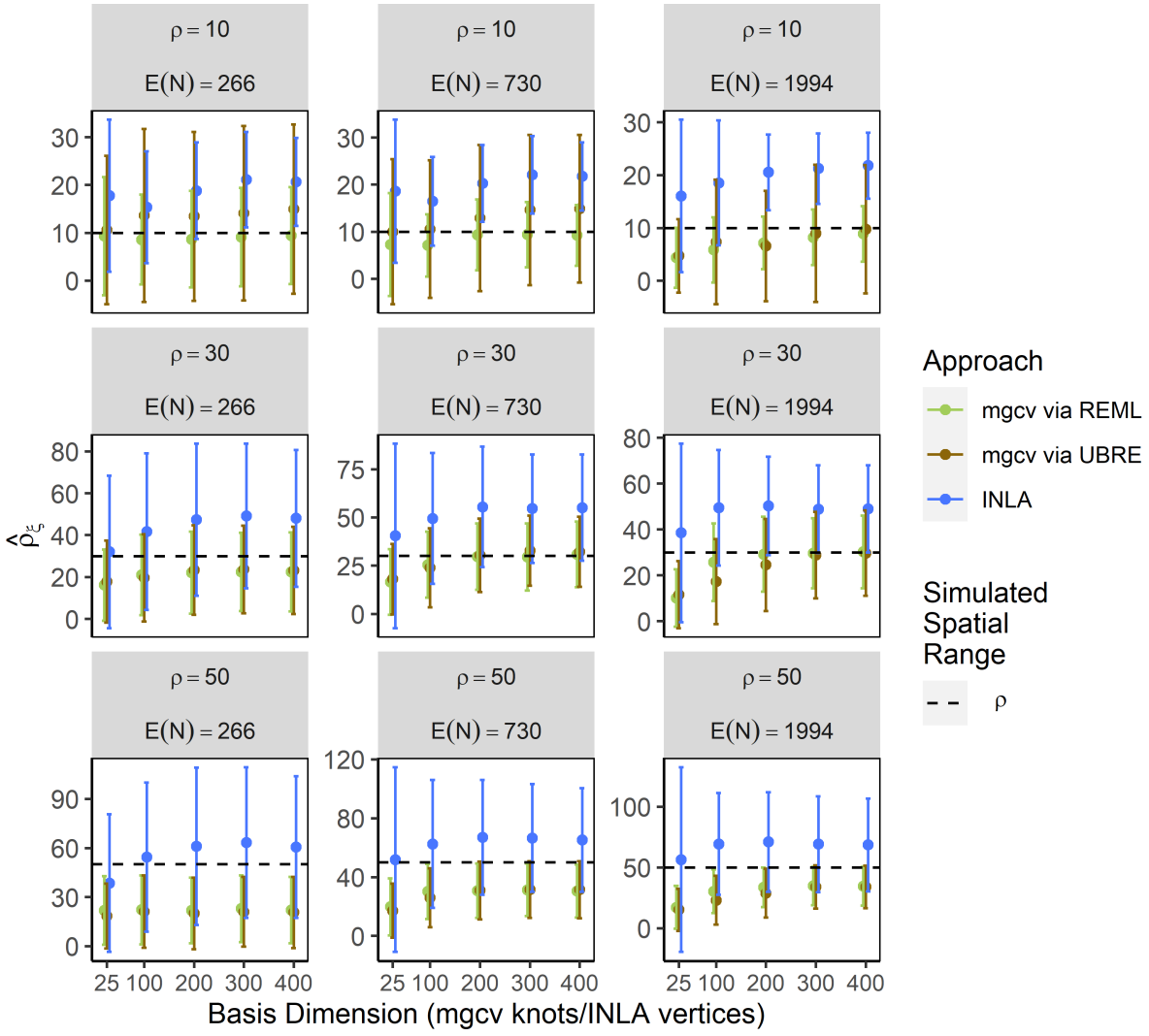


Figure S6: Estimated range parameters, $\rho$, for models fitted via `R-INLA` and `mgcv`, for a range of increasingly large basis dimension ($k = 25, 100, 200, 300, 400$). Panels indicate different values of spatial range of the latent field ($\rho$) and expected point pattern size ($E(N)$, *i.e.* the amount of data). We found that models fitted via `mgcv` tended to perform better than `R-INLA` under these simulation settings, and that when using `mgcv` setting `method="REML"` outperformed the default in more scenarios.

## S1.4 Estimating spatial range of the latent field

Table S1 shows that the default value for the spatial range parameter tends to work well (in terms of estimating the underlying intensity and performing inference on fixed effects) however, it may sometimes be desirable to estimate the range of the latent Gaussian field. Section 3.2 describes how the spatial range parameter, $\rho$ can be estimated when setting `bs="gp"` within the bivariate smooth that estimates the latent field of LGCP. In the full simulations described in Section S1 we also compared the ability of `mgcv` (using both criteria described in Section S1.1) and `R-INLA` to estimate the true spatial range parameter of the data generating process.

Figure S6 shows the estimated spatial range parameter $\hat{\rho}$ for `R-INLA` as well as `mgcv` when using the different fitting criteria, (S:eq1) and (S:eq2). We found that using the REML criterion tended to more

accurately estimate $\rho = 10$ in the data poor settings ($E(N) = 266,730$).

We note that the process for estimating the spatial range of the latent field is not particularly efficient. There would be scope for improving this (and the efficiency of fitting models using the `bs="gp"` smoothers in `mgcv` more broadly) if sparse basis function matrices could be utilized. Implementing this goes beyond the scope of the current paper but we provide some discussion here. Since the range parameter dictates how densely populated the basis function matrices are with non-zero elements, a combination of a short spatial range parameter and sparse matrix calculations can vastly improve computation times (a similar principal is used in Dovers et al., 2023). The key to efficient model fitting then becomes using as small a spatial range as possible while still capturing the underlying spatial correlation of the Gaussian field. This strategy is at odds with the current default, as recommended in Kammann and Wand (2003), which is set to be the largest distance between locations in the data. However, this was not done with point process models in mind and as such will be calculated on the quadrature points also, often resulting in a longer than necessary range and in turn denser basis function matrices. Instead a default of the largest distance between *point events only* could induce more sparsity to be utilized by sparse matrix calculations.

Beyond using sparse matrices, scope for improving the estimation of the spatial range parameter is somewhat limited by the fact that most of the computationally demanding aspects of the optimization are a function of $\rho$ when using basis functions of the form of (9). INLA and nearest neighbour Gaussian processes (Datta et al., 2016) mitigate this using different kinds of spatial structures as (what are essentially) positions of the basis function nodes. Another approach would be to use automatic differentiation (AD) within `mgcv`. This would lead to faster model fits since gradient information can speed up optimization algorithms (*e.g.* Shanno, 1970) and enable automated approximation of intractable integrals using the Laplace approximation — to approximate (S:eq2) for example. AD is used on `R` in modeling software, for example in `scampr` (Dovers et al., 2023) and `glmmTMB` (Brooks et al., 2017) by interfacing with Template Model Builder (`TMB` Kristensen et al., 2016).

Despite the efficiency drawbacks, we provide a wrapper function for users to implement the procedure presented in Section 3.2 which additionally uses warm starts for parameter values and returns a final model with an estimated spatial range parameter:

```
gam_lgcp <- function(formula, data, weights = NULL, coord.names = c("x", "y"), k = 200,
    range.interval, opt.tolerance = 3, subset = NULL, na.action,
    offset = NULL,  optimizer = c("outer", "newton"), control = list(), scale = 0,
    select = FALSE, knots = NULL, sp = NULL, min.sp = NULL, H = NULL, gamma = 1,
    fit = TRUE, paraPen = NULL, G = NULL, in.out = NULL, drop.unused.levels = TRUE,
    drop.intercept = NULL, nei = NULL, discrete = FALSE, ...) {

  mc <- match.call() # gets the arguments (must be updated for LGCP as below)
  call.list <- as.list(mc)
  # check the form of the weights
  object.supplied <- tryCatch(!is.null(weights), error = function(e) FALSE)
  if (!object.supplied) {
    weight.name <- deparse(substitute(weights))
    if (weight.name == "NULL") {
      stop("weights must be supplied for fitting a LGCP")
    } else {
      wt.vec <- as.vector(data[,weight.name])
    }
  } else {
    wt.vec <- weights
  }
  # checks
  if ((!all(coord.names %in% colnames(data)))) {
    stop(paste0("One of 'coord.names', ", paste(coord.names, collapse = " or "),
```

```
    ", not found 'data'"))
  }
  # get the response variable out of the formula
  resp <- all.vars(formula[[2]])
  data$new.response <- data[ , resp] / wt.vec
  # alter the call according to requirements for a LGCP
  call.list$family <- poisson()
  call.list$data <- data
  call.list$weights <- wt.vec
  call.list$method <- "REML"
  # update the formula for an initial fit
  call.list$formula <- as.formula(paste0("new.response ~ ", as.character(formula)[3],
    " + s(", paste(coord.names, collapse = ", "), ", bs=\"gp\", k=", deparse(k), ", m=3)"))
  # remove the function of the call
  call.list[[1]] <- NULL
  # fit an initial model to obtain warm starting parameters
  init.mod <- do.call(mgcv::gam, call.list)
  warm.starts <- init.mod$coefficients
  # set basis function coefs to zero
  warm.starts[(length(init.mod$coefficients) - k + 1):length(init.mod$coefficients)] <- 0
  # update the starting parameters
  call.list$start <- warm.starts
  # set up the object function to be optimized
  objective_fn <- function(rho) {
    call.list$formula <- as.formula(paste0("new.response ~ ", as.character(formula)[3],
        " + s(", paste(coord.names, collapse = ", "), ", bs=\"gp\", k=", deparse(k),
        ", m=c(3,", deparse(rho), "))"))
    tmp.m = do.call(mgcv::gam, call.list)
    return(tmp.m$gcv.ubre) # the "method" specific criterion
  }
  # calculate the optimum
  opt <- optimize(objective_fn, interval = range.interval,
    tol = opt.tolerance)
  # adjust the formula for the optimized spatial range parameter
  call.list$formula <- as.formula(paste0("new.response ~ ", as.character(formula)[3],
    " + s(", paste(coord.names, collapse = ", "), ", bs=\"gp\", k=", deparse(k),
    ", m=c(3,", deparse(opt$minimum), "))"))
  # fit the final model
  res <- do.call(mgcv::gam, call.list)
  return(res)
}
```

The above takes the same arguments as `mgcv::gam()` — except for `method` and `family` which we fix as needed for fitting a LGCP — along with additional arguments:

- `coord.names` to provide the names of spatial coordinates found in the `data` provided.
- `k` to specify the basis dimension.
- `range.interval` to specify the interval over which to estimate the spatial range. Can be narrowed to improve computation time.
- `opt.tolerance` to control the tolerance level within `optimize()`. Can be altered to improve computation time.

The right hand side of the `formula` argument specifies the linear predictor *excluding* the bivariate smoother (which is added automatically using `coord.names`) and the left hand side is simply the binary point event/quadrature identifier within the data provided — this will automatically be divided by the weights as in (4). For example, the model fitted in Section 3 could be fitted with an estimated spatial range parameter via:

```
m <- gam_lgcp(pt ~ X, data = data, weights = wt, range.interval = c(min_dist, max_dist))
```

## S1.5 INLA specification

The following code was used to fit `R-INLA` in the simulations presented. We set up the mesh object, used as integration points used to approximate the marginalization of the latent field:

```
mesh <- inla.mesh.2d(loc.domain = quad[ , c("x", "y")],
 max.edge=c(5,10), cutoff=2, offset = c(5,10), max.n.strict = c(k, 0))
```

where `quad` is a data frame of the regular grid of quadrature points, spanning the domain and `k` is the "basis function dimension" to be fitted — we use this term loosely since these are actually mesh vertices but are the nearest equivalent to basis function dimension in `mgcv`. Priors for the hyperparameters of the Matérn covariance are specified two ways:

```
# set the spde representation to be the mesh with penalized complexity priors
spde.pcmatern <- inla.spde2.pcmatern(mesh, alpha = 1.5, prior.sigma = c(10, 0.01),
    prior.range = c(1, 0.01))
# and with the default Matern settings
spde.matern <- inla.spde2.matern(mesh)
```

These correspond to penalized complexity priors (Fuglstad et al., 2019) and the default respectively. The model is then fitted using standard code according to Illian et al. (2012); Simpson et al. (2016):

```
# make A matrix for point pattern
data_A <- inla.spde.make.A(mesh = mesh, loc = as.matrix(pp[,c("x","y")]))

# make A matrix for quadrature points
quad_data_A <- inla.spde.make.A(mesh = mesh, loc = as.matrix(quad[,c("x","y")]))

# make A matrix for the prediction points
pred_data_A <- inla.spde.make.A(mesh = mesh, loc = as.matrix(pred[ , c("x","y")]))

# set the numbers of various points
nq <- nrow(quad)
n <- nrow(pp)
np <- nrow(pred)

# change data to include 0s for nodes and 1s for presences
y.pp <- rep(0:1, c(nq, n))

# add expectation vector (area for integration points/nodes and 0 for presences)
e.pp <- c(quad$quad.size, rep(0, n))

# combine integration point A matrix over quadrature with point pattern data A matrix
A.pp <- rbind(quad_data_A, data_A)

# create data stack
```

```
stk_data <- inla.stack(data=list(y=y.pp, e = e.pp),
 effects=list(list(data.frame(Intercept=rep(1,nq+n)), env = c(quad$env, data$env)),
 list(i=1:mesh$n)), A=list(1,A.pp), tag="data")

# create the prediction stack
stk_pred_response <- inla.stack(data=list(y=NA),
 effects = list(list(data.frame(Intercept=rep(1,np))), env = pred$env, list(i=1:mesh$n)),
 A=list(1,1, pred_data_A), tag='pred_response')

# combine the stacks
stk <- inla.stack(stk_data, stk_pred_response)

# fit the models
result_pcmatern <- inla(y ~ Intercept + env + f(i, model = spde.pcmatern) - 1,
 family="poisson", data=inla.stack.data(stk),
 control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
 control.family = list(link = "log"), E = inla.stack.data(stk)$e,
 control.compute = list(cpo=TRUE, waic = TRUE, dic = TRUE)
)
result_default <- inla(y ~ Intercept + env + f(i, model = spde.matern) - 1,
 family="poisson", data=inla.stack.data(stk),
 control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
 control.family = list(link = "log"), E = inla.stack.data(stk)$e,
 control.compute = list(cpo=TRUE, waic = TRUE, dic = TRUE)
)
```

# S2 Analysing the Gorilla Nesting Data

This section serves as a vignette demonstrating the analyses conducted throughout Section 5 on the gorillas nesting data (Funwi-Gabga and Mateu, 2012) in `R`, including:

- Converting the data into the appropriate format to fit point process models (PPMs) with `mgcv`
- Fitting a LGCP model to the data via the `mgcv` package and selecting an appropriate basis dimension $k$ and estimating the spatial range parameter $\rho$.
- Performing model diagnostics by interfacing with the `spatstat` package
- Performing model selection, comparing inhomogeneous Poisson and log-Gaussian Cox processes and the form of the fixed effect covariates.

Load/install the required packages:

```
if(!require(mgcv, quietly = T)){
  install.packages("mgcv")
  library(mgcv)
}
if(!require(spatstat, quietly = T)){
  install.packages("spatstat")
  library(spatstat)
}
```

## S2.1 Data Preparation

The gorillas data is available in the required format within the `scampr` package but can be built from `spatstat` according to the following code. Additionally, this sets up the point weights for fitting the PPM, and centers and scales covariates to assist convergence of fitting routines.

```
data(gorillas, package = "spatstat.data")
pp = data.frame(gorillas,
    lapply(gorillas.extra,function(x){x[gorillas]}),pt=1,wt=1e-6)
q_xy = data.frame(gorillas.extra[[1]])[,c("x","y")] # extract x and y from window
quad = data.frame(q_xy,lapply(gorillas.extra,function(x){x[q_xy]}),
    pt=0,wt=area(gorillas$window)/nrow(q_xy))
dat = merge(pp, quad, all=T)

# center and scale covariates
dat$elevation <- scale(dat$elevation)
dat$slopeangle <- scale(dat$slopeangle)
dat$waterdist <- scale(dat$waterdist)

head(dat)
##          x        y aspect  elevation     heat slopeangle slopetype vegetation
## 1 580455.7 676812.9      W -0.4254230 Moderate -1.3677928    Valley  Grassland
## 2 580455.7 676843.6     NW -0.4305752  Warmest -1.0094739  Midslope  Grassland
## 3 580455.7 676874.3     NW -0.5027051  Coolest -0.8500476    Valley    Primary
## 4 580486.4 676536.5     NW -0.3945102 Moderate -0.2545790    Valley  Disturbed
## 5 580486.4 676567.2      W -0.5078572 Moderate  0.8568781    Valley  Disturbed
## 6 580486.4 676597.9      W -0.5233137  Warmest  0.5965742    Valley  Disturbed
##    waterdist pt       wt group season date
## 1  1.2223995  0 944.4757  <NA>   <NA> <NA>
## 2  0.9175975  0 944.4757  <NA>   <NA> <NA>
## 3  0.6493147  0 944.4757  <NA>   <NA> <NA>
## 4 -0.9844239  0 944.4757  <NA>   <NA> <NA>
## 5 -0.8152447  0 944.4757  <NA>   <NA> <NA>
## 6 -0.4795709  0 944.4757  <NA>   <NA> <NA>
```

## S2.2 Choose an appropriate basis dimension

First, we fit a LGCP to the data using a couple of candidate basis dimensions, $k = 200, 400$ and then check the effective degrees of freedom to determine the adequacy of $k$. For demonstrative purposes we fit linear terms for covariates for now.

```
m_k200 <- gam(pt/wt ~ elevation + waterdist + slopeangle + heat + slopetype + vegetation +
    s(x, y, bs = "gp", k = 200),
    data=dat, family=poisson(), weights=wt, method="REML")
m_k400 <- gam(pt/wt ~ elevation + waterdist + slopeangle + heat + slopetype + vegetation +
    s(x, y, bs = "gp", k = 400),
    data=dat, family=poisson(), weights=wt, method="REML")
```

Then, check the ratio of effective degrees of freedom (of the total model) to the basis dimension used:

```
sum(m_k200$edf)/m_k200$smooth[[1]]$margin[[1]]$bs.dim
## [1] 0.3240267
sum(m_k400$edf)/m_k400$smooth[[1]]$margin[[1]]$bs.dim
## [1] 0.1839065
```

Since we are working off a rule of $\frac{1}{4}$, we decide that $k = 400$ is sufficient. There is probably some $200 < k < 400$ that would also meet this threshold — we do not explore this here, however obtaining a smaller $k$ may be useful to reduce computation times for subsequent analyses, such as estimating spatial range and/or model selection.

## S2.3 Estimate the spatial range parameter

The above models were fitted with the default range parameter which we can optimize by minimizing the REML criterion over a candidate interval, using `optimize()`. We recommend the candidate interval be the minimum (excluding zero) to maximum between-point distances found in the point pattern. There are a variety of ways to calculate such a range — we have found the `rdist()` function from the `fields` package to be the fastest approach.

```
# determine the range of between-point distances
dists <- fields::rdist(dat[dat$pt == 1, c("x", "y")])
range_interval <- range(dists[dists != 0])
# set up the function to be minimized
objective_fn = function(rho) {
    tmp.m = gam(
        pt/wt ~ elevation + waterdist + slopeangle + heat + slopetype + vegetation +
        s(x, y, bs = "gp", k = 400, m = c(3,rho)),
        data=dat, family=poisson(), weights=wt, method="REML")
    return(tmp.m$gcv.ubre) # the "method" specific criterion
}
# find the optimized range parameter
opt = optimize(objective_fn, interval = range_interval)
```

This gives an estimated $\rho$ that can be retrieved as:

```
opt$minimum
## [1] 568.5373
```

We can then fit the final model — using linear terms and default the Matérn covariance function (Kammann and Wand, 2003) with the estimated spatial range:

```
m <- gam(pt/wt ~ elevation + waterdist + slopeangle + heat + slopetype + vegetation +
    s(x, y, bs = "gp", k = 400, m = c(3, opt$minimum)),
    data=dat, family=poisson(), weights=wt, method="REML")
```

## S2.4 Model Diagnostics

We can use an inhomogeneous K function to examine the validity of the Poisson assumption by interfacing with functionality from the `spatstat` library to:

- predict the intensity over a fine enough resolution grid of the domain (so that it can form a pixel image/raster)
- convert the vector of predictions into an `im` object for interfacing with `spatstat`
- simulate point patterns over the domain from the `im` object to create a simulation envelope

Here we have used a dense, regular grid of the domain as quadrature points so we can use these to form a pixel image of the intensity surface to simulate point patterns from. The following also fits an inhomogeneous Poisson process (IPP) for comparison. Note that an IPP can be fitted via `glm()` (as presented in the manuscript), or via `gam()`. We use the latter here as we wish to later compare these models to an IPP fitted with smooth terms on covariates.

```
# fit an IPP model to contrast with the fitted LGCP
m_ipp <- gam(pt/wt ~ elevation + waterdist + slopeangle + heat + slopetype + vegetation,
    data=dat, family=poisson(), weights=wt, method="REML")

# set the domain data points (in this case the quadrature we used)
domain.grid <- dat[dat$pt == 0, ]

# predict intensity values
domain.grid$z_ipp = predict(m_ipp, newdata=domain.grid, type = "response")
domain.grid$z = predict(m, newdata=domain.grid, type = "response")

# create the pixel images (uses the window supplied in the original gorillas data)
pred_ipp.im = as.im(domain.grid[,c("x","y","z_ipp")], W = gorillas$window)
pred.im = as.im(domain.grid[,c("x","y","z")], W = gorillas$window)

# calculate the observed K functions
K_obs_ipp <- Kinhom(gorillas, lambda = pred_ipp.im, correction = "border")
K_obs <- Kinhom(gorillas, lambda = pred.im, correction = "border")

# simulate the envelopes/bounds
K_env_ipp <- envelope(gorillas, fun = Kinhom,
    simulate = expression(rpoispp(lambda = pred_ipp.im)))
K_env <- envelope(gorillas, fun = Kinhom,
    simulate = expression(rpoispp(lambda = pred.im)))
```

The observed K functions use a "border" correction, see Baddeley and Turner (2000) for details. The K functions and corresponding envelopes can be plotting using the following code which produces Figure 2 in the main manuscript.

```
layout(mat=matrix(1:2, nrow=2, ncol=1, byrow=TRUE), widths=1, heights=c(0.5,0.5))

par(mar = c(2.1,3.1,2.1,0))
plot(K_env_ipp$r, K_env_ipp$mmean, type="n",
    ylim=range(c(K_env_ipp$obs, K_env_ipp$hi, K_env_ipp$lo)), ylab="", xlab="",
    xaxt="n", yaxt="n")
axis(side = 2, at = seq(0,3e6,by=1e6), labels = c("0", seq(1e6,3e6,by=1e6)))
polygon(c(rev(K_env_ipp$r), K_env_ipp$r), c(rev(K_env_ipp$hi), K_env_ipp$lo),
    col="grey80", border=NA)
lines(K_env_ipp$r, K_env_ipp$mmean, lty="dashed")
lines(K_obs_ipp$r, K_obs_ipp$border, col="red")
mtext(text="A: Poisson Process", side=3, cex=1, line=0.5, adj=0)

par(mar = c(3.1,3.1,1.1,0))
plot(K_env$r, K_env$mmean, type="n", ylim = range(c(K_env$obs, K_env$hi, K_env$lo)),
     ylab="", xlab="", yaxt="n")
axis(side=2, at=seq(0,3e6,by = 1e6), labels = c("0", seq(1e6,3e6,by=1e6)))
polygon(c(rev(K_env$r), K_env$r), c(rev(K_env$hi), K_env$lo), col="grey80", border=NA)
lines(K_env$r, K_env$mmean, lty="dashed")
lines(K_obs$r, K_obs$border, col="red")
mtext(text = "distance (m)", side=1, srt=90, cex=1, line=2)
mtext(text = "Inhomogeneous K Function", side=2, srt=90, cex=1, xpd=T, outer=T, line=-1)
mtext(text = "B: log-Gaussian Cox Process", side=3, cex=1, line=0.5, adj=0)
```

```
legend(x=0, y=4e6, legend = c("Observed", "Theoretic", "95% Sim. Bounds"),
       col = c("red", "black", "grey80"), lty = c("solid", "dashed", "solid"), cex=1,
       lwd=c(2, 2, 2), bty="n")
```

The IPP model appears to violate the Poisson assumption across all distances whereas, the LGCP is within the simulation envelope at short ranges, *i.e.* distances $<$ 1km roughly. Note that we use simulation envelopes here for demonstration however, a formal (global) envelope test can be performed used the `GET` package (Myllymäki and Mrkvička, 2019).

## S2.5 Model Selection

We can perform model selection using some information criterion, such as AIC. First, we fitted second-order polynomial terms on the covariates since we might expect gorillas to have a preferred (or avoided) elevation, gradient and distance from water, rather than linear relationships to these.

```
# fit an IPP for comparison
m_ipp_poly <- gam(pt/wt ~ poly(elevation, 2) + poly(waterdist, 2) + poly(slopeangle, 2) +
    heat + slopetype + vegetation , data=dat, family=poisson(), weights=wt, method="REML")
# fit the LGCP
m_poly <- gam(pt/wt ~ poly(elevation, 2) + poly(waterdist, 2) + poly(slopeangle, 2) +
    heat + slopetype + vegetation +
    s(x,y, bs = "gp", k = 400, m = c(3, opt$minimum)),
    data=dat, family=poisson(), weights=p.wt, method="REML")
```

We then fitted the covariates more flexibly using smooths within the GAM framework:

```
# fit an IPP for comparison
m_ipp_sm <- gam(pt/wt ~ s(elevation) + s(waterdist) + s(slopeangle) + heat + slopetype +
    vegetation, data=dat, family=poisson(), weights=wt, method="REML")
# fit the LGCP
m_sm <- gam(pt/wt ~ s(elevation) + s(waterdist) + s(slopeangle) + heat + slopetype +
    vegetation + s(x, y, bs = "gp", k = 400, m = c(3, opt$minimum)),
    data=dat, family=poisson(), weights=wt, method="REML")
```

We can compare a range of information criteria between the models that use linear, polynomial and smooth terms for covariates, as well as those that with (or without) a latent field:

```
info_crit <- function(x){c(`GAM criterion`=ifelse(is.null(x$gcv.ubre),NA,x$gcv.ubre),
    logLik = logLik(x), AIC = AIC(x), BIC = AIC(x, k = log(sum(dat$pt))))}
data.frame(`IPP Linear`=info_crit(m_ipp), `IPP Poly`=info_crit(m_ipp_poly),
    `IPP Smoooth`=info_crit(m_ipp_sm), `LGCP Linear` = info_crit(m),
    `LGCP Poly` = info_crit(m_poly), `LGCP Smoooth` = info_crit(m_sm))
```

```
##                IPP.Linear  IPP.Poly IPP.Smoooth LGCP.Linear LGCP.Poly LGCP.smooth
## GAM criterion   15239.06  15189.40    15186.14    14816.92  14790.53    14816.80
## logLik         -15222.26 -15196.00   -15157.49   -14673.91 -14672.70   -14673.35
## AIC             30476.53  30429.99    30362.40    29526.57  29529.30    29531.85
## BIC             30548.09  30514.97    30468.44    29926.27  29940.54    29945.90
```

Results show that there is strong support for the inclusion of the latent field (via a LGCP, *i.e.* including a bivariate smoother on coordinates) whereas, there is little support for the use of polynomial or smooth terms on the covariates.

# References


Baddeley, A. and Turner, R. (2000), "Practical Maximum Pseudolikelihood for Spatial Point Patterns: (with Discussion)," *Australian & New Zealand Journal of Statistics*, 42, 283–322.

Brooks, M. E., Kristensen, K., Van Benthem, K. J., Magnusson, A., Berg, C. W., Nielsen, A., Skaug, H. J., Machler, M., and Bolker, B. M. (2017), "`glmmTMB` balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling," *The R Journal*, 9, 378–400.

Datta, A., Banerjee, S., Finley, A. O., and Gelfand, A. E. (2016), "Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets," *Journal of the American Statistical Association*, 111, 800–812.

Dovers, E., Brooks, W., Popovic, G. C., and Warton, D. I. (2023), "Fast, Approximate Maximum Likelihood Estimation of Log-Gaussian Cox Processes," *Journal of Computational and Graphical Statistics*, 32, 1660–1670.

Fuglstad, G.-A., Simpson, D., Lindgren, F., and Rue, H. (2019), "Constructing priors that penalize the complexity of Gaussian random fields," *Journal of the American Statistical Association*, 114, 445–452.

Funwi-Gabga, N. and Mateu, J. (2012), "Understanding the nesting spatial behaviour of gorillas in the Kagwene Sanctuary, Cameroon," *Stochastic Environmental Research and Risk Assessment*, 26, 793–811.

Illian, J. B., Sørbye, S. H., and Rue, H. (2012), "A toolbox for fitting complex spatial point process models using integrated nested Laplace approximation (INLA)," *The Annals of Applied Statistics*, 6, 1499–1530.

Kammann, E. and Wand, M. P. (2003), "Geoadditive models," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 52, 1–18.

Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H., and Bell, B. M. (2016), "`TMB`: Automatic Differentiation and Laplace Approximation," *Journal of Statistical Software*, 70, 1–21.

Myllymäki, M. and Mrkvička, T. (2019), "`GET`: Global envelopes in `R`," *arXiv preprint arXiv:1911.06583*.

Shanno, D. F. (1970), "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, 24, 647–656.

Simpson, D., Illian, J., Lindgren, F., Sørbye, S. H., and Rue, H. (2016), "Going off grid: computationally efficient inference for log-Gaussian Cox processes," *Biometrika*, 103, 49–70.

Wood, S. N. (2004), "Stable and efficient multiple smoothing parameter estimation for generalized additive models," *Journal of the American Statistical Association*, 99, 673–686.

— (2011), "Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73, 3–36.

— (2017), *Generalized additive models: an introduction with R*, CRC press.

Youngman, B. D. and Economou, T. (2017), "Generalised additive point process models for natural hazard occurrence," *Environmetrics*, 28, e2444.