Stat 343, Thu 15-Oct-2020 -- Thu 15-Oct-2020

Probability and Statistics

Fall 2020

Thursday, October 15th 2020

Wk 7, Th

Topic:: Kernel density estimation

Read:: FASt 3.5

Families of r.v.s

- have seen several varieties, both of discrete type and also continuous
- serve as models for what is seen in the real world (real data)
- some arise
  - out of clear assumptions
    examples include: binomial/multinomial, hypergeometric, poisson
  - seemingly from mathematical "play" (kernel function to density)
    Example: Pruim says of Weibull: "Often Weibull distributions are used simply because they provide a good fit to data."

Our modern life has called into question the practice of "putting things in a (well-understood) box"

Rather than force a dataset into a certain model, how about letting the data decide?

3.5 **Kernel density estimation** data driven

- histograms and smoothing
- candidate (kernel) functions

$\hat{f}$

Suppose K is a kernel function with $\int_{-\infty}^{\infty} f(x)\,dx = a$, and $x_1, x_2, \ldots, x_n$ are numbers. Let us define

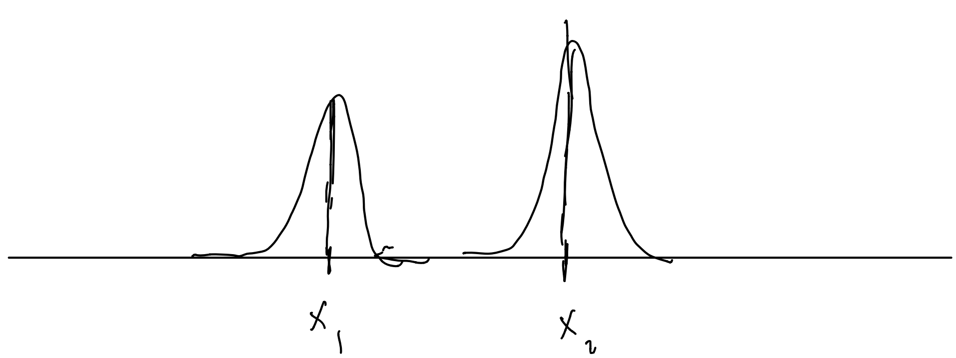
$$\hat{f}(x) \;=\; \frac{1}{na}\sum_{i=1}^{n} K(x - x_i). \tag{1}$$

Note that

$$\int_{-\infty}^{\infty} \hat{f}(x)\,dx \;=\; \frac{1}{na}\sum_{i=1}^{n}\int_{-\infty}^{\infty} K(x - x_i)\,dx \;=\; \frac{1}{na}\sum_{i=1}^{n} a \;=\; 1,$$

which means that $\hat{f}$ is a density function.

K is kernel fn.



Base kernel fn. $K(x)$

- centered at 0

  $\longrightarrow$ $K(x - x_i)$ is shifted so as to be centered at x_i

- many candidates

Kernel density estimation

T. Scofield (heavily influenced by FAStR2, Section 3.5)

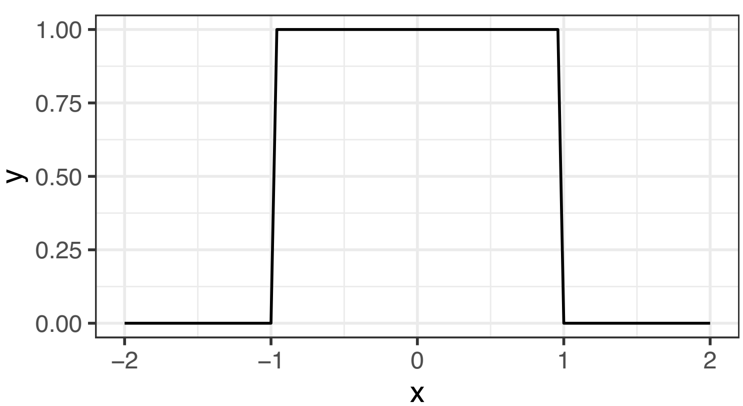
October 15, 2020

Special kernel function definitions

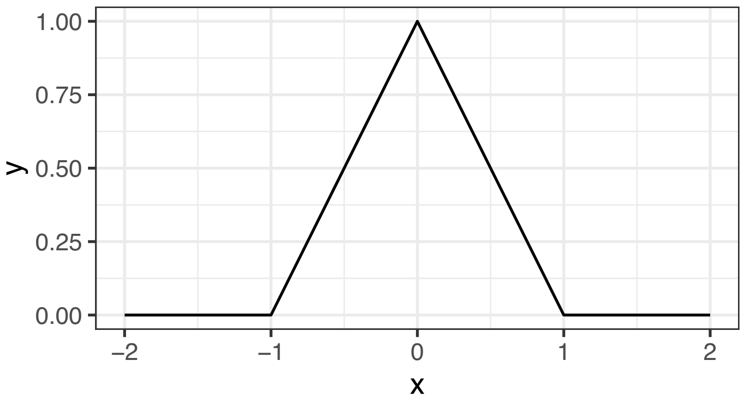
```
kernelRect <- function(x) return( as.numeric(-1 < x & x < 1) )
kernelTri <- function(x) return( (1-abs(x)) * as.numeric(abs(x)<1) )
kernelQuad <- function(x) return( (1-x^2) * as.numeric(abs(x)<1) )
kernelGauss <- dnorm    # the standard normal pdf

kde <- function(data, kernel=kernelRect, ...) {
  n <- length(data)
  scalingConstant <-
    integrate(function(x){kernel(x, ...)}, -Inf, Inf) %>% value()
  function(x) {
    mat <- outer(x, data, FUN=function(x,data) {kernel(x-data, ...)})
    val <- rowSums(mat)
    val <- val / (n * scalingConstant)
    return(val)
  }
}
```

Several potential base kernel functions, which I have given lengthy names when defining them above, can be plotted as in the textbook:

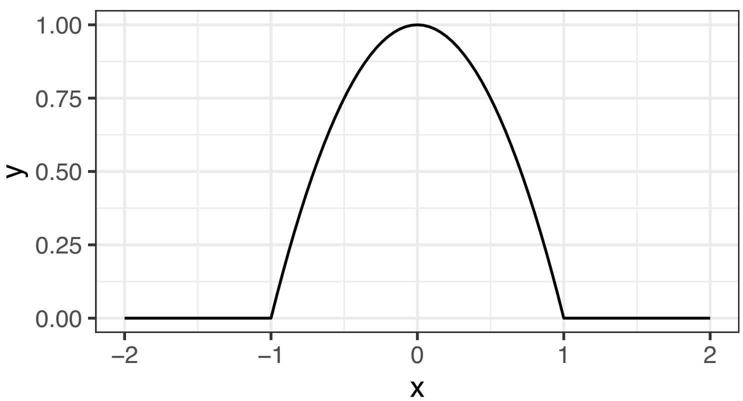
```
gf_fun( kernelRect(t) ~ t, xlim=c(-2,2) )
```



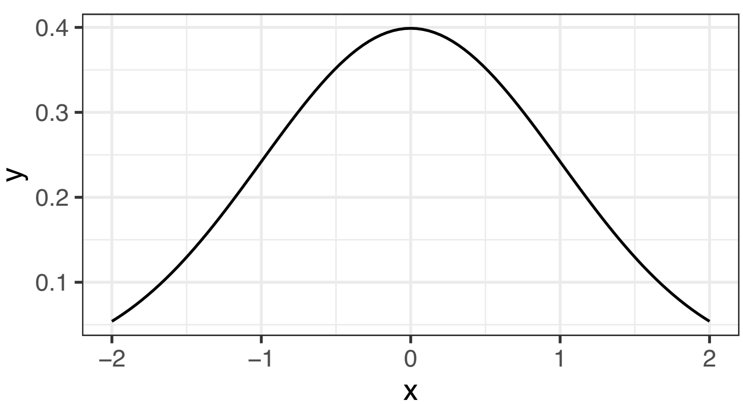

```
gf_fun( kernelTri(t) ~ t, xlim=c(-2,2) )
```




```
gf_fun( kernelQuad(t) ~ t, xlim=c(-2,2) )
```

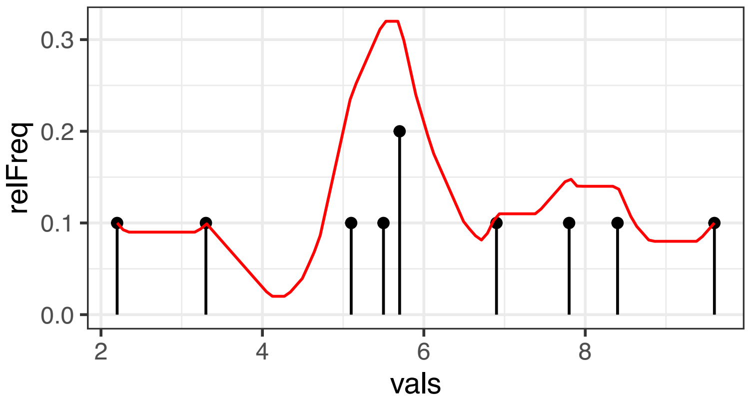



```
gf_fun( kernelGauss(t) ~ t, xlim=c(-2,2) )
```



Selecting the triangle kernel, I use the vector **x** as defined below as data and display the resulting pdf.

```
x <- c(2.2, 3.3, 5.1, 5.5, 5.7, 5.7, 6.9, 7.8, 8.4, 9.6)
g <- kde(x, kernel=kernelTri)
xTable <- tally(~x)
relFreq <- as.numeric(xTable) / 10
vals <- as.numeric(rownames(xTable))

gf_point(relFreq ~ vals) %>% gf_segment(0+relFreq ~ vals+vals) %>%
  gf_fun( g(t) ~ t, xlim=c(0,12), color="red" )
```



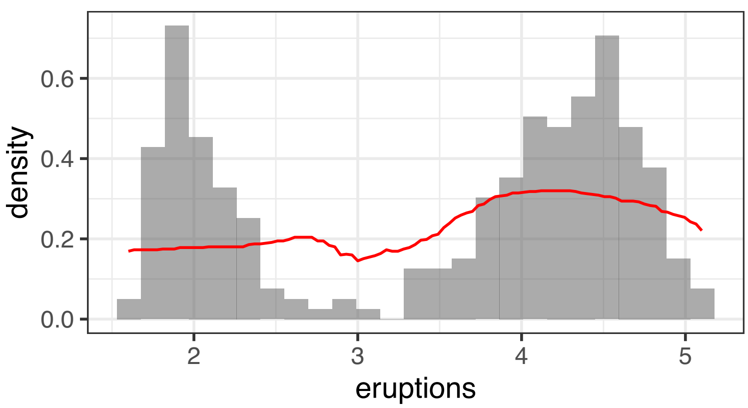
It really is a pdf, as integration shows:

```
integrate( g, 0, 12 )
```

```
## 1.000001 with absolute error < 7.9e-05
```

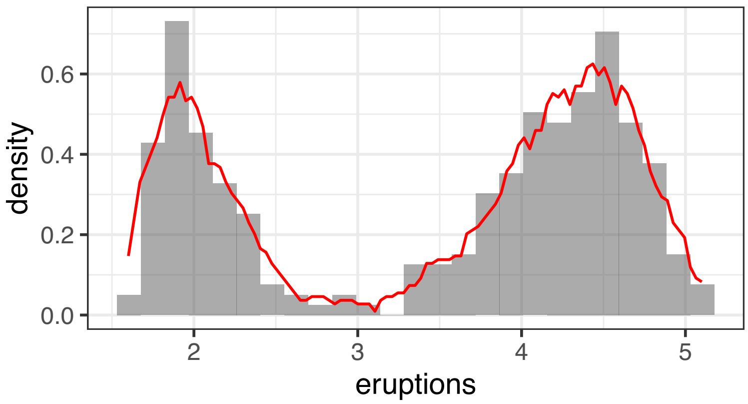
Turning to a real dataset (eruption lengths at Old Faithful geyser):

```
gf_dhistogram(~ eruptions, data=faithful) %>%
  gf_fun(kde(faithful$eruptions, kernel=kernelRect)(t) ~ t, xlim=c(0,6),
         color="red")
```



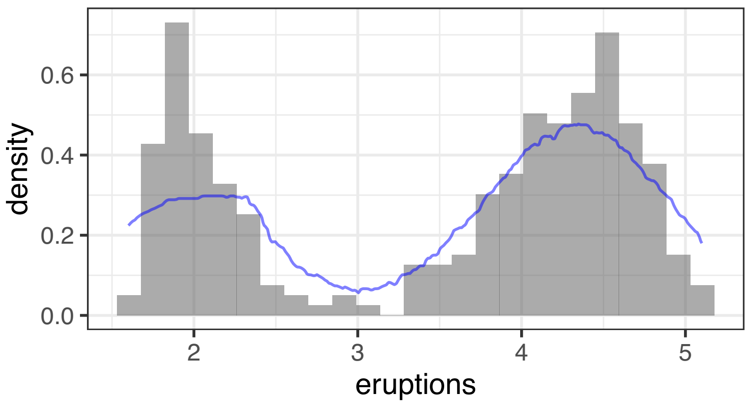
This kernel density estimator is pretty unresponsive to the changes in this geyser data. It might be better with another base kernel function (besides the `kernelRect()` one), but it will also get better if we decrease the bandwidth. Using the functions we defined above is possible, making sure to enact a horizontal shrink on the `kernelRect()` function.

```
g <- kde(faithful$eruptions, kernel=function(x) {kernelRect(5*x)})
gf_dhistogram(~ eruptions, data=faithful) %>%
  gf_fun(g(t) ~ t, xlim=c(0,6), color="red")
```

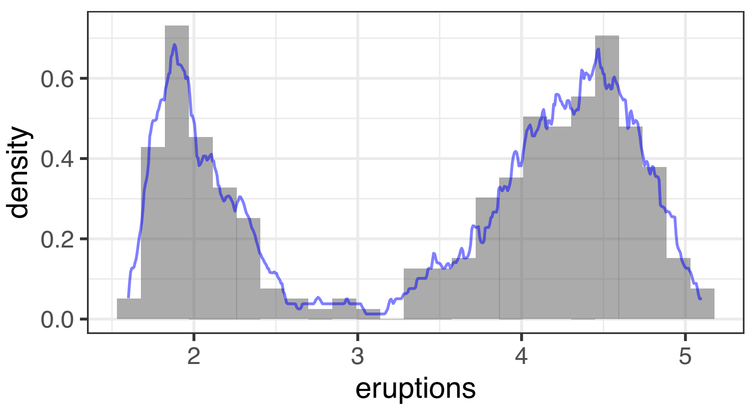


Or, we can abandon those functions for ones more tailored to the purpose (having an `adjust` switch for changing bandwidth), functions like `density()` and `gf_dens()`.

```
gf_dhistogram(~eruptions, data=faithful) %>%
  gf_dens(~ eruptions, data=faithful, kernel="rectangular", color="blue")
```

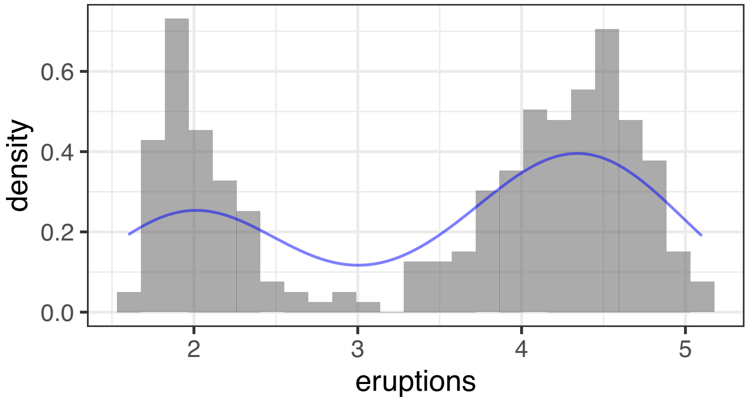



```
gf_dhistogram(~eruptions, data=faithful) %>%
  gf_dens(~ eruptions, data=faithful, adjust=0.25,
          kernel="rectangular", color="blue")
```



Other types of base kernel functions are available. Type `help(density)` to see a list. Below I use a Gaussian base kernel function with larger bandwidth, and a triangle one with smaller bandwidth.

```
gf_dhistogram(~eruptions, data=faithful) %>%
  gf_dens(~ eruptions, data=faithful, adjust=1.5,
          kernel="gaussian", color="blue")
```




```r
gf_dhistogram(~eruptions, data=faithful) %>%
  gf_dens(~ eruptions, data=faithful, adjust=0.25,
          kernel="triangular", color="blue")
```