Introducing the package `medalplot`

Andrew Zammit-Mangion and Jonathan C. Rougier

May 31, 2014

1 Introduction

In geostatistics (and also in other applications in science and engineering) we are now performing updates on Gaussian process models with thousands of components. These large-scale inferences involve computational challenges, because the updating equations cannot be solved as written, owing to the size and cost of the matrix operations. They also involve representational challenges, to account for judgements of heterogeneity concerning the underlying fields, and diverse sources of observations.

Recently, we presented a diagnostic and visualisation tool for large-scale Gaussian updates, the 'medal plot'. This provides information about both the initial and updated uncertainty around the observations, and the sharing of observations, for example across a spatial domain. It allows us to 'sanity-check' the code implementing the update, but it can also reveal unexpected features in our modelling.

This vignette presents the *R-Software* package `medalplot`, and shows, through a simple example how it can be used in a simple 1D setting. Although this example uses dense matrices which are of the order of 10^2 in size, the package is set up to remain computationally efficient with huge (10^6) sparse matrices and a large number (10^6) of medals.

2 The example: Medal plots for a 1D Gaussian process

We will use the medal plot to study uncertainty in the case of a simple 1D Gaussian process, a random function fully defined by its expectation (which we take to be zero everywhere) and its covariance function. The function we will employ is the Matérn function

$$k_1(r) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa r)K_\nu(\kappa r), \tag{1}$$

where σ^2 is the marginal variance, κ is the scaling parameter, ν is the smoothness parameter and K_ν is the modified Bessel function of the second kind. This function is implemented as follows:

```
Matern <- function(r=0:100,nu=3/2,var=1,kappa=0.1) {
  K <- var/((2^(nu-1))*gamma(nu))*
        (kappa*abs(r))^nu*besselK(kappa*abs(r),nu=nu)
  diag(K) = var
  return(K)
}
```

Specifically we will consider the process X on a 1D grid with 99 cells . Denote the centre of each grid cell as s_i and let $\boldsymbol{x} = X(\boldsymbol{s})$. Then

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \Sigma) \tag{2}$$

where Σ is a valid covariance matrix constructed by evaluating the Matérn function with the associated distance matrix of the grid passed as an argument:

```
n <- 99
s <- 1:n
V <- Matern(as.matrix(dist(s)), nu = 3/2,var=4, kappa = 0.1)
```

The precision matrix Q is then the inverse of `V`:

```
Q <- chol2inv(chol(V))
```

We now assume that the process is observed (in noise) at some points $\boldsymbol{s}_y$ which, for convenience, coincide with the centre of the grid boxes. Then $\boldsymbol{y}$ is simply a subset of $\boldsymbol{x}$ corrupted by additional Gaussian noise $\boldsymbol{e}$:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} \tag{3}$$

where $\boldsymbol{e} = \mathcal{N}(\boldsymbol{0}, \sigma_v^2 \boldsymbol{I})$ and $\boldsymbol{A}$ identifies which locations are being observed. The locations and noise parameters are set as follows:

```
sy <- c(10,14,18,22,26,30,60,80,85,90)
sigmav <- c(1.6,1.6,1.6,0.8,1.6,1.6,1.5,2.2,2.2,2.2)
Qo <- diag(1/sigmav^2) # Precision matrix
```

Note that we defined $\texttt{prec}(\boldsymbol{e}) = \boldsymbol{Q}_o$. To construct the matrix $\boldsymbol{A}$, which is an incidence matrix in this example, we simply set the appropriate elements to one as follows:

```
ny <- length(sy)
A <- matrix(0,ny,n)
for(i in 1:length(sy))   A[i,sy[i]] = 1
```

Now we have all the elements in place to run the medal plot function. In this example we are going to generate a medal for each observation location (simply by leaving the `subset` argument empty). The function call, following loading of the library, is as follows

```
library(medalplot)
M <- medalplot(Q=Q,Qo=Qo,A=A)
```

The function returns, in `M`, the radii for the three disks, and the associated colours. Note that since we have not set the `subset` argument, `M` has as many rows as there are observations (in this case 10).

```
print(M)

##     r1     r2     r3 col_outer col_inner
## 1  1.6 1.2494 0.9781 #3A3A98FF   #FFD700
## 2  1.6 1.2494 0.8033 #3A3A98FF   #FFD700
## 3  1.6 1.2494 0.6825 #3A3A98FF   #FFD700
## 4  0.8 0.7428 0.6014 #3A3A98FF   #FFD700
## 5  1.6 1.2494 0.7201 #3A3A98FF   #FFD700
## 6  1.6 1.2494 0.9375 #3A3A98FF   #FFD700
## 7  1.5 1.2000 1.1751 #3A3A98FF   #FFD700
## 8  2.0 1.4799 1.1945 #832424FF   #FFD700
## 9  2.0 1.4799 1.1277 #832424FF   #FFD700
## 10 2.0 1.4799 1.2329 #832424FF   #FFD700
```

3 Plotting the medals

For plotting the medals we will make use of the following libraries:

```
library(ggplot2)
library(plyr)
library(ellipse)
```

First, we add some more details to `M`; these include the horizontal position of the medals `M$x`, which we will set to the observation locations $\boldsymbol{s}_y$, the vertical position of the medals `M$y`, and the standard deviation of $\boldsymbol{e}$, `M$sigmav`

```
M$x <- sy
M$y <- 2.7
M$sigmav <- sigmav
```

We then construct a function which takes the augmented `M` as an argument and returns a `ggplot` object defining the medals. This function, given in Appendix A, can be used for any similar problem on a plane.

In addition, to further interpret the medals, we will also plot the posterior variance at each point. For this we carry out a simple Gaussian update:

$$\boldsymbol{Q}^* = \boldsymbol{A}^T \boldsymbol{Q}_o \boldsymbol{A} + \boldsymbol{Q} \tag{4}$$

implemented as

```
Qtot <- t(A)%*%Qo%*%A + Q
Sigma <- chol2inv(chol(Qtot))
x_std <- sqrt(diag(Sigma))
```

where $\boldsymbol{Q}^*$ is the posterior precision matrix. We now can plot the medals, together with the observations, prior variance and posterior variance using several (but standard) `ggplot` functions:

```
### Prior and posterior uncertainty
X <- data.frame(s=s,std = x_std,prior_std = sqrt(diag(V)))

### Bars for observation uncertainty
Obs_bars <- ddply(M,"s",function(df) {
  X <- data.frame(s1 = c(df$x-0.5, df$x-0.5, df$x + 0.5, df$x + 0.5),
                  y1 = c(0,df$sigmav,df$sigmav,0))
  })

### Final plot without medals
g <- ggplot() +
   geom_ribbon(data=X,aes(x=s,ymin = 0,ymax = prior_std),
               colour="black",fill="#FF525A",alpha=1) +
   geom_ribbon(data=X,aes(x=s,ymin = 0,ymax = std),
               fill="#FFEB80",colour="black",alpha=1)   +
   geom_polygon(data=Obs_bars,aes(x=s1,y=y1,group=s),
                fill="blue") +
   theme(panel.background = element_rect(fill='white',colour="black"),
```

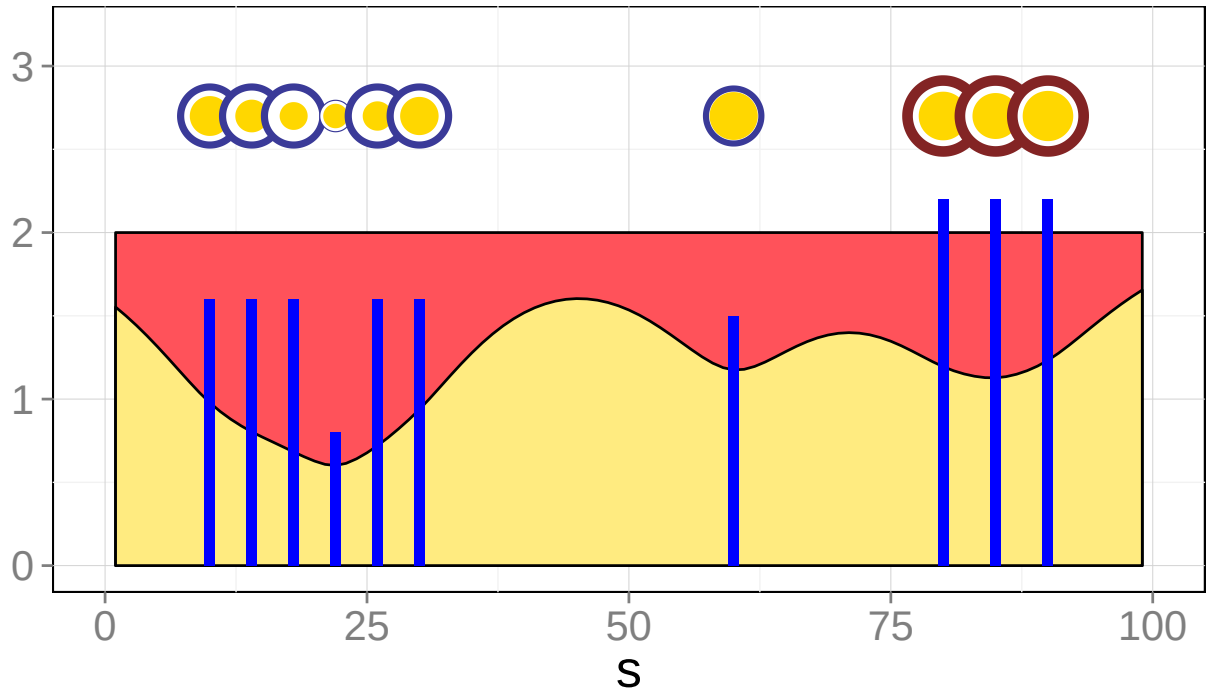

Figure 1: Medals showing the relation between the prior, posterior and observation uncertainty and the effect of vicinity of the observations on each other. The blue bars denote the observation uncertainty, the red shading the prior uncertainty and the yellow shading the posterior uncertainty. For interpretation of the medals see main text.

```
          legend.position="none",
          panel.grid.major =  element_line(colour = "light gray", size = 0.05),
          text = element_text(size=20)) +
    xlab("s") + ylab("") + xlim(0,100) + ylim(0,3.2)


### Add medals to plot
g <- g_medals(g=g,data=M,print_middle=T,alpha=1,scale=c(0.8,0.05),clamp_below=F)
print(g)
```

4 Interpretation

The medals in Fig. 1 contain a lot of information on the underlying system. First, the outer disk colours of the medals on the right-hand-side are red since the uncertainty is being constrained by the prior variance and not the observation variance (since the blue bars are overshooting the red surface). The posterior uncertainty on the lone observation in the middle is constrained by the observation error. However it does not have any white disk, since it is not borrowing any information from nearby observations. This is distinct from the observations on the left hand side, the uncertainty of which is fully constrained by that of the observations. Note how an accurate

observation in the middle of this group influences the posterior uncertainty on those in its vicinity. In particular, the gold disk is seen to increase in size the further away we get from this central observation. This causes the white disk to decrease in size, as less information is borrowed at the edges than towards the centre of this group.

Even in such a simple example, the amount of information conveyed by the medals is considerable. It is envisioned that these medals could be useful in a variety of settings especially in spatially and spatio-temporal problems.

Acknowledgments

The authors would like to thank Botond Cseke for considerable tips guaranteeing the computational efficacy of this package.

A Code for plotting medals

```
g_medals <- function(data,print_middle=T,alpha=1,scale=0.004,
                     clamp_below=F,show_rim=F,g=ggplot()) {


  .ellipseFun <- function(center=c(0,0),scale=c(1,4),npoints=100) {
    df <- data.frame(ellipse(0,scale=scale,npoints=npoints))
    df$x <- df$x + center[1]
    df$y <- df$y + center[2]
    df
  }

  size_outer <-  (data$r1)
  size_inner <- (data$r3)
  size_middle <- (data$r2)
  ## If white medal is big, show rim colour
  if(show_rim) {
    size_middle <- pmin(size_middle,0.9*size_outer)
    size_inner <- pmin(size_inner,0.9*size_middle)
  }

  ## If medals are too small scale them up
  if(clamp_below) {
    min_size <- min(diff(range(data$x)),diff(range(data$y)))/900
    ind <- which(size_outer < min_size)
    scales <- min_size/size_outer
    size_outer[ind] <- min_size
    size_inner[ind] <- (size_inner*scales)[ind]
    size_middle[ind] <- (size_middle*scales)[ind]
  }

  mobs <- nrow(M)

  for (i in 1:mobs) {
```

```
    ## Plot outer medal
    g <- g + geom_polygon(data=.ellipseFun(c(data$x[i],data$y[i]),
                                           size_outer[i]*scale,
                                           npoints=100),
                          aes(x,y),fill=data$col_outer[i],alpha=alpha)

    ## Plot middle medal
    if(print_middle)
      g <- g + geom_polygon(data=.ellipseFun(c(data$x[i],data$y[i]),
                                             size_middle[i]*scale,
                                             npoints=100),
                            aes(x,y),fill="white",alpha=alpha)

    ## Plot inner medal
    g <- g + geom_polygon(data=.ellipseFun(c(data$x[i],data$y[i]),
                                           size_inner[i]*scale,
                                           npoints=100),
                          aes(x,y),fill=data$col_inner[i])

  }
  return(g)
}
```