A very short inroduction into the aws package

Jörg Polzehl

January 11, 2021

This document illustrates the capabilities of the **aws** package. For a more comprehensive overview and more realistic examples we refer to **PoPaTa18** and appendix A of **MRBIbook2019**.

The package contains functions for adaptive smoothing (filtering) in various settings, including regular (and irregular) designs in 1D, 2D, 3D and SE(3), for univariate and vector valued observations.

The code in this vignette is restricted to 1D and 2D examples due to limited computation time.

1 Some artificial data

First, we generate some artificial examples:

The first example employs a local constant univariate regression function

```
> library(aws)
> fofx1 <- c(rep(0,25),rep(-1,20),rep(1,20), rep(-2,10),rep(2,5),
+            rep(-1,25),rep(-.5,30),rep(0,35))
> set.seed(1)
> y1 <- rnorm(fofx1,fofx1,.3)
```

The second example uses a local constant image

```
> u1 <- matrix(0,64,64)
> ind0 <- seq(0,1,length=64)
> ind <- outer(ind0^2,ind0^2,"+")
> u1[ind  > .95] <- u1[ind >.95] + 2
> u1[ind < .6] <- u1[ind < .6] -2
> u1[ind < .35] <- u1[ind < .35] +3
> u1[ind < .15] <- u1[ind < .15] -2
> u1[ind < .05] <- u1[ind < .05] +3
> u1 <- u1*(1-2*outer(ind0,ind0,">"))
> z1 <- u1+rnorm(u1)
```

In the third example a smooth image is added leading to a locally smooth image

```
> u2 <- u1+5*ind
> z2 <- u2+rnorm(u1)
```

2 Nonparametric smoothing

Nonparametric smoothing using FFT is implemented in function `kernsm` for 1D, 2D and 3D data.

```
> yhat0 <- kernsm(y1, h=10)
```

3 Adaptive weights smoothing

Function `aws` implements the structural adaptive smoothing methods developed in **PoSp00** and **PoSp05**.

```
> yhat1 <- aws(y1, hmax=100)

> par(mfrow=c(1,3), mar=c(3,3,3,1), mgp=c(2,1,0))
> plot(y1)
> lines(yhat1@theta, col=2)
> lines(fofx1, col=3)
> title("AWS estimate")
> plot(yhat1@ni)
> title("Sum of weights")
> plot(y1)
> lines(kernsm(y1,.609)@yhat, col=2)
> lines(fofx1, col=3)
> title("MSE optimal kernel estimate")
```
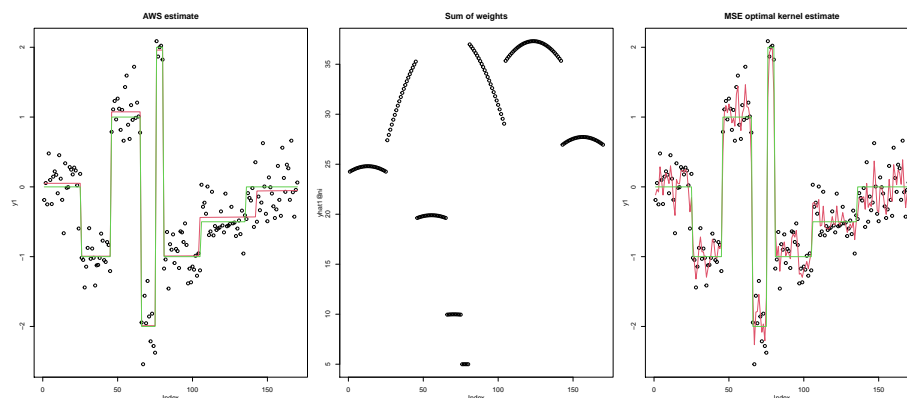

The left Figure shows the the data, estimated regression function (red) in comparison to the true function (green). The central panel provides, for each design point, the sum of weights emploid in the last step of the AWS algorithm, while the right panel illustrates the behaviour of a kernel estimate with MSE optimal bandwidth.

For the 2D examples we employ both the functions `aws` and `paws`. The latter function implements the patchwise adaptive weights algorithm described in **PoPaTa18**.

```
> setCores(2)
> zhat1a <- aws(z1, hmax=8)
> zhat1b <- paws(z1, hmax=10, patchsize=1)
```

```
> par(mfrow=c(2,3), mar=c(3,3,3,1), mgp=c(2,1,0))
> image(z1, col=grey(0:255/255))
> title("Noisy original")
> image(zhat1a@theta, col=grey(0:255/255))
> title("AWS reconstruction")
> image(zhat1a@ni, col=grey(0:255/255))
> title("AWS sum of weights")
> image(u1, col=grey(0:255/255))
> title("True image")
> image(zhat1b@theta, col=grey(0:255/255))
> title("PAWS reconstruction")
> image(zhat1b@ni, col=grey(0:255/255))
> title("PAWS sum of weights")
```
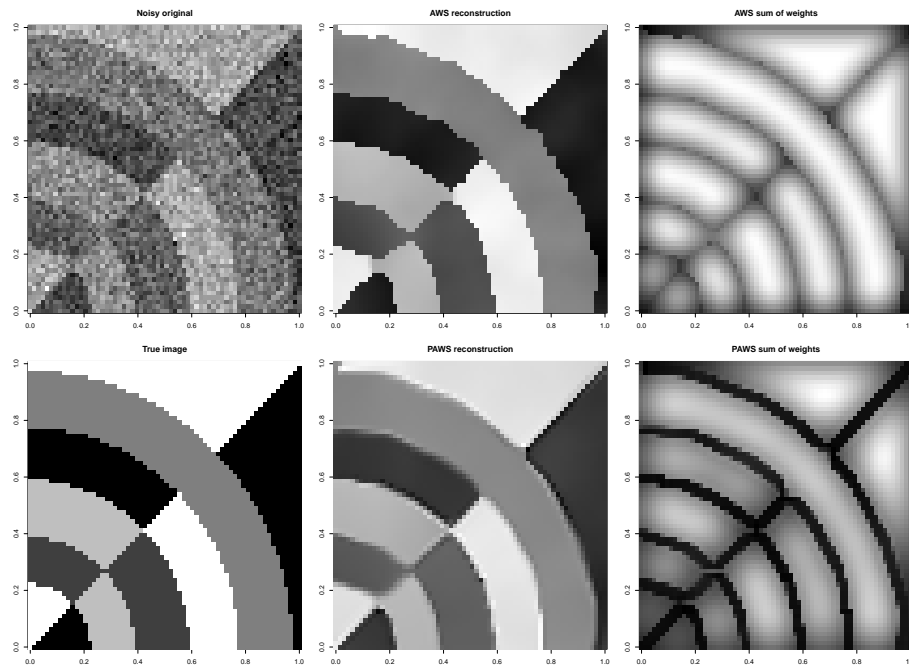

The Figure illustrates the results obtained using both methods in comparison with the noisy original and the true image.

To illustrate the dependence of the obtained reconstruction quality we use the second, locally smooth, 2D example.

```
> zhat2a <- aws(z2, hmax=8)
> zhat2b <- paws(z2, hmax=10)

> par(mfrow=c(2,3), mar=c(3,3,3,1), mgp=c(2,1,0))
> image(z2, col=grey(0:255/255))
> title("Noisy original")
> image(zhat2a@theta, col=grey(0:255/255))
> title("AWS reconstruction")
> image(zhat2a@ni, col=grey(0:255/255))
> title("AWS sum of weights")
```
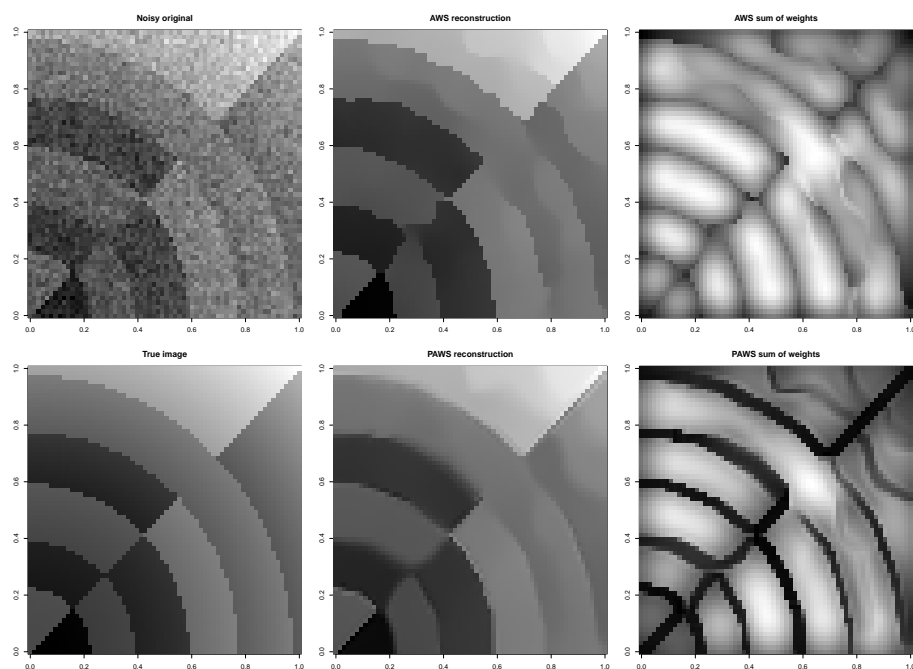
```
> image(u2, col=grey(0:255/255))
> title("True image")
> image(zhat2b@theta, col=grey(0:255/255))
> title("PAWS reconstruction")
> image(zhat2b@ni, col=grey(0:255/255))
> title("PAWS sum of weights")
```


Note that AWS enforces the structural assumption of a local constant image if large maximal bandwidths are used. This drawback is overcome in PAWS which allows for smooth image gradients and prefers smooth discontinuities.

Both functions handle 1D, 2D and 3D images.

4 Intersection of confidence intervals

The package also containes functions implementing the Intersection of confidence intervals approach from **katkov06**. The approach is based on adaptation techniques that combine results obtained by kernel smoothing for a sequence of bandwidths and for orientation (sector) dependent support of the kernel.

```
> zhat1c <- kernsm(z1,.9)@yhat
> zhat1d <- ICIsmooth(z1, hmax=8, thresh=.8, presmooth=TRUE)@yhat
> zhat1e <- ICIcombined(z1, hmax=8, nsector=8, thresh=.8,
+                       presmooth=TRUE)@yhat
```

We here apply sets of parameters choosen to provide good MSE for reconstruction results.
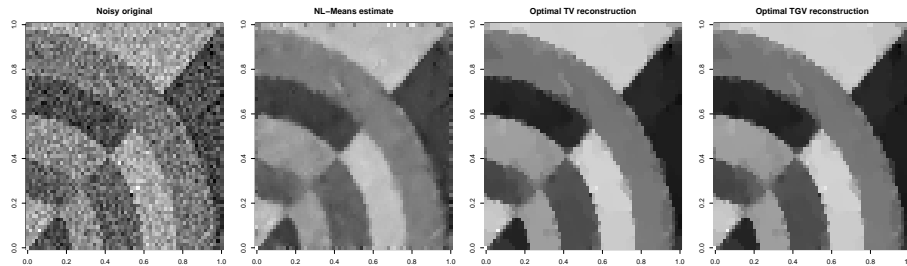
```
> par(mfrow=c(1,4), mar=c(3,3,3,1), mgp=c(2,1,0))
> image(z1, col=grey(0:255/255))
```

```
> title("Noisy original")
> image(zhat1c, col=grey(0:255/255))
> title("optimal kernel estimate")
> image(zhat1d, col=grey(0:255/255))
> title("adaptation over h")
> image(zhat1e, col=grey(0:255/255))
> title("adaptation over h and sectorial")
```


| Noisy original | NL–Means estimate | Optimal TV reconstruction | Optimal TGV reconstruction |

5 Non-local means filter

For comparisons the NL Means algorithm (**Coupe08**), (**Coupe12**) in 1D, 2D and 3D is provided with function `nlmeans`.

```
> zhat1f <- nlmeans(z1, .85, 1, searchhw=6)$theta
```

6 Total variation methods

Additionally functions `TV_denoising` and `TGV_denoising` implement total variation (**rudin1992nonlinear**) and total generalized variation (**TGV**) methods for image denoising in 2D.

```
> zhat1f <- TV_denoising(z1, .93)

238 Chambolle-Pock iterations completed

> zhat1g <- TGV_denoising(z1, .92, 4)

1000 Chambolle-Pock iterations completed

> par(mfrow=c(1,4), mar=c(3,3,3,1), mgp=c(2,1,0))
> image(z1, col=grey(0:255/255))
> title("Noisy original")
> image(zhat1e, col=grey(0:255/255))
> title("NL-Means estimate")
> image(zhat1f, col=grey(0:255/255))
> title("Optimal TV reconstruction")
> image(zhat1g, col=grey(0:255/255))
> title("Optimal TGV reconstruction")
```
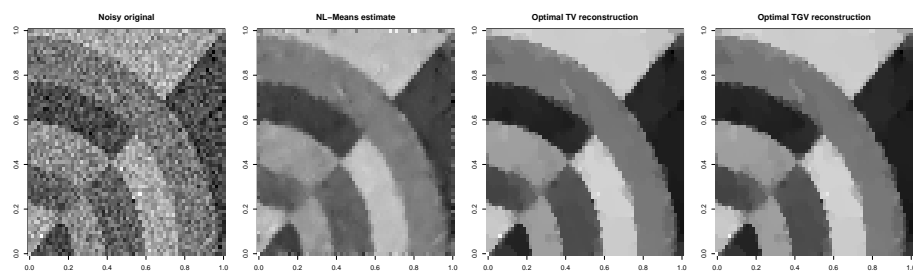
| Noisy original | NL–Means estimate | Optimal TV reconstruction | Optimal TGV reconstruction |

The figure provides reconstructions for the first (local constant) 2D example using NL Means, TV and TGV denoising. For all three methods parameters are optimized for the data at hand.

7 Other content

The package **aws** also contains functions for locally adaptive variance estimation, versions of AWS and PAWS for vector valued data (used e.g. in package **qMRI**) and AWS methods for data in SE(3) (these methods are used for smoothing of diffusion weighted data in package **dti**). Most of the computationally intensive code is parallelized using openMP.