4.4.3 Other smoothing methods

Regression splines, as described above, are attractive because they fit easily within a linear model framework. We can fit them by specifying an appropriate X-matrix, in a call to `lm()`. The discussion will now move to consider methods that do not fit within the `lm()` linear model framework. In a technical sense, the smoothing methods that will now be described are no longer linear in the parameters. We consider them here because they are natural and attractive extensions of the linear model framework.

Consider first locally weighted regression methods, implementated in different ways and with different functionality in the R functions `lowess()` and `loess()`. These rely on repeated linear fits to successive subsets of the data. Figure 1.6, where the function `lowess()` was used to fit a curve to the `fruitohms` data, is one of a number of examples of its use that appear elsewhere in this text.

By contrast with `lowess()`, which can only fit curves, `loess()` can fit surfaces as well as curves. It takes up to four numerical predictors. The default for `loess()` is a non-resistant smooth; for a resistant smooth, specify `family=symmetric`. Neither function has a mechanism for calculating pointwise confidence bounds. See the relevant help pages for details of options that are available to control the fitting process.

The following description is directly relevant to `lowess()`. Calculations for `loess()` follow the same general pattern. The method is *local*, in the sense that the fitted value $m(x)$ at a point x uses only the data within a specified neighborhood of x. Points nearest to x have highest weight. Those farther away have a reduced weight or no weight. Fits are resistant to outliers, with observations that generate large residuals given low weights. The fitting process uses an iterative method, with the residual at the previous iteration determining the weight at the current iteration.

Kernel smoothing methods further widen the range of possibilities. For example, see the documentation for the function `KernSmooth::locpoly()`.

On lowess smoothing, see Cleveland (1981). Venables and Ripley (2002) has a useful brief discussion of smoothing methods. Fan and Gijbels (1996) discusses kernel smoothing, splines and lowess. See also Hall (2001).

4.4.4 *Quantile regression

Working with a linear model framework, the discussion has until now focused almost exclusively on models for the expected value or mean of the response variable as a function of (or conditioned on) one or more covariates. Models for the quantiles of the response (e.g., 25% or lower quartile, median, 75% or upper quartile) are a further important and useful extension of the linear model framework. Here, we take a look at the new and exciting possibilities offered by the `quantreg` package (Koenker, 2015).

2012 World Bank data on fertility and life expectancy

There is a close correlation, across countries, between fertility (number of children born per female) and life expectancy. Figure 4.10 is designed to explore the connection, while not providing insight on how it may arise.

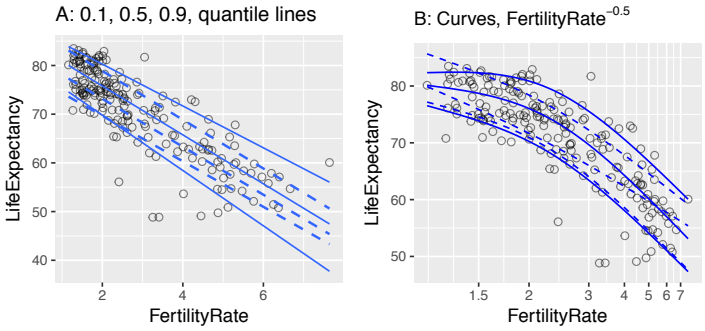

Figure 4.10: Panel A shows 10%, 50Panel B shows shows 2 d.f. (+ 1 d.f. for intercept) fitted natural spline curves, with a FertilityRate$^{-0.5}$ transformation on the x-axis. Solid lines or curves are for unweighted fits, while dashed lines or curves are from weighting by population.

Figure 4.10A shows a median lines, together with 10% and 90% quantile lines. For the solid curves, all countries were given equal weight, while the dashed curves were from use of populations as weights.

The footnote has code, which assumes a live internet connection, that may be used to input the data that is used.[7]

Code for fitting the quantiles that are shown in Figure 4.10A is:

```
nsq159.rq <- quantreg::rq(LifeExpectancy ~ FertilityRate, data=wdi,
                          tau=c(0.1,0.5,0.9))
pred159 <- as.data.frame(cbind(wdi[,"FertilityRate"],
                               predict(nsq159.rq)))
```

Figure 4.10A did not use this code directly, but instead used `ggplot2::geom_quantile()` to combine fitting the lines with creation of the graphics object used in creating the plot. Observe how, in Figure 4.10A, a relatively small number of high mortality points have a large influence on the slope of the fitted line. Hence the use, for Panel B, of the power transformation of `FertilityRate` that was suggested by the use of the function `car::powerTransform(FertilityRate)`.[8]

For Panel B, the following code can be used to calculate the solid quantile curves:

```
## Panel B: Power transformed x
wdi[, "rt2invFert"] <- -wdi[["FertilityRate"]]^-0.5
nsq10_50_90.rq <- quantreg::rq(LifeExpectancy ~ splines::ns(rt2invFert, 2),
                               data=wdi, tau=c(0.1,0.5,0.9))
pred159 <- as.data.frame(cbind(wdi[,"rt2invFert"],predict(nsq10_50_90.rq)))
```

For the dashed curves, include the argument `weight=population`.

[7] `if(!is.element("WDI", installed.packages()[,1]) )install.packages("WDI")`
`inds <- c('SP.DYN.TFRT.IN','SP.DYN.LE00.IN', 'SP.POP.TOTL')`
`indnams <- c("FertilityRate", "LifeExpectancy", "population")`
`wdi2012 <- WDI::WDI(country="all", indicator=inds, start=2012, end=2012, extra=TRUE)`
`colnum <- match(inds, names(wdi2012))`
`names(wdi2012)[colnum] <- indnams`
`wdi2012 <- na.omit(droplevels(subset(wdi2012, !region %in% "Aggregates")))`
`wdi <- wdi2012[order(wdi2012[, 'FertilityRate']), ]`
`save(wdi, file="wdi.RData")`

[8] `## Code used to suggest a transformation to a more symmetric distribution`
`summary(car::powerTransform(wdi[["FertilityRate"]]))`

An alternative that avoids explicit use of a regression spline basis

The function `quantreg::rqss()`, with the function `qss()` used to specify the fitting of a smooth, offers an alternative to the explicit use of a regression spline basis. At the time of writing, the parameter `lambda`, with larger values of `lambda` leading to a smoother curve, takes a specified fixed value. The function `ggplot2::geom_quantile()` can be used to handle the fitting of curves, and create a `ggplot2` graph object. Try, as an example:

```
wdi[, "rt2invFert"] <- -wdi[,'FertilityRate']^-0.5
gph1 <- ggplot(wdi, aes(rt2invFert, LifeExpectancy))+geom_point()
gph1+geom_quantile(method="rqss", quantiles=c(.1,.5,.9), lambda=0.1) +
   scale_x_continuous(breaks = -(2:7)^(-0.5), labels = paste(2:7)) +
  xlab("Fertility Rate")
```

Monotone and/or convexity constraints can be imposed.

4.5 *Generalized additive models (GAMs) — Roughness penalty methods

Rather than restrict the number of basis curves for a smooth, a better general strategy is to choose a relatively large degrees of freedom basis, with a penalty applied to prevent the choice of fitted curves or surfaces that are unduly wiggly. Under the assumption that observations are independent with homogeneous variance, the penalty can be chosen automatically.

Contrast the penalization approach used for GAM smooths with control of complexity by restriction to a small (low degree of freedom) basis set. Use of too few basis curves can lead to a curve that fails to capture all the nuances of the data while choosing too many is likely to result in excessive bumpiness, i.e., the details of the curve capture noise. Roughness penalties are a mechanism for allowing, in principle at least, a choice of curve or surface that is constrained only by the effect of penalties that limit wiggliness.

For complete generality in the choice of curve, cubic smoothing spline methods assign a knot to each predictor value, while applying a roughness penalty that constrains the fitted spline to pass smoothly through the cloud of observations. Without such a constraint, the spline would interpolate the observations, usually rendering a rough curve. There are then as many basis curves as there are distinct observations. Thin plate splines circumvent the overt choice of knots. Complete generality in the choice of curve requires, again, as many basis functions as there are distinct observations.

To reduce the time and memory requirements, a reduced set of basis curves may be used, albeit a large enough set that the choice of curve is not unduly constrained. The "cubic regression splines" that are provided in the `mgcv` package place knots at equally spaced quantiles of the data, as does the function `smooth.spline()` (`stats` package). Penalized splines, as implemented by `pspline()` in the `survival` package, are another approach that places knots at values that are evenly spaced through the data.

To keep computational demands within reasonable bounds and allow use of a reduced set of basis functions, thin plate regression splines (t.p.r.s) are used in place of true thin plate splines. Thin plate regression splines bases provide workable low rank (rank = number of basis curves or functions) approximations.

Note also the `gam` package (not to be confused with the `gam()` function in `mgcv`), which is for some purposes an alternative to `mgcv`. For brief comments on the comparison between