Bandwidth selection for multivariate kde tuned for anomaly detection

Rob J Hyndman

```
source("before-each-chapter.R")
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become er
-- Attaching packages ------------------------------------- weird 0.0.0.9000 --

v ks 1.14.1

-- Conflicts ------------------------------------------------ weird_conflicts --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()

[conflicted] Will prefer dplyr::select over any other package.
[conflicted] Will prefer dplyr::filter over any other package.
```

Robust covariance estimation

The sample covariance matrix is a useful measure of the spread of a multivariate distribution, given by

$$\boldsymbol{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})', \tag{1}$$

However, it is sensitive to outliers, and so is not suitable for our purposes. There have been many robust estimators of covariance proposed in the literature, but we will discuss only one, relatively simple, estimator known as the "orthogonalized Gnanadesikan/Kettenring" (OGK) estimator (Gnanadesikan and Kettenring 1972; Maronna and Zamar 2002).

Suppose we have two random variables X and Y. Then the variance of their sum and difference is given by

$$\begin{aligned}\text{Var}(X+Y) &= \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X,Y)\\ \text{Var}(X-Y) &= \text{Var}(X) + \text{Var}(Y) - 2\text{Cov}(X,Y).\end{aligned}$$

The difference between these two expressions is

$$\text{Var}(X+Y) - \text{Var}(X-Y) = 4\text{Cov}(X,Y),$$

so that the covariance can be expressed as

$$\text{Cov}(X,Y) = \frac{1}{4}\left[\text{Var}(X+Y) - \text{Var}(X-Y)\right].$$

Now we can use the robust IQR estimate of variance, to estimate the two variances on the right hand side, giving

$$\hat{s}(X,Y) = \frac{1}{4}\left[s^2_{\text{IQR}}(X+Y) - s^2_{\text{IQR}}(X-Y)\right].$$

We can repeat this for each pair of variables, to obtain a robust estimate of the covariance matrix, $\boldsymbol{S}^*$. The diagonals can be obtained using the same robust measure of variance. This is known as the Gnanadesikan-Kettenring estimator. The resulting matrix is symmetric, but not necessarily positive definite, which is a requirement of a covariance matrix. So some additional iterative steps are applied to "orthogonalize" it.

1. Compute the eigenvector decomposition of $\boldsymbol{S}^*$, so that $\boldsymbol{S}^* = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$.
2. Project the data onto the basis eigenvectors
3. Estimate the variances (robustly) in the coordinate directions.
4. Then the robust covariance matrix is given by

$$\boldsymbol{S}_{\text{OGK}} = \boldsymbol{U}\boldsymbol{\Lambda}^*\boldsymbol{U}^{-1}, \tag{2}$$

   where $\boldsymbol{\Lambda}^*$ is a diagonal matrix with the robust variances on the diagonal.

These orthogonalization steps are usually repeated one more time.

This procedure is implemented in the `covOGK` function in the `robustbase` package (Maechler et al. 2023).

Multivariate kernel density estimation

Suppose our observations are d-dimensional vectors, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$. Then the multivariate kernel density estimate is given by (Scott 2015)

$$\hat{f}(\boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n} K_H(\boldsymbol{y} - \boldsymbol{y}_i), \tag{3}$$

where K_H is a multivariate probability density with covariance matrix $\boldsymbol{H}$. For example, a multivariate Gaussian kernel is given by

$$K_H(\boldsymbol{u}) = (2\pi)^{-d/2}|\boldsymbol{H}|^{-1/2}\exp\{-\tfrac{1}{2}\boldsymbol{u}'\boldsymbol{H}^{-1}\boldsymbol{u}\}.$$

Bandwidth matrix selection

The optimal bandwidth matrix (minimizing the mean integrated squared error between the true density and its estimate) is of the order $n^{-2/(d+4)}$. If such a bandwidth matrix is used, then the estimator converges at rate $n^{-4/(d+4)}$, implying that kernel density estimation becomes increasingly difficult as the dimension d increases. This is to be expected given the curse of dimensionality, as the number of observations required to obtain a good estimate increases exponentially with the dimension. In practice, we rarely attempt to estimate a density in more than $d = 3$ dimensions.

If the underlying density is Normal with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$, then the optimal bandwidth matrix is given by

$$\boldsymbol{H} = \left(\frac{4}{d+2}\right)^{2/(d+4)} n^{-2/(d+4)}\boldsymbol{\Sigma}. \tag{4}$$

Replacing $\boldsymbol{\Sigma}$ by the robust covariance matrix $\boldsymbol{S}_{\text{OGK}}$ (Equation 2), we obtain a robust normal reference rule.

This is optimal for densities with the same curvature as a normal density, and will probably still give a good estimate for other densities as it is consistent for any smooth f.

This provides a good estimate of the overall density (as it minimizes the MISE), but it is not particularly good in the tails of the distribution. Instead we usually need a larger density matrix.

We propose choosing

$$\boldsymbol{H} = k_n \boldsymbol{S}_{\text{OGK}},$$

where k_n is determined by controlling the false anomaly rate for a multivariate Gaussian distribution.

```
library(weird)
library(mvtnorm)
d <- 2
nreps <- 500
sim_kn <- tidyr::expand_grid(
  n = c(100, 200, 500, 1000, 2000, 5000),
  kn = c(1,2,3,5,10),
  nfalse = NA
)
sim_tda <- tibble(
  n = unique(sim_kn$n),
  nfalse = NA
)
no_false <- any_false <- rep(NA, nreps)
for(i in seq(NROW(sim_kn))) {
  cat(paste("n =", sim_kn$n[i], "kn =", sim_kn$kn[i],"\n"))
  for(j in seq(nreps)) {
    y <- rmvnorm(sim_kn$n[i], sigma = diag(d))
    H <- kde_bandwidth(y, multiplier = sim_kn$kn[i])
    scores <- weird:::calc_kde_scores(y, H=H)
    lookout_prob <- lookout(density_scores = scores$scores, loo_scores = scores$loo_scores)
    # Number of false anomalies
    no_false[j] <- sum(lookout_prob < 0.05)
  }
```

```
  # Average number of false anomalies
  sim_kn$nfalse[i] <- mean(no_false)
}
```

```
#> n = 100 kn = 1
#> n = 100 kn = 2
#> n = 100 kn = 3
#> n = 100 kn = 5
#> n = 100 kn = 10
#> n = 200 kn = 1
#> n = 200 kn = 2
#> n = 200 kn = 3
#> n = 200 kn = 5
#> n = 200 kn = 10
```

```
#> Warning in fpot.norm(x = x, threshold = threshold, model = model, start =
#> start, : optimization may not have succeeded
```

```
#> Warning in fpot.norm(x = x, threshold = threshold, model = model, start =
#> start, : optimization may not have succeeded
```

```
#> Warning in fpot.norm(x = x, threshold = threshold, model = model, start =
#> start, : optimization may not have succeeded
```

```
#> n = 500 kn = 1
#> n = 500 kn = 2
#> n = 500 kn = 3
#> n = 500 kn = 5
```

```
#> Warning in fpot.norm(x = x, threshold = threshold, model = model, start =
#> start, : optimization may not have succeeded
```

```
#> Warning in fpot.norm(x = x, threshold = threshold, model = model, start =
#> start, : optimization may not have succeeded
```

```
#> Warning in fpot.norm(x = x, threshold = threshold, model = model, start =
#> start, : optimization may not have succeeded
```

```
#> n = 500 kn = 10
```

```
#> Warning in fpot.norm(x = x, threshold = threshold, model = model, start =
#> start, : optimization may not have succeeded
```

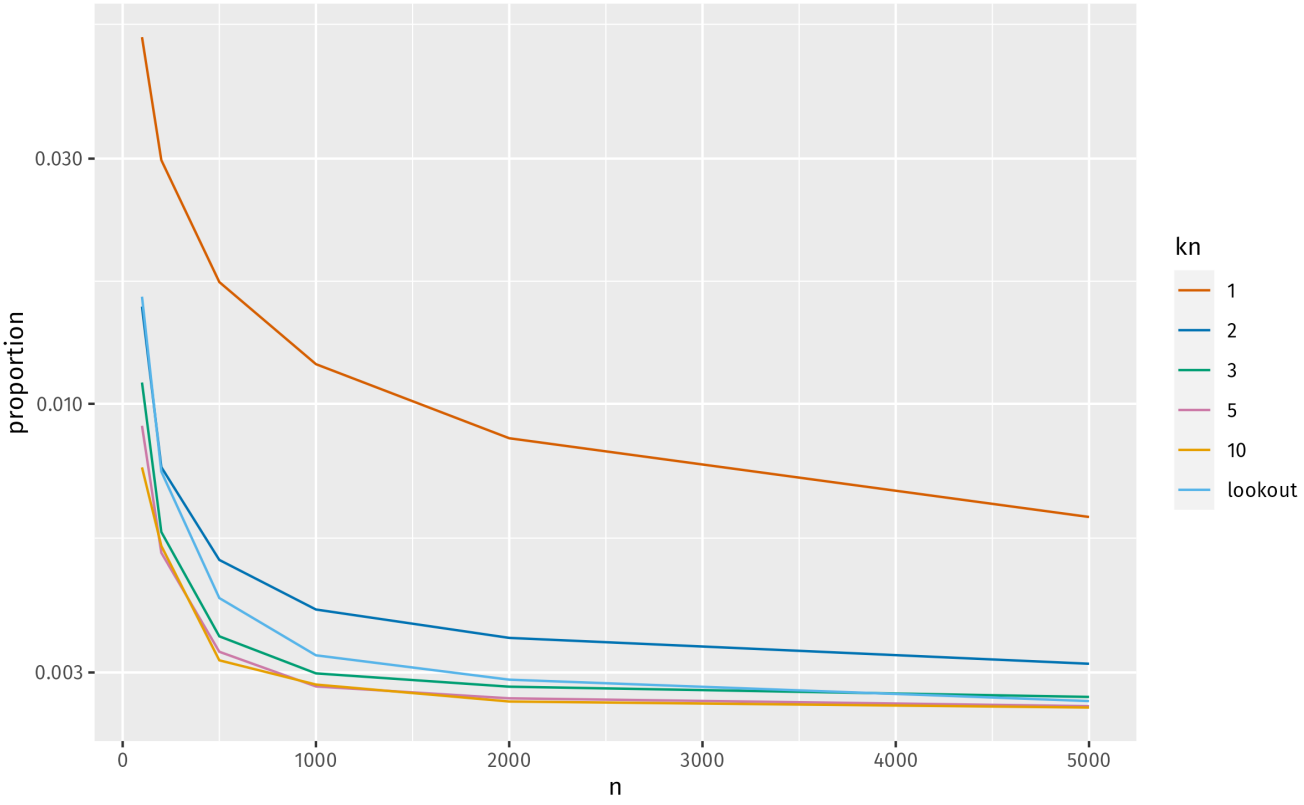
```
#> n = 1000 kn = 1
#> n = 1000 kn = 2
#> n = 1000 kn = 3
#> n = 1000 kn = 5
```

```
#> n = 1000 kn = 10
#> n = 2000 kn = 1
#> n = 2000 kn = 2
#> n = 2000 kn = 3
#> n = 2000 kn = 5
#> n = 2000 kn = 10
#> n = 5000 kn = 1
#> n = 5000 kn = 2
#> n = 5000 kn = 3
#> n = 5000 kn = 5
#> n = 5000 kn = 10
```

```
for(i in seq(NROW(sim_tda))) {
  cat(paste("n =", sim_tda$n[i], "\n"))
  for(j in seq(nreps)) {
    y <- rmvnorm(sim_tda$n[i], sigma = diag(d))
    H <- kde_bandwidth(y, method = "lookout")
    scores <- weird:::calc_kde_scores(y, H=H)
    lookout_prob <- lookout(density_scores = scores$scores, loo_scores = scores$loo_scores)
    # Number of false anomalies
    no_false[j] <- sum(lookout_prob < 0.05)
  }
  # Average number of false anomalies
  sim_tda$nfalse[i] <- mean(no_false)
}
```

```
#> n = 100
#> n = 200
#> n = 500
#> n = 1000
#> n = 2000
#> n = 5000
```

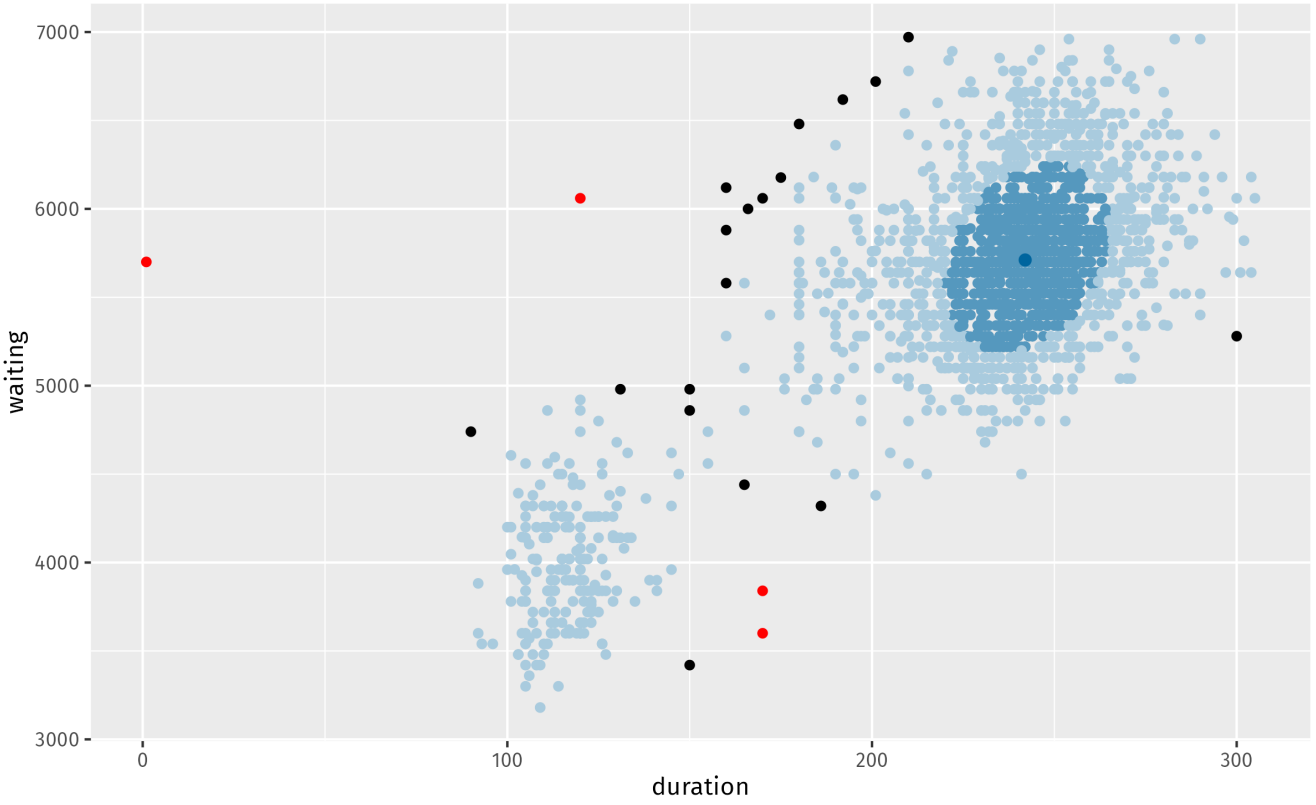
```
# Plot proportion of false anomalies vs n
sim_kn |>
  mutate(kn = as.character(kn)) |>
  bind_rows(
    sim_tda |> mutate(kn = "lookout")
  ) |>
  mutate(
    proportion = nfalse/n,
    kn = factor(kn, levels=c("1","2","3","5","10","lookout"))) |>
  ggplot(aes(x = n, y = proportion)) +
    geom_line(aes(col = kn, group = kn)) +
  scale_y_log10()
```



Some examples

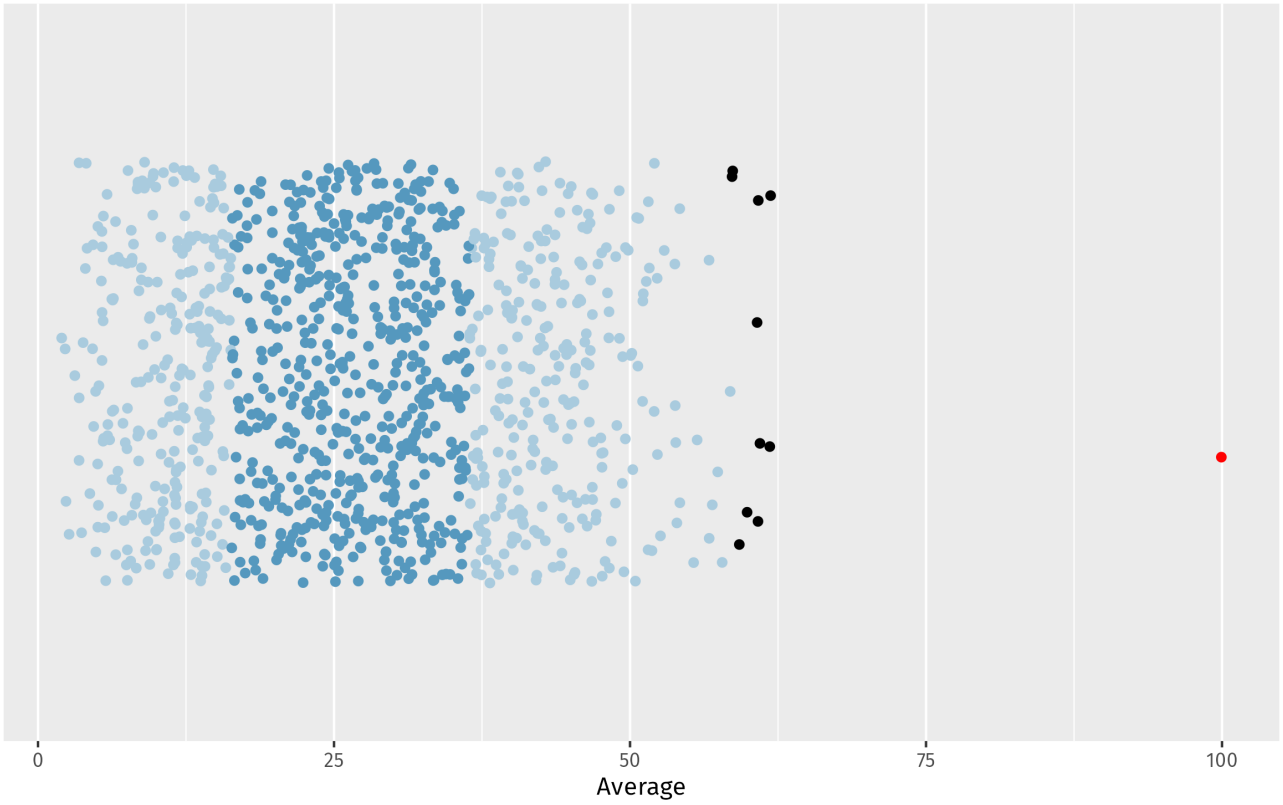
Old Faithful data

```
of <- oldfaithful |>
  filter(duration < 7000, waiting < 7000) |>
  select(duration, waiting)
gg_hdrboxplot(of, duration, waiting, scatterplot = TRUE,show_lookout = TRUE,
    H = kde_bandwidth(of, multiplier = 3))
```



Cricket batting averages

```
bat_ave <- cricket_batting |>
  filter(Innings > 20)
bat_ave |>
  gg_hdrboxplot(Average, show_lookout = TRUE, scatterplot = TRUE,
      h = kde_bandwidth(bat_ave$Average, multiplier = 3))
```



References

Gnanadesikan, R., and J. R. Kettenring. 1972. "Robust Estimates, Residuals, and Outlier Detection with Multiresponse Data." *Biometrics* 28 (1): 81–124.

Maechler, M, P Rousseeuw, C Croux, V Todorov, A Ruckstuhl, M Salibian-Barrera, T Verbeke, M Koller, E L T Conceicao, and M A di Palma. 2023. *robustbase: Basic Robust Statistics*. http://robustbase.r-forge.r-project.org/.

Maronna, R. A., and R. H. Zamar. 2002. "Robust Estimates of Location and Dispersion of High-Dimensional Datasets." *Technometrics* 44 (4): 307–17.

Scott, D W. 2015. *Multivariate Density Estimation: Theory, Practice, and Visualization*. 2nd ed. Wiley.