Smoothing of wood density profiles

Facundo Muñoz

January 30, 2014

The goal is to find points in time where density peaks. But mesaurements of wood density are noisy, therefore it is necessary to discriminate true peaks from espurious ones. Here we use a sample dataset to test different approaches. We have a human-based classification of peaks as a target.

1 Exploratory analysis

`R` packages used[1]:

```
library(gdata)
library(ggplot2)
library(plyr)
library(splines)
library(mgcv)
library(INLA)
library(ppc)
```

Load dataset. I create an index variable x for the temporal dimension.

```
prof <- read.xls(file.path("..", "data", "Example.xls"))[, 1:2]
prof <- transform(prof, x = 1:nrow(prof))
```

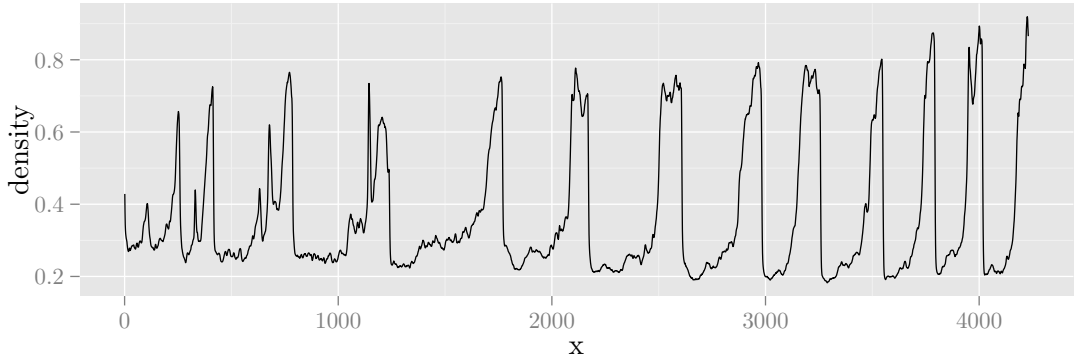
Summary of the dataset:

```
summary(prof)
```

```
##       year           density            x
##  Min.   :   0.0   Min.   :0.182   Min.   :   1
##  1st Qu.:   0.0   1st Qu.:0.241   1st Qu.:1058
```

[1] `ppc` package is not cukrrently available on CRAN. Download from `http://statweb.stanford.edu/~tibs/PPC/Rdist/index.html`, and install locally from source.

```
##  Median :   0.0   Median :0.283   Median :2116
##  Mean   :   6.4   Mean   :0.367   Mean   :2116
##  3rd Qu.:   1.0   3rd Qu.:0.413   3rd Qu.:3173
##  Max.   :2012.0   Max.   :0.919   Max.   :4230

qplot(x, density, data = prof, geom = "line")
```



2 Finding peaks in raw data

A first and direct approach is to work with the raw data. R-package `ppc` provides the function `ppc.peaks()` to find local maxima in numeric vectors.

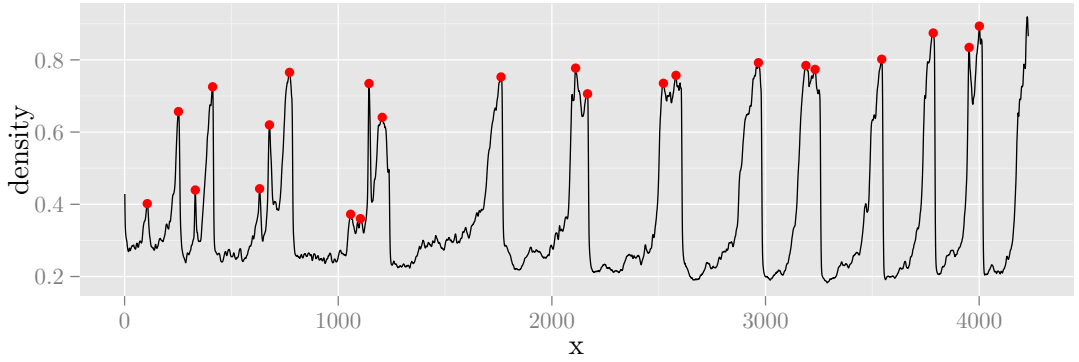
A *local maximum* is defined as a value which is greater than all other elements within a window centered at that element. One can configure the window width through the parameter `span` which represents the proportion of total points in the window.

Trying several values of `span` we can get close to the desired set of peaks. However, we also find some peaks in lower ranges. It is safe to assume that a peak must be above a given threshold, such as 0.35. With this restriction, we can find exactly the desired set of peaks with span = 0.015. So we build a function to include the `threshold` restriction.

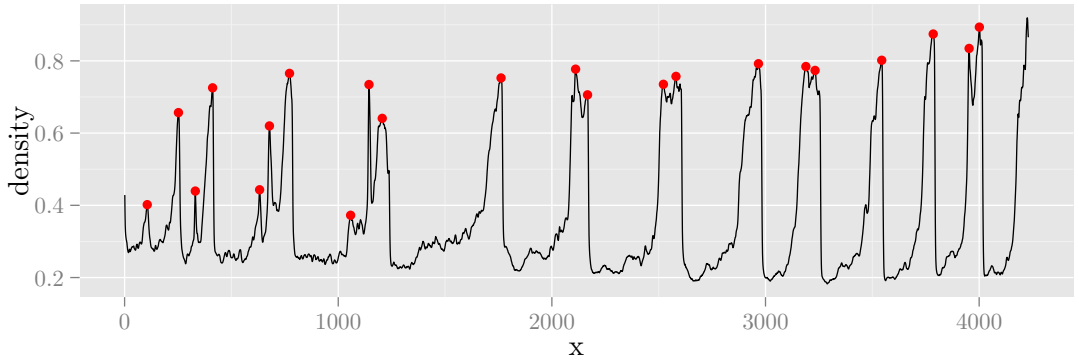
Observed peaks over a given threshold and with local maximas in a window of absolute width span.absμm.

```
density.peaks <- function(x, span.abs = 60, threshold = 0.35) {
    pks <- ppc.peaks(x, span = span.abs/length(x))
    trs <- x > threshold
    return(pks & trs)
}

span.l <- list(too_much = 40, exact = 60, too_few = 90)
pks <- llply(span.l, function(x) density.peaks(prof$density,
    span = x))
plot.pks <- function(z, dat = prof) {
```

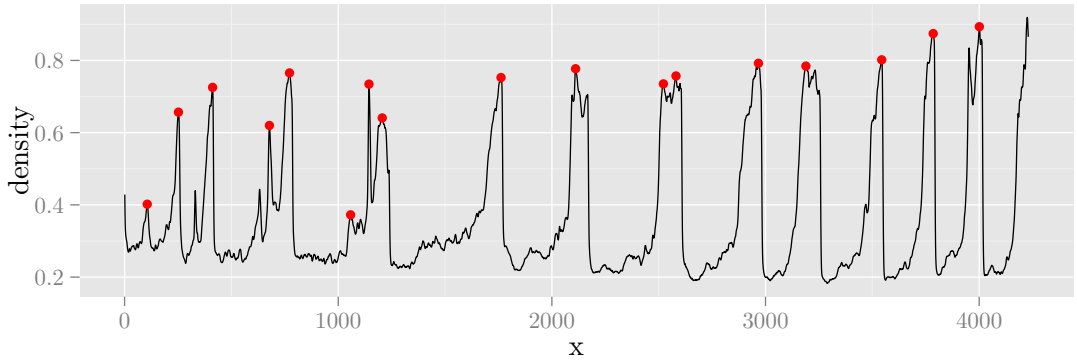
```
    p <- qplot(x, density, data = dat, geom = "line") + geom_point(aes(x,
        density), data = dat[z, ], col = "red")
}
p.l <- llply(pks, plot.pks)
print(p.l)

## $too_much
```




```
##
## $exact
```




```
##
## $too_few
```



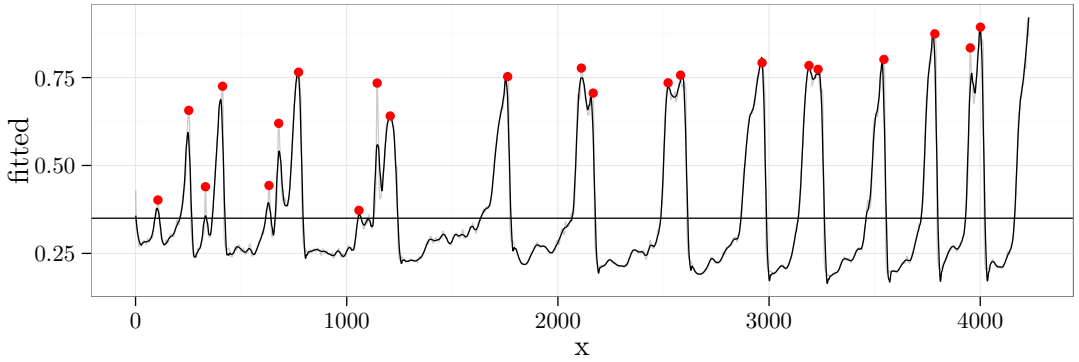
For this example dataset, `span` = 0.015 finds the required peaks, corresponding with an absolute span of 63.45. However, using span.abs = $60\mu m$ works equally good.

3 Finding peaks in smoothed data

Smoothing the raw data to get rid of most noise, thus filtering-out spurious peaks.

There are several smoothing methods, nearly all depending on some *smoothing parameter*. One simple approach is using a Local Polynomial Regression Fitting method.

```
res_loess <- loess(density ~ x, data = prof, span = 0.015)
summary(res_loess)

## Call:
## loess(formula = density ~ x, data = prof, span = 0.015)
##
## Number of Observations: 4230
## Equivalent Number of Parameters: 195.5
## Residual Standard Error: 0.0282
## Trace of smoother matrix: 213.9
##
## Control settings:
##   normalize:  TRUE
##   span     :  0.015
##   degree   :  2
##   family   :  gaussian
##   surface  :  interpolate   cell = 0.2

dat <- with(res_loess, data.frame(x, y, fitted))

qplot(x, fitted, data = dat, geom = "line") + geom_line(alpha = 0.2,
    aes(y = y)) + theme_bw() + geom_point(aes(x, y), data = dat[pks$exact,
    ], col = "red") + geom_hline(aes(yintercept = 0.35))
```



Smoothed peaks correspond nicely with desired peaks, but:

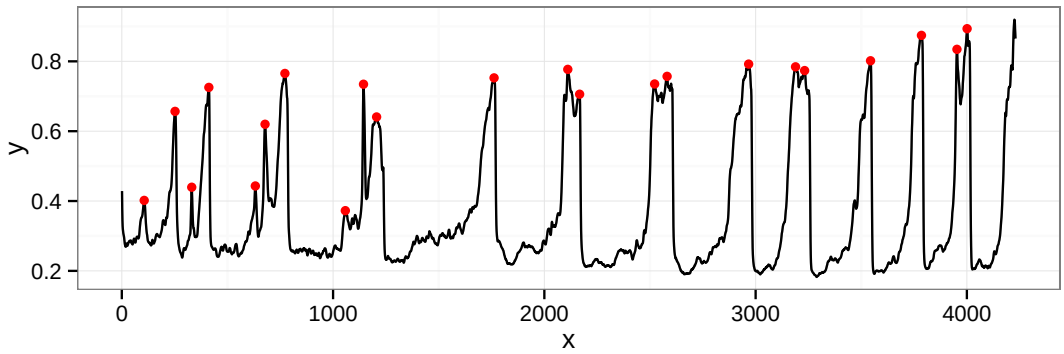
1. It still depends on a `span` parameter that determines the degree of smoothing. Moreover, the optimal value matches the one used with the raw data. So we keep all the drawbacks.

2. Furthermore, we still need to find peaks in the smoothed curve. So we have yet another `span` parameter to determine (although it should be much more robust).

3. The peaks in the smoothed curve do not match exactly the position of peaks in the raw data. So a matching algorithm is needed, unless smoothed peaks are more believable.

3.1 Other smoothing approaches

```
isp1 <- interpSpline(1:nrow(prof), prof$density)

qplot(x, y, data = as.data.frame(predict(isp1, 1:nrow(prof))),
    geom = "line") + geom_line(alpha = 0.2, aes(y = density),
    data = prof) + theme_bw() + geom_point(aes(x, density), data = prof[pks$exact,
    ], col = "red")
```



```
str(splineKnots(isp1))

##  num [1:4230] 1 2 3 4 5 6 7 8 9 10 ...

splineOrder(isp1)

## [1] 4
```

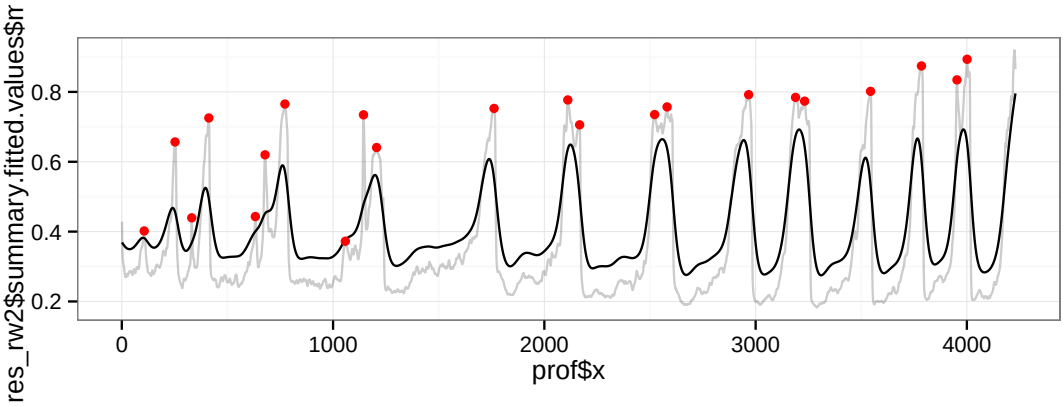
This can only perform exact interpolation?

4 Finding peaks in modelled data

Building a statistical model for the data has the appealing feature of estimating the *true* but unobserved density. Thus finding peaks in an estimated curve would give more reliable estimates of *when* peaks occured and their *true value*.

```
#### RW2 modelling with INLA ####

res_rw2 <- inla(density ~ f(x, model = "rw2"), data = transform(prof,
    xcopy = x), family = "beta", control.family = list(hyper = list(theta = list(initial
    fixed = TRUE))), control.predictor = list(compute = TRUE),
    control.compute = list(dic = TRUE))
summary(res_rw2)

##
## Call:
## c("inla(formula = density ~ f(x, model = \"rw2\"), family = \"beta\", ",  "    data =
##
## Time used:
##  Pre-processing    Running inla Post-processing
##          0.2485         15.1227          0.6138
##           Total
##         15.9850
##
## Fixed effects:
##                mean     sd 0.025quant 0.5quant 0.975quant
## (Intercept) -0.3912 0.0156    -0.4218  -0.3912    -0.3607
##                mode kld
## (Intercept) -0.3912   0
##
## Random effects:
## Name   Model
##  x   RW2 model
##
```

```
## Model hyperparameters:
##                  mean      sd        0.025quant 0.5quant
## Precision for x 104995.59  20298.66  70497.84  103183.45
##                  0.975quant mode
## Precision for x 150012.58   99721.22
##
## Expected number of effective parameters(std dev): 84.01(3.193)
## Number of equivalent replicates : 50.35
##
## Deviance Information Criterion: -2434.50
## Effective number of parameters: 83.82
##
## Marginal Likelihood:  1087.24
## Posterior marginals for linear predictor and fitted values computed

qplot(prof$x, res_rw2$summary.fitted.values$mean, data = prof,
    geom = "line") + geom_line(alpha = 0.2, aes(y = density)) +
    theme_bw() + geom_point(aes(x, density), data = prof[pks$exact,
    ], col = "red")
```



4.1 Other modelling approaches

```
res_gam <- gam(density ~ s(x, k = 500), data = prof)

summary(res_gam)

##
## Family: gaussian
## Link function: identity
##
## Formula:
```

```
## density ~ s(x, k = 500)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.366783   0.000188    1952   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##      edf Ref.df    F p-value
## s(x) 493    499 1899  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.996   Deviance explained = 99.6%
## GCV score = 0.00016913  Scale est. = 0.00014939  n = 4230

# plot(res_gam, pages = 1, residuals = TRUE)
# gam.check(res_gam)
```

5 Conclusions

Working with the raw data seems to be good enough. It has the drawback of needing to tune a parameter to yield the interesting peaks, no more, no less. But similar drawbacks remain in the other approaches as well.

If needed, the algorithm could be further refined, by using a third parameter (besides `span` and `threshold`), say `intensity`, giving a threshold to discriminate small peaks from large peaks. In this case, it might be interesting to use the first and second differences to find peaks, measure their intensity and filter by span. Something like `which(diff(sign(diff(tt)))==-2)+1`.