Regridding and interpolation of climate data for impacts modelling — some cautionary notes

Online supplement

Richard E. Chandler[1] Clair Barnes[1] Chris Brierley[2] Raquel Alegre[3]

University College London
[1] *Department of Statistical Science* [2] *Department of Geography*
[3] *Research Software Development Group*
Contact: r.chandler@ucl.ac.uk

EGU General Assembly 2022, Session HS7.1

Contents

1 Introduction and setup

2 Main presentation example 2: simulation experiment for daily precipitation extremes

3 One possible solution: multiple imputation

4 More surprises? Impacts model calibration

Introduction and setup

About this document

This presentation has been created using an `RMarkdown` script `EGU22-4004_CodeSupplement.Rmd`: the script and associated files can be downloaded in a `.zip` archive, so that users can reproduce the results and experiment with alternative scenarios.

The material supplements the presentation EGU22-4004 'Regridding and interpolation of climate data for impacts modelling — some cautionary notes', given at the annual meeting of the EGU in May 2022.

In the script, the code chunks are written in R. The `PDF` presentation is produced by running the script from within RStudio.[1] A working LaTeX installation is required to produce the presentation — guidance is here if needed. Alternatively, the code chunks can be run separately to produce output within the `RStudio` console.

[1] Attempts to produce alternative output formats may not work 100%, because the `RMarkdown` script contains LaTeX code.

Requirements

R libraries

The script uses the following libraries:

- ismev, geoR: these can be installed from CRAN, although the easiest way to do it is via the `Install Packages` option from the `Tools` menu in `RStudio`.
- Rglimclim: this is not available from CRAN, but must be downloaded from the Rglimclim home page. `Windows` users should download the `.zip` distribution, all others should use the `.gz` version. Then use the `Install Packages` option from the `Tools` menu in `RStudio`, and select `Package Archive File` from the "Install from" drop-down menu.

Data files (all provided in the accompanying `.zip` archive)

The following files should be present in the working directory:

- siteinfo.dat: contains coordinates of locations used in simulation experiments.
- logistic.def, gammamdl.def: `Rglimclim` model definition files, needed to produce simulated precipitation sequences.
- BeamerHeader.tex, TexHeader.tex, beamercolorthemeUCLslides.sty: provide additional LaTeX functionality needed to compile `Markdown` to either a `Beamer` presentation or `PDF` document

Main presentation example 2: simulation experiment for daily precipitation extremes

Recap: purpose of the example

- Simulate 30-year “daily precipitation sequences” at 12 locations
- Use kriging to create gridded daily dataset from simulations
  - Regular grid: 12 nodes
- Compare annual maxima / GEV return levels for original & gridded data

Design considerations

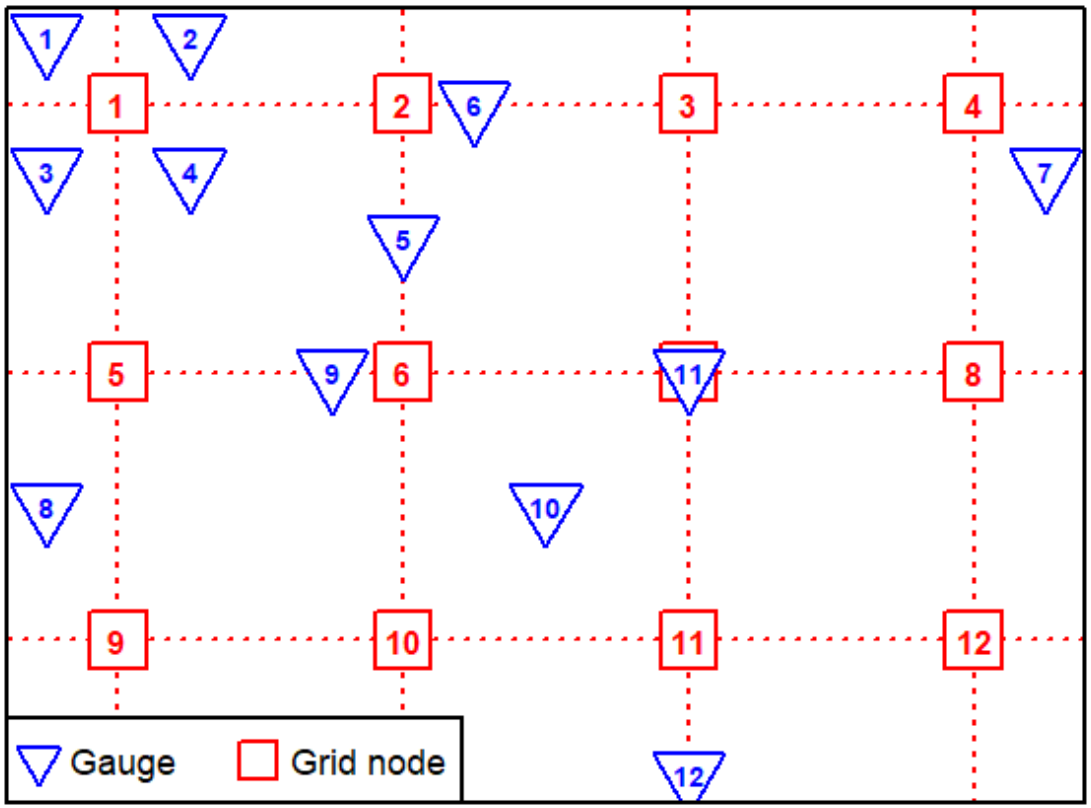
- Equal numbers of stations and grid nodes: expect gridding to have limited effect?
- "Station" locations chosen to illustrate various scenarios e.g.
  - Grid node 7 coincides with station 11
  - For grid node 1, four closest stations are equidistant and close
  - Grid node 12 has no stations within neighbouring squares
- Use simplified but realistic simulation structure:
  - Based on model of Yang et al. (*Water Resources Research*, 2005, doi:10.1029/2004WR003739) for 50km $\times$ 40km region of southern England
  - Regional & topographic effects removed from original model: ‘Flatland’ version where all locations experience identical precipitation regimes
  - Inter-site dependence ensures that $\sim$ 75% of days have sites all wet or all dry, wet-day inter-site correlations $\sim$ 0.6–0.95.

R code: load libraries and define site / grid locations

```
library(Rglimclim)   # For generation of synthetic data
library(geoR)        # For interpolation
library(ismev)       # For extreme value analysis
set.seed(1111)       # Set random number sequence to repeatable initial state
#
#   Next commands read locations from file, & convert to format
#   needed by Rglimclim. NB "regions" in Rglimclim define groups
#   of locations: used here to distinguish between "sites" and
#   "grid nodes"
#
site.data <- read.table("siteinfo.dat",header=TRUE)
sites <- (site.data$Region ==1) # Logical vector
StationInfo <-
  make.siteinfo(site.data[sites,], coord.cols = 2:3,
                site.names=rownames(site.data[sites,]),
                attr.names = c("East", "North"), region.col = 1)
print(StationInfo)
12 sites defined, each with the following attributes:

1. East
2. North
```

Map of locations

- `R` code suppressed in output: see `.Rmd` file for details



Steps in simulation exercise

NB results will vary between simulations due to stochastic nature of simulation model - edit `set.seed(1111)` line of `.Rmd` file for alternative realisations

1. Simulate 30 years of daily data at 'station' locations
   a. Define models to be used for simulation
   b. Simulate
2. Interpolate to regular grid
   a. Identify appropriate spatial model (**NB** without reference to model used for simulation)
   b. Use model to interpolate to grid nodes
3. Estimate specified return levels from original and regridded simulations
   a. Extract 720 annual daily maxima (30 years, 12 stations + 12 grid nodes) & plot their distributions across simulations (**NB** 'Flatland' scenario ensures distributions should be identical at all locations)
   b. Fit Generalised Extreme Value (GEV) distributions separately to 360 'station' maxima & 360 maxima from gridded data
   c. Calculate return levels from fitted distributions

Step 1a: simulation models

- Based on Generalized Linear Models (GLMs) of Yang et al. (*Water Resources Research*, 2005, doi:10.1029/2004WR003739), modified for Flatland
  - Separate models for daily precipitation occurrence and wet-day amounts
  - **Occurrence model**: let p_{st} be probability of non-zero rainfall at location s and day t, then

$$\log\left(\frac{p_{st}}{1-p_{st}}\right) = \beta_0 + \beta_1 x_{1st} + \ldots + \beta_p x_{pst}$$

    where $x_{1st}, \ldots, x_{pst}$ are values of p associated covariates ('predictors')
  - **Wet-day amounts model**: if location s is wet on day t then amount is drawn from gamma distribution with constant shape parameter ν and varying mean μ_{st}, with

$$\log \mu_{st} = \gamma_0 + \gamma_1 \xi_{1st} + \ldots + \gamma_q \xi_{qst}$$

    where now $\xi_{1st}, \ldots, \xi_{qst}$ are covariates (possibly different from x's above)
  - Interactions allow covariate to modulate each other's effects
  - Small non-zero values handled separately; and inter-site dependence incorporated via different schemes for occurrence and amounts[2]
- Implementation in `Rglimclim` (Chandler 2020, *Env. Mod. Software*, doi:10.1016/j.envsoft.2020.104867)

[2] Inter-site dependence model for occurrence updated since Yang et al. 2005

Occurrence model

```
DAILY RAINFALL OCCURRENCE MODEL FOR SIMULATION
==============================================

Response variable: Precipitation (mm)

Main effects:
-------------
                                            Coefficient
                                  Constant      -1.1693
1       Mean of I(Precipitation (mm)[t-1]>0)       1.4130
2       Daily annual cycle, cosine component       0.2631
3         Daily annual cycle, sine component      -0.0554

Two-way interactions:
---------------------
                                            Coefficient
        Mean of I(Precipitation (mm)[t-1]>0)      -0.1575
    with Daily annual cycle, cosine component
        Mean of I(Precipitation (mm)[t-1]>0)       0.0652
      with Daily annual cycle, sine component

Global quantities:
------------------
     'Soft' threshold for +ve values:      0.5000

No dispersion parameters defined

Spatial dependence structure:
-----------------------------
Structure used: Exponential correlation function: exp
     Correlation decay rate, phi:      0.0325

**** NOTE: distance-dependent weights (for averages of previous days'
values and / or correlations) are based on distances calculated from
     East
and North
```

Amounts model

```
DAILY RAINFALL AMOUNTS MODEL FOR SIMULATION
===========================================

Response variable: Precipitation (mm)

Main effects:
-------------
                                          Coefficient
                                Constant      1.6823
1       Ln(1+Mean of Precipitation (mm)[t-1])      0.1156
2        Daily annual cycle, cosine component     -0.1226
3          Daily annual cycle, sine component     -0.1198

Two-way interactions:
---------------------
                                          Coefficient
        Ln(1+Mean of Precipitation (mm)[t-1])      0.0665
    with Daily annual cycle, cosine component
        Ln(1+Mean of Precipitation (mm)[t-1])     -0.0341
      with Daily annual cycle, sine component

Global quantities:
------------------
     'Soft' threshold for +ve values:      0.5000

Dispersion parameter:      0.7221

Spatial dependence structure:
-----------------------------
Structure used: Exponential correlation function: exp
     Correlation decay rate, phi:      0.1250

**** NOTE: distance-dependent weights (for averages of previous days'
values and / or correlations) are based on distances calculated from
     East
and North
```

Step 1b: simulation code

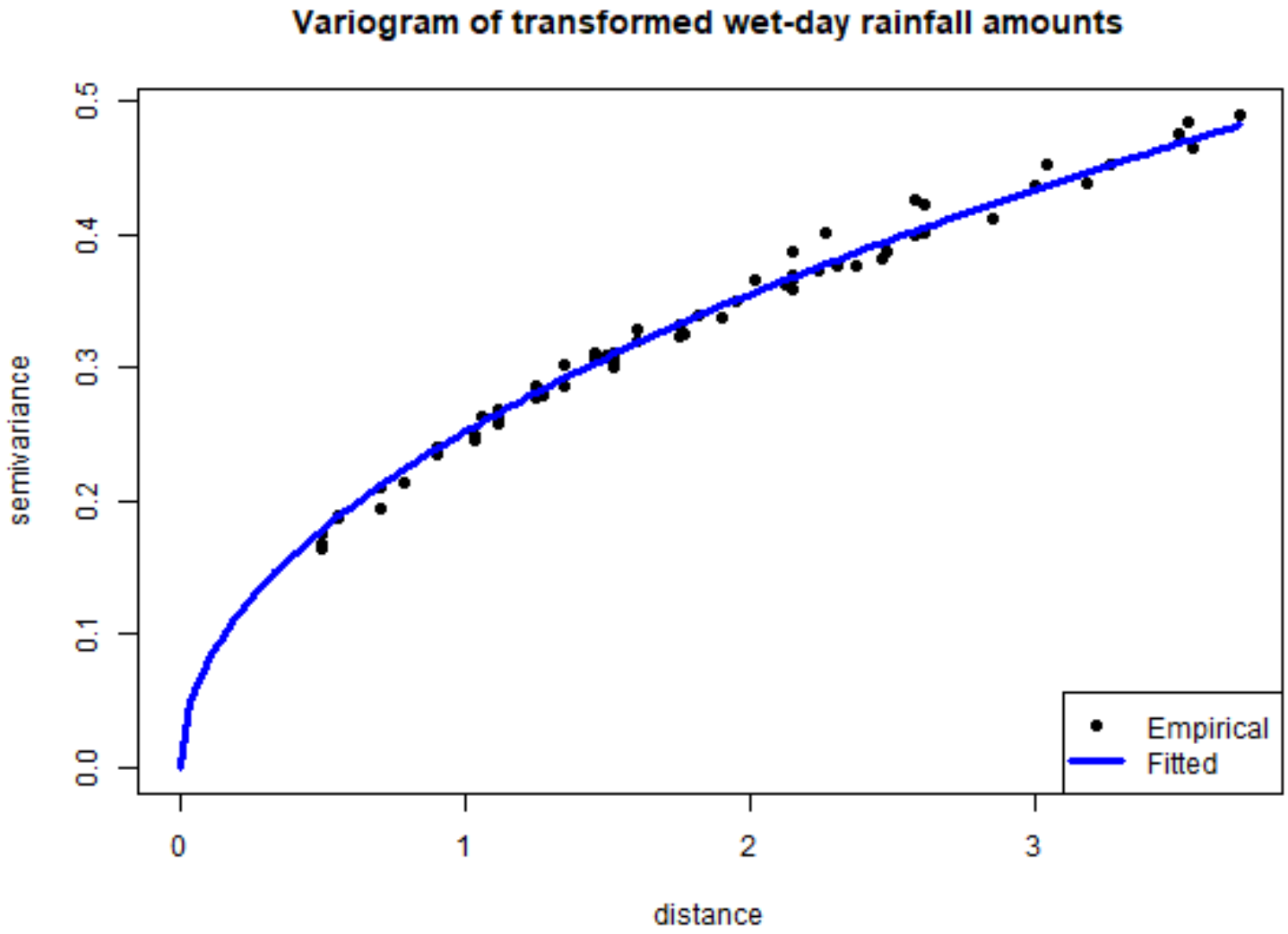
```
#
#   Rglimclim wants a data file from which to initialise simulations:
#   produce a dummy one, then simulate 31-year labelled with an arbitrary
#   set of dates & discard the first year as a "spin-up", to ensure
#   that the retained sample is not influenced by initial values.
#
write.GLCdata(data.frame(Year=1900, Month=1, Day=1, Site="S001", Rain=0),
              file="DummyData.dat", check.file=FALSE)
SimDir <- "PrecipSim"
if (dir.exists(SimDir)) unlink(SimDir, recursive =TRUE)
SimPrecip <-
  GLCsim(list(Occurrence=OccModel, Intensity=AmountsModel),
         StationInfo, start=196901, end=199912, nsims=1,
         impute.until = 190001, output="daily", daily.start = 197001,
         simdir=SimDir, file.prefix="Sim", data.file="DummyData.dat")

print(SimPrecip)
Object of class GLCsim:
========================

Variables taken from data file DummyData.dat
Variables simulated:
    1.  Precipitation (mm) (model type: logistic-gamma)

Simulation period: 1/1969 to 12/1999
No imputation performed
1 realisations generated

Output files generated: daily
Daily output written from 1/1970 to 12/1999
Output directory: C:/Users/richard/Documents/RESEARCH/PROJECTS/UKMO_EUROCORDEX/EGU2022/REC_TALK/SUPPLEMENT/PrecipSim
Prefix for output filenames: Sim
```

Step 2a: identify spatial model for interpolation

Approach is based on kriging, using variogram for days with at least one wet site

```
#
# Read daily simulation file, & reshape to give one row per day.
# The "date" variable is used to create a unique index for each
# day, which is only needed for the reshape command - hence
# deleted afterwards.
#
daily.sim <- read.GLCdata(paste(SimDir, "Sim_Daily_Sim1.dat", sep="/"))
daily.sim$Date <- (1e4*daily.sim$Year) + (100*daily.sim$Month) + daily.sim$Day
daily.sim <- reshape(daily.sim, direction="wide", v.names="Var1",
                     idvar="Date", timevar="Site")[,-4]
names(daily.sim)[-(1:3)] <- rownames(site.data)[sites]
#
# Calculate separate variograms for each day with at least one wet
# station (if it's zero everywhere then the kriging predictor will
# be zero); then average to obtain an overall variogram.
#
wetdays <- rowSums(daily.sim[,-(1:3)]) > 0
alpha <- 1/3 # Do kriging on cube-root scale (more closely matches assumptions)
vgram <- variog(coords=site.data[sites,-1],
                data=t(daily.sim[wetdays,-(1:3)]^alpha), option="cloud")
vgram$v <- rowMeans(vgram$v) # Average the separate daily variograms
vgram$var.mark <- mean(vgram$var.mark) # Overall variance of data
vgram$beta.ols <- mean(vgram$beta.ols) # Overall mean(!)
#
# Fit model directly to sample variogram. Although the data were
# generated using exponential correlation structures (for latent fields
# and Anscombe residuals in occurence and amounts models respectively),
# this does *not* mean that an exponential model will be the best fit
# to the variations in cube-root precipitation. Experimentation
# suggests that a powered exponential model fits well.
#
vgram.fitted <- variofit(vgram,cov.model="stable",fix.nugget=TRUE,
                         nugget=0,ini.cov.pars=c(1,1))
```

Empirical and fitted variograms

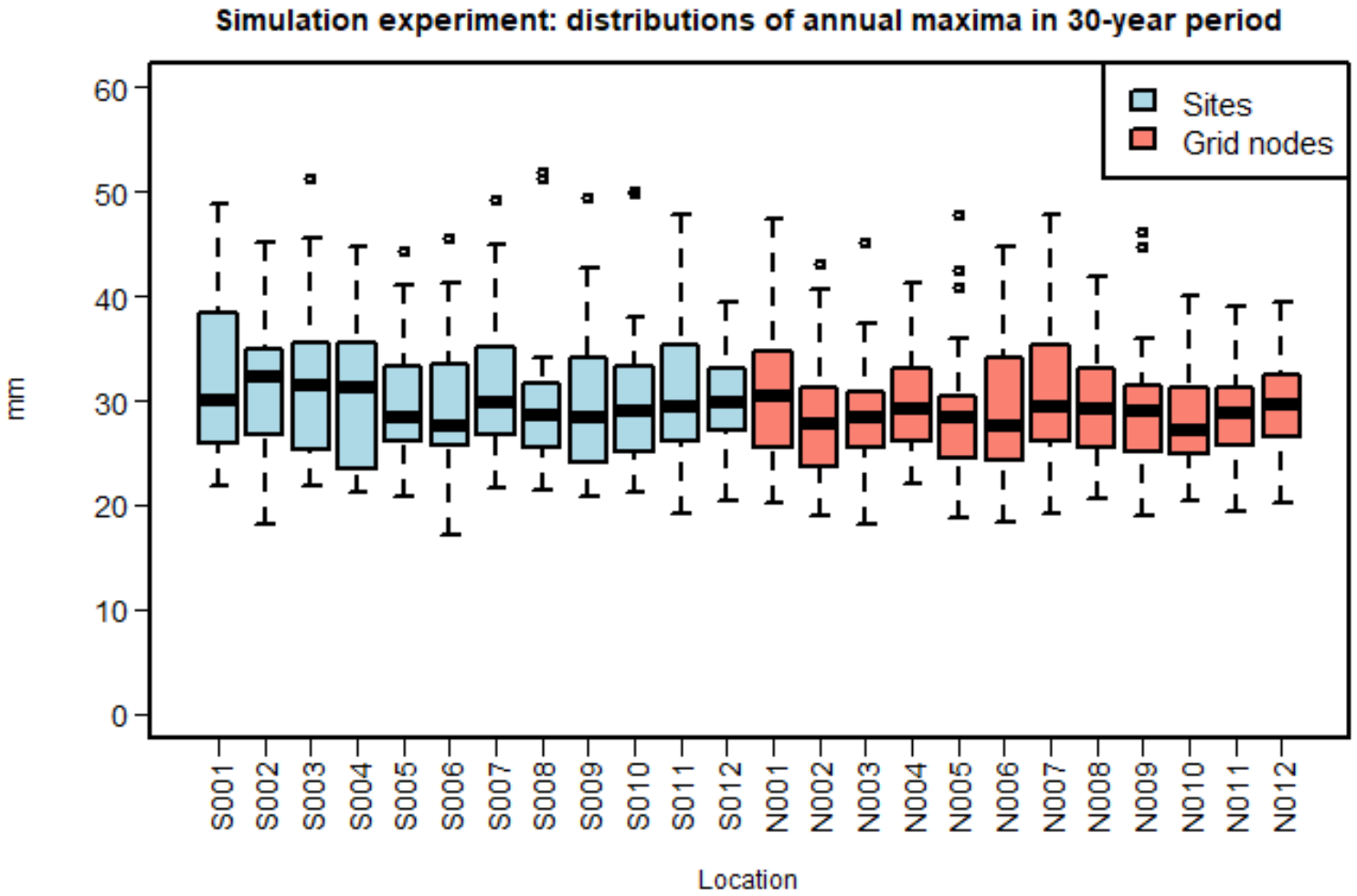


Step 2b: Interpolation code

```
#
#   Create storage for gridded data by copying daily.sim, renaming
#    columns & setting precip values to NA on any non-dry day
#
daily.grid <- daily.sim
names(daily.grid) <- c("Year","Month","Day",rownames(site.data)[!sites])
daily.grid[wetdays,-(1:3)] <- NA
#
#   Now do the kriging, separately for each day, to estimate the
#   cube-root daily values at the grid locations * back-transform.
#
krige.def <- krige.control(type.krige="ok",obj.model=vgram.fitted)
for (i in which(wetdays)) {
  daily.grid[i,-(1:3)] <-
    krige.transform(coords=site.data[sites,-1], data=t(daily.sim[i,-(1:3)]),
               locations=site.data[!sites,-1], krige=krige.def, alpha=alpha)
}
#
#   Merge estimates with "station" values. NB the merge command puts
#   the dates out of order: this is corrected in the subsequent line.
#
daily.data <- merge(daily.sim,daily.grid)
daily.data <- daily.data[order(daily.data$Year, daily.data$Month, daily.data$Day),]
print(head(daily.data, 3), digits=2) # Show what result looks like
   Year Month Day S001 S002 S003 S004 S005 S006 S007 S008
1  1970     1   1 0.86 0.71 0.86  1.2  1.0 0.98  1.4    0
12 1970     1   2 0.00 0.00 0.00  0.0  0.0 0.00  0.0    0
23 1970     1   3 0.00 0.00 0.00  0.0  0.0 0.00  0.0    0
   S009 S010 S011 S012 N001 N002 N003 N004 N005 N006 N007
1  0.93  1.7  1.1  2.1  1.3  1.5    2  2.1 0.69  1.5  1.1
12 0.00  0.0  0.0  0.0  0.0  0.0    0  0.0 0.00  0.0  0.0
23 0.00  0.0  0.0  0.0  0.0  0.0    0  0.0 0.00  0.0  0.0
   N008 N009 N010 N011 N012
1   2.4 0.59  1.6  2.4  2.8
12  0.0 0.00  0.0  0.0  0.0
23  0.0 0.00  0.0  0.0  0.0
```

Step 3a: distributions of annual maxima

- `R` code suppressed in output: see `.Rmd` file for details



Steps 3b & 3c: fit GEV distributions and calculate return levels

```
return.levels <- data.frame(matrix(nrow=3,ncol=nrow(site.data)+2))
names(return.levels) <- c(names(daily.data)[-(1:3)],"AllSites","AllNodes")
rownames(return.levels) <- c("10 year","50 year","100 year")
quant <- 1/c(10,50,100)
#
#   Separate GEV fits to data from each location
#
for (i in 1:nrow(site.data)) {
 gev.model <- gev.fit(daily.max[,i+1])
 return.levels[,i] <- gevq(gev.model$mle,quant)
}
#
#   Pooled fits to data from all sites ...
#
gev.model <- gev.fit(unlist(daily.max[,(1+(1:nrow(site.data))[sites])]))
gev.pars <- gev.model$mle
return.levels[,i+1] <- gevq(gev.model$mle,quant)
#
#   ... and to data from all grid nodes
#
gev.model <- gev.fit(unlist(daily.max[,(1+(1:nrow(site.data))[!sites])]))
return.levels[,i+2] <- gevq(gev.model$mle,quant)
#
#   Finally, calculate actual return periods for grid-based estimates
#
grid.periods <- data.frame("Period (years)" =
                           1 / (1-gevf(gev.pars,return.levels[,i+2])))
rownames(grid.periods) <- c("10 years","50 years","100 years")
```

Results from return-level estimation

Table 1: Return level estimates for stations

	S001	S002	S003	S004	S005	S006	S007	S008	S009	S010	S011	S012	AllSites
10 year	42.7	39.5	41.4	39.6	37.4	38.6	40.3	37.5	39.4	39.3	38.8	35.7	39.4
50 year	53.9	43.8	50.8	46.5	45.8	46.7	51.0	48.5	53.4	51.0	45.9	38.5	47.9
100 year	58.8	45.0	54.7	49.0	49.5	49.9	55.9	54.0	60.6	56.8	48.7	39.3	51.5

Table 2: Return level estimates for grid nodes

	N001	N002	N003	N004	N005	N006	N007	N008	N009	N010	N011	N012	AllNodes
10 year	39.6	35.9	36.3	37.5	36.8	37.5	38.8	37.0	36.8	36.3	35.1	35.6	37.1
50 year	47.7	43.8	42.1	44.5	44.9	45.7	45.9	42.6	43.9	43.2	38.8	39.3	43.7
100 year	50.9	47.1	44.2	47.4	48.4	49.1	48.7	44.6	46.8	46.0	40.0	40.5	46.4

Table 3: Actual return periods for gridded return-level estimates

	10 years	50 years	100 years
Period..years.	7	22	37

Comments on example

- Designed to be "fairly realistic"
- Results are not context-dependent: example illustrates fundamental mathematics
- **Experimentation encouraged!** E.g.:
  - Change the seed for the random number generator (search for `set.seed`)
  - Modify the models e.g. change the correlation decay parameter in `gammamdl.def`: larger values imply weaker inter-site dependence
  - Change the interpolation method (code it yourself!).
- **NB** results agree empirically with those of Liu et al. (2021), *Int. J. Climatol.*, doi:10.1002/joc.7389), although their conclusion is too strong.[3]

Take-home message

It's not difficult to find situations in which the use of gridded data can lead to *potentially* catastrophic underestimation of risk

[3] They write "Since none of the statistical interpolation methods provided good performance, a dynamical downscaling method ... is needed to downscale extreme events". Their results do not support this claim, but *do* support the statement "Since none of the approaches considered provided good performance, alternatives are needed to downscale extreme events".

One possible solution: multiple imputation

Recap: from main presentation summary

- Possible solution is to take multiple samples from uncertainty distribution at desired locations
  a. Removes bias if all samples are propagated through impacts models
  b. For computationally intensive impacts models, may need small number of carefully-chosen samples
  c. Doesn't fix the problem when calibrating impact models

Focus here on points 2a and 2c

- Continuing previous example, sample from distribution of daily precipitation at grid nodes
- Use `Rglimclim` directly (software allows imputation of 'missing' data)

Example 2 continued: gridded data via imputation

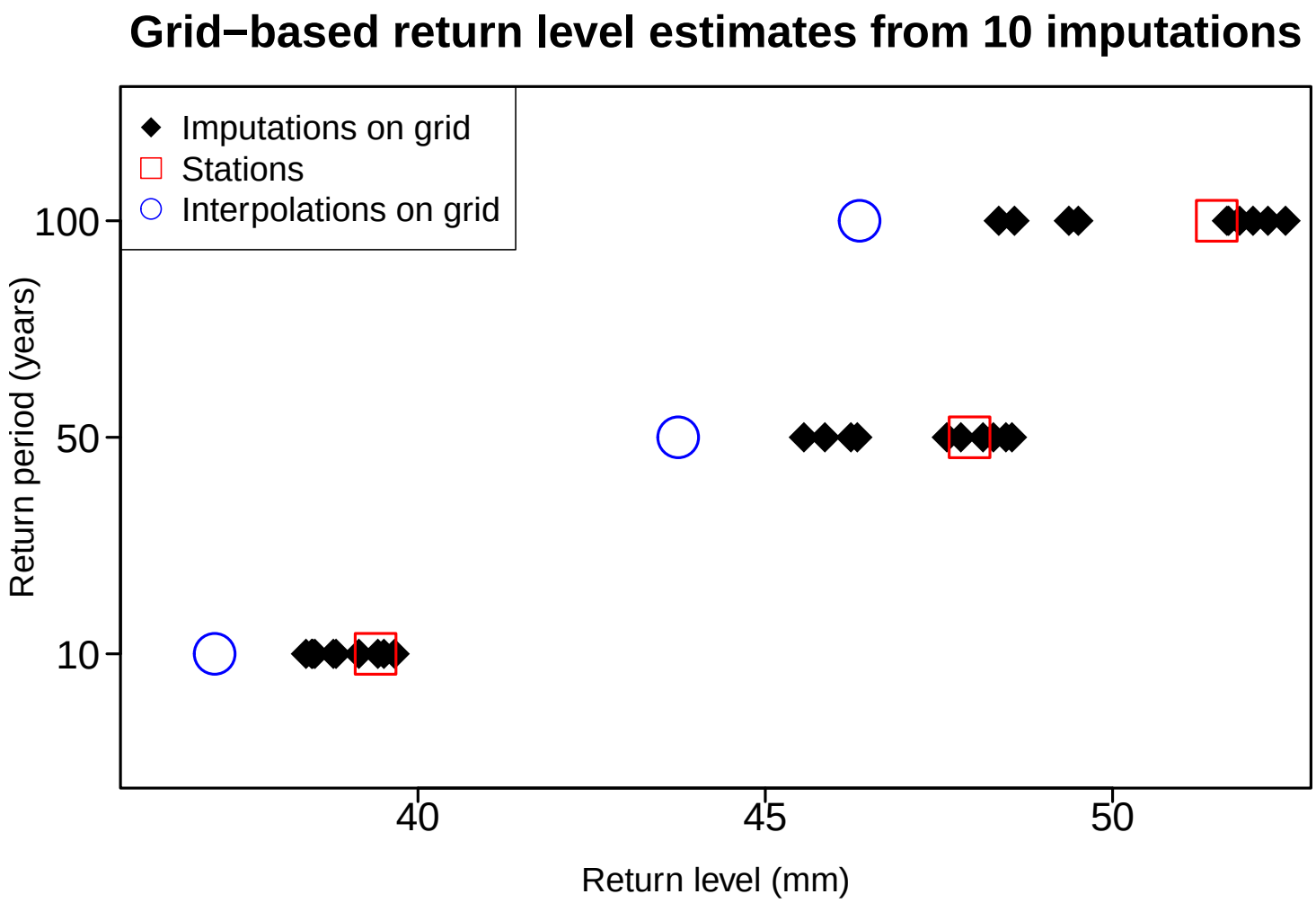
- 'Imputation' = sampling missing data from conditional distribution given available observations
- Multiple samples quantify uncertainty due to missing data if needed

Imputation for Example 2 in `Rglimclim`

```
#
#   Define Rglimclim object containing information on all sites
#   and grid nodes. As before, "regions" define groups of sites.
#   NB Rglimclim will not simulate separately for two co-located
#   sites: here, grid node 7 is the same as site 11 so I'll just
#   remove the grid node from the list of sites for the moment
#
UniqueSites <- !(rownames(site.data)=="N007")
AllSites.info <-
  make.siteinfo(site.data[UniqueSites,],
                coord.cols = 2:3, region.col=1,
                site.names=rownames(site.data)[UniqueSites],
                attr.names = c("East", "North"))
#
#   Here's the imputation code: takes previous simulated data &
#   carries out 10 imputations of values at missing locations
#
ImpDir <- "PrecipImputation"
if (dir.exists(ImpDir)) unlink(ImpDir, recursive =TRUE)
ImputedPrecip <-
  GLCsim(list(Occurrence=OccModel, Intensity=AmountsModel),
         AllSites.info, start=196901, end=199912, nsims=10,
         impute.until = 199912, output="daily", daily.start = 197001,
         simdir=ImpDir, file.prefix="Imp",
         data.file=paste(SimDir, "Sim_Daily_Sim1.dat",sep="/"))
```

Example 2: return level estimates for imputed data

- `R` code suppressed in output: see `.Rmd` file for details



More surprises? Impacts model calibration

Model calibration with uncertain inputs

A simple situation?

- Impacts model (e.g. rainfall runoff model) takes inputs $\boldsymbol{x}$ and produces outputs $\boldsymbol{y}$ given parameters $\boldsymbol{\theta}$: $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ say
- Models usually calibrated by choosing $\boldsymbol{\theta}$ to match data as closely as possible
  - Calibration usually done by optimising some objective function: $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{x})$ say
- Suppose impacts model requires gridded precipitation inputs $\boldsymbol{x}$ but only station values are available — what to do?
  - **Standard practice (perhaps)**: interpolate to required grid & treat results as $\boldsymbol{x}$ in optimisation
  - **Alternative**: use multiple imputations to capture uncertainty in $\boldsymbol{x}$; do optimisation with each one
    - Should propagate uncertainty in $\boldsymbol{x}$ through to $\boldsymbol{\theta}$, right?

Another simulation experiment

- Same layout of stations and grid nodes as before
- Consider very simple 'impacts model': $y = f(\boldsymbol{x}, \boldsymbol{\theta})$ where
  - $\boldsymbol{x}$ represents collection of daily precipitation amounts at 12 grid nodes
  - $\boldsymbol{\theta}$ has components θ_0 & θ_1
  - $f(\boldsymbol{x}, \theta) = \theta_0 + \theta_1 \overline{x}$ where $\overline{x}$ is mean of daily precipitation over grid nodes
- Assume model is correct (unreasonably favourable to modeller!)
- Use least-squares estimation to calibrate model i.e. estimate θ_0 and θ_1 from data on $\boldsymbol{x}$ and y
  - True values of $\boldsymbol{x}$ unobserved, just have station 'observations'

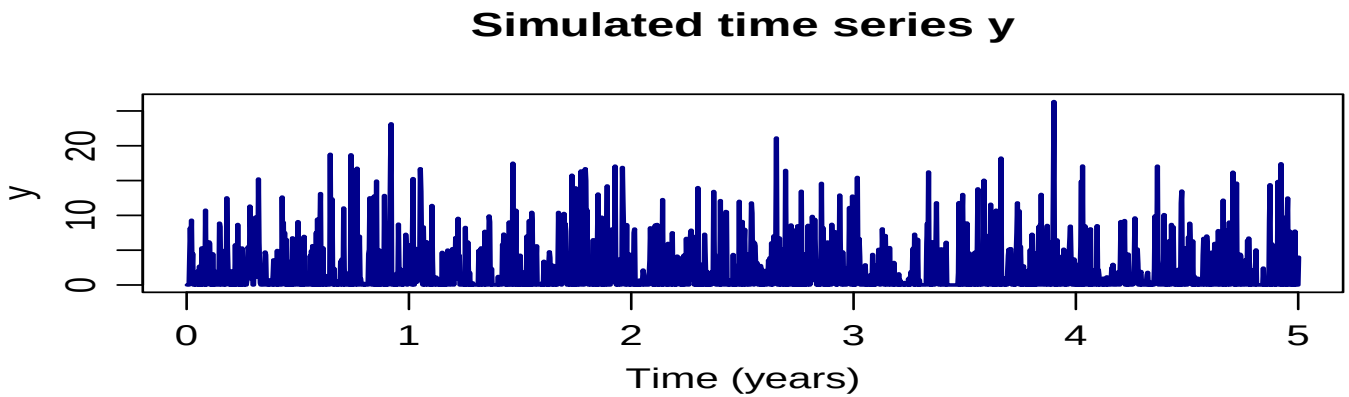
Experiment setup

1. Simulate 5 years of daily precipitation data at all stations and grid nodes[4].
2. **Define truth:** for each day, take simulated grid node values as $\boldsymbol{x}$ and compute $y = \overline{x}$ i.e. set $\theta_0 = 0$ & $\theta_1 = 1$.
   - Result: 5-year daily time series of system output y
3. Pretend grid node values are unavailable and try to estimate $\boldsymbol{\theta}$ using y together with *station* time series from step 1
   - **Option 1:** use kriging to estimate $\boldsymbol{x}$ from station values each day, & estimate $\boldsymbol{\theta}$ using least-squares regression of y on estimated $\boldsymbol{x}$
   - **Option 2:** use 10 sets of imputations of $\boldsymbol{x}$, & obtain separate least-squares estimates of $\boldsymbol{\theta}$ for each.
4. Compare estimates of θ_0 and θ_1 with known values of 0 and 1 respectively.

[4] Only 5 years here, to reduce the computing time

Steps 1 & 2: simulate daily precip & time series *y*

```
unlink(SimDir, recursive =TRUE) # Remove previous files
SimPrecip <-
  GLCsim(list(Occurrence=OccModel, Intensity=AmountsModel),
         AllSites.info, start=199901, end=200412, nsims=1,
         impute.until = 199801, output="daily", daily.start = 200001,
         simdir=SimDir, file.prefix="Sim", data.file="DummyData.dat")
#
#   Read the data back in, reshape to one row per day and
#   calculate "output" y
#
SimData <- Precip <-
  read.GLCdata(paste(SimDir, "Sim_Daily_Sim1.dat", sep="/"))
Precip$Date <- (1e4*Precip$Year) + (100*Precip$Month) + Precip$Day
Precip <- reshape(Precip, direction="wide", v.names="Var1",
                  idvar="Date", timevar="Site")[,-4]
names(Precip)[-(1:3)] <- rownames(site.data)[UniqueSites]
Precip$N007 <- Precip$S011 # As usual
theta0 <- 0; theta1 <- 1
y <- theta0 + (theta1 * rowMeans(Precip[,grep("N", names(Precip))]))
```



Step 3: estimate θ_0 and θ_1 using station data (option 1)

- Use existing variogram model, since underlying process hasn't changed

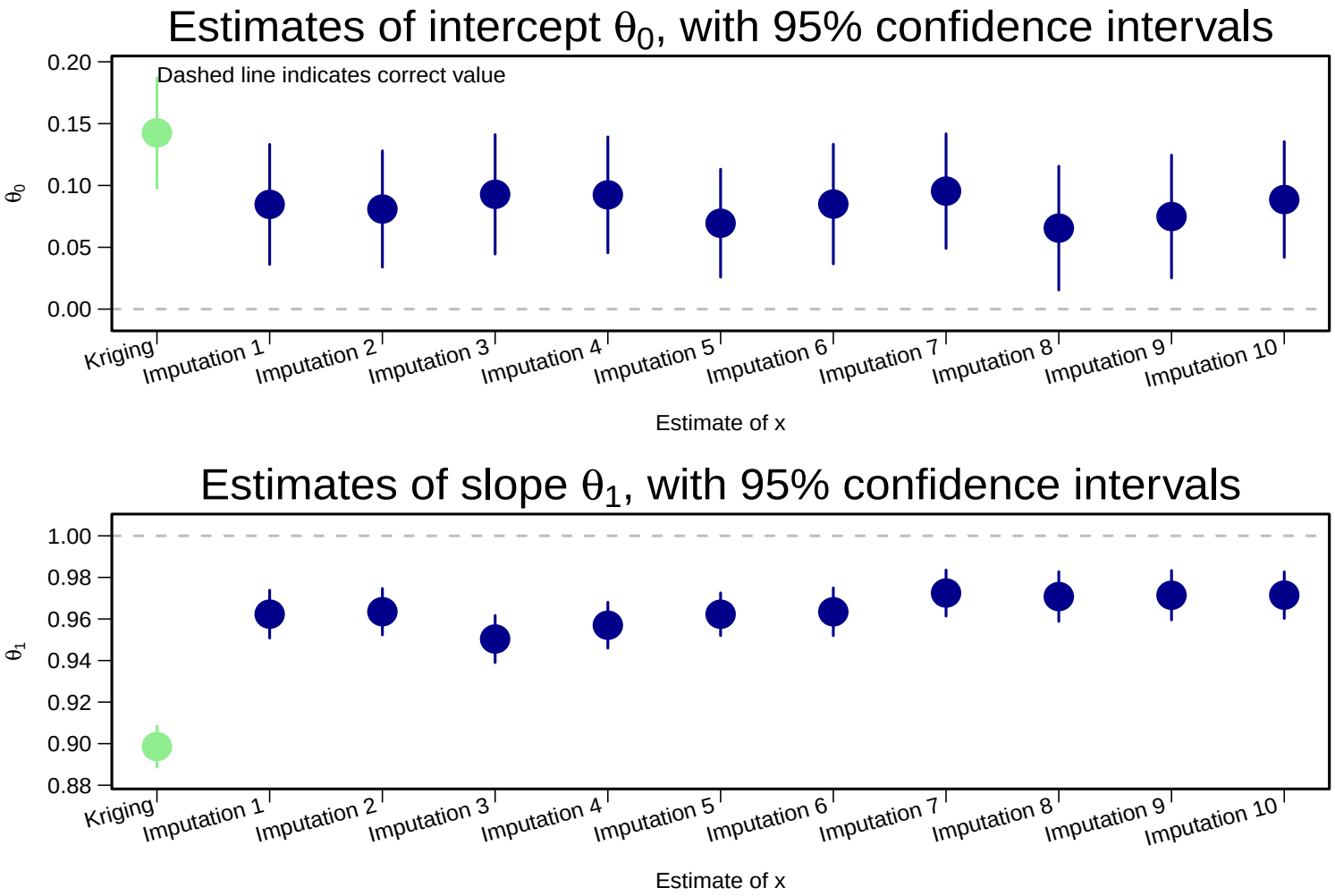
```
SiteCols <- grep("S", names(Precip))
wetdays <- (rowSums(Precip[,SiteCols]) > 0)
x.kriged <- matrix(0, nrow=nrow(Precip), ncol=12)
colnames(x.kriged) <- names(site.data)[!sites]
for (i in which(wetdays)) {
  x.kriged[i,] <-
    krige.transform(coords=site.data[sites,-1], data=t(Precip[i,SiteCols]),
                locations=site.data[!sites,-1], krige=krige.def, alpha=alpha)
}
#
#   Here's the least-squares estimation: store the
#   estimates and their confidence intervals
#
Model <- lm(y ~ rowMeans(x.kriged))
CI <- confint(Model) # 95% confidence intervals for parameters
Theta.Interp <- as.data.frame(matrix(c(coef(Model), CI[1,], CI[2,]),nrow=1))
names(Theta.Interp) <- c("Theta0", "Theta1", "LL95.0","UL95.0", "LL95.1", "UL95.1")
```

Step 3: estimate θ_0 and θ_1 using station data (option 2)

```
#
#   Start by writing data file containing only station data, to use for
#   imputation
#
write.GLCdata(SimData[grep("S", SimData$Site),],
              file="StationData.dat", check.file=FALSE)
unlink(ImpDir, recursive =TRUE) # Remove previous files
ImputedPrecip <-
  GLCsim(list(Occurrence=OccModel, Intensity=AmountsModel),
         AllSites.info, start=199901, end=200412, nsims=10,
         impute.until = 200412, output="daily", daily.start = 200001,
         simdir=ImpDir, file.prefix="Imp", data.file="./StationData.dat")
#
#   Write a function to do least-squares estimation using a single
#   imputation, and apply it to each file
#
Calibrate <- function(file, y) {
  x.data <- read.GLCdata(file)
  GridRows <- grep("N", x.data$Site)
  Date <- (1e4*x.data$Year) + (100*x.data$Month) + x.data$Day
  x.means <- tapply(x.data$Var1[GridRows],
                    INDEX=list(Date[GridRows]), FUN=mean)
  Model <- lm(y ~ x.means) # Estimate theta using least-squares
  CI <- confint(Model) # 95% confidence intervals for parameters
  z <- as.data.frame(matrix(c(coef(Model), CI[1,], CI[2,]),nrow=1))
  names(z) <- c("Theta0", "Theta1", "LL95.0","UL95.0", "LL95.1", "UL95.1")
  z
}
FileList <- list.files(ImputedPrecip$SimDir, full.names=TRUE)
Theta.Impute <- sapply(FileList, FUN=Calibrate, y=y)
colnames(Theta.Impute) <- paste("Set",1:10,sep="")
Theta.Impute <- as.data.frame(t(Theta.Impute))
```

Examine calibration results

- R code suppressed in output: see .Rmd file for details



Comments on simulation results

- All estimates of $\boldsymbol{\theta}$ are biased:
  - Estimates of θ_0 (resp. θ_1) are too high (resp. low)
- Bias not explained by estimation uncertainty:
  - All 95% confidence intervals miss correct value
- Phenomenon is regression dilution bias: caused by treating regression of y on $\hat{\boldsymbol{x}}$ as regression of y on $\boldsymbol{x}$
  - Magnitude of bias depends on variance of difference $\hat{\boldsymbol{x}} - \boldsymbol{x}$
  - Imputation variance is roughly double that from interpolation ...
  - ... but in this case, interpolation is also slightly biased due to kriging transformed data

Confronting the calibration problem

- Example shows that ignoring errors / uncertainty in inputs can lead to biased / non-physical model calibration
- How to address this? Ideas from statistical literature:
  - SIMEX (SIMulation-based EXtrapolation) — add extra noise to inputs and then extrapolate back to zero noise e.g. Cook & Stefanski (1994), *J. Amer. Statist. Assoc.*, doi:10.1080/01621459.1994.10476871
  - Bayesian methods — represent all quantities explicitly (computationally challenging) e.g. Chapter 10 of Wang et al. 2018, *Bayesian Regression Modeling with Inla*, doi:10.1201/9781351165761
  - Estimating equations (EEs) — cheap and cheerful?

Estimating equations in under a minute

- Idea: calibration often solves EE $\boldsymbol{g}\left(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y}\right)=0$ (e.g. $\boldsymbol{g}(\cdot)$ is gradient vector of objective function).
- If 'target' value of $\boldsymbol{\theta}$ is $\boldsymbol{\theta}_0$ i.e. $\boldsymbol{y}=f(\boldsymbol{x},\boldsymbol{\theta}_0)$ then unbiased EE has $\mathbb{E}\left[\boldsymbol{g}\left(\boldsymbol{\theta}_0;\boldsymbol{x},\boldsymbol{y}\right)\right]=0$.
- Under fairly general conditions, unbiased EEs lead to decent estimators of $\boldsymbol{\theta}$ in large samples.
- Details: Jesus & Chandler (2011), *Interface Focus*, doi:10.1098/rsfs.2011.0057.
- Implication: bias-correct the EE for use of $\hat{\boldsymbol{x}}\neq\boldsymbol{x}$, and correction of $\hat{\boldsymbol{\theta}}$ will follow.

Wrapping up:

- Use of gridded / interpolated data can have surprising consequences that are widely underappreciated
- Recap recommendation from main presentation: consider data availability when developing models
- All welcomed:
  - Constructive comments;
  - Thoughtful questions;
  - Ideas for mutually rewarding collaboration
- Correspondence to r.chandler@ucl.ac.uk