Package `widals`

Dave Zes

March 2, 2014

1 Intro: What the Heck Is "WIDALS"?

WIDALS stands for weighting by inverse distance with adaptive least squares. WIDALS is an algorithm that can be used to predict space-time data where sensors have fixed location, and the time intervals are (about) equal. The WIDALS algorithm is — comparatively — very fast, meaning that it is well suited for very large data sets, e.g., the number of time points is in the hundreds or thousands, and the number of spacial locations is in the thousands or even tens-of-thousands.

The algorithm, in essence, is brutishly simple. An initial adaptive least squares (ALS) stage fits inputs (covariates) to the observations (data). The second stage attempts to improve prediction quality by summing to the ALS predictions a "stochastic adjustment" calculated from spacial correlation present in the ALS residuals.

WIDALS is fairly easy to fit, very robust, and has the side-effect, because of the adaptive least squares (ALS) stage, of offering the user a sense of how strongly inputs (the user supplied covariates) contribute to the observations.

The bad news is that WIDALS is purely pragmatic. Almost no theory exists describing properties of the WIDALS predictors — what we mean by this is that given data that arose from the sort of generative system commonly regarded as sufficiently flexible to describe phenomenon of interest, like, say, air pollution, little if nothing is known about the theoretical properties of WIDALS's predictions. That said, the silver lining is that *if* we are in fact handling a massive data set, we can use its size to our advantage. We can, for example, use empiric techniques, such as resampling, to get a sense of how well WIDALS is performing.

The most remarkable feature of WIDALS, though, is its tiny hyperparameter space. A full model can be fit with a mere 5 non-negative values (2 ALS, 3 weighting). This implies something more:

multiple ALS stages can be run, either in parallel or in series, or in some combination, akin to *path-model* solutions. This extensibility is not explored in the `R` documentation *per se*, but we will give it a try here in these vignettes.

Finally, before diving in, the user should be cautioned that while, given hyperparameters, this package makes it very easy to interpolate (predict to unmonitored locations), providing means of locating these hyperparameters, i.e., *fitting*, brings some coding complexities. In this package, we fit WIDALS using (metaheuristic) stochastic search using `MSS.snow()`. This function creates functions using `fun.load()`. Both call upon variables out-of-scope. This property may madden a CS "best-practices" user, but by abiding certain cautions, potential mis-assignments are easily avoided.

For any given `R` session, only work with `widals`.

Prior to fitting with `MSS.snow()`, make sure all these values are correctly assigned (and living in the Global Environment):

`Z` — should always be the data matrix of supporting sites.
`Z.na` — if used, should always be a boolean matrix indicating missing locations in `Z`.
`locs` — should always be the spacial locations associated with `Z`.
`Hs` — should always be the spacial covariates associated with `Z`.
`Ht` — should always be the temporal covariates associated with `Z`.
`Hst.ls` — should always be the space-time covariates associated with `Z`.
These are additional arguments and should be correctly assigned prior to calling `MSS.snow()`:
`lags`
`b.lag`
`cv`
`xgeodesic`
`ltco`
`GP`
`run.parallel`
`stnd.d`
`train.rng`
`test.rng`

2 ALS Examples

Let's jump in and work with the included Californis Ozone [1] demonstration data set.

Important: It is intended that the User run through *all* the code in this Section and the next.

```
options(stringsAsFactors=FALSE)
k.cpus <- 2 #### set the number of cpus for snowfall
library(widals)
data(O3)
Z.all <- as.matrix(O3$Z)[366:730, ]
locs.all <- O3$locs[ , c(2,1)]
hsa.all <- O3$helevs
xdate <- rownames(Z.all)
tau <- nrow(Z.all)
n.all <- ncol(Z.all)
xgeodesic <- TRUE
```

Notice above that we use `Z.all`, `locs.all`, `hsa.all`, `n.all`, to hold our "full" data, locations, spacial covariate, number of sites, respectively.

From these, we can make assignments to the objects needed to fit our ALS model:

```
Z <- Z.all
locs <- locs.all
n <- n.all
dateDate <- strptime(xdate, "%Y%m%d")
doy <- as.integer(format(dateDate, "%j"))
Ht <- cbind( sin(2*pi*doy/365), cos(2*pi*doy/365) )
Hs.all <- matrix(1, nrow=n.all)
Hst.ls.all <- NULL
Hs <- Hs.all
Hst.ls <- Hst.ls.all
Ht.original <- Ht
##########################
rm.ndx <- create.rm.ndx.ls(n, 14)
b.lag <- 0
train.rng <- 30:tau
```

```
test.rng <- train.rng
GP <- c(1/10, 1)
k.glob <- 9
rho.upper.limit <- 100
rgr.lower.limit <- 10^(-7)
sds.mx <- seq(2, 0.01, length=k.glob) * matrix(1, k.glob, length(GP))
run.parallel <- TRUE
sfInit(TRUE, k.cpus)
FUN.source <- fun.load.hals.a
FUN.source()
MSS.snow(FUN.source, NA, p.ndx.ls, f.d, sds.mx=sds.mx,
  k.glob, k.loc.coef=7, X = NULL)
sfStop()
```

Let's confirm the observation RMSE:

```
Z.als <- Hals.snow(1, Z = Z, Hs = Hs, Ht = Ht, Hst.ls = Hst.ls,
  b.lag = b.lag, GP.mx = matrix(GP, 1, 2))
resids <- Z-Z.als
sqrt( mean( resids[test.rng, ]^2 ) )
```

Add site elevations to the spacial covariates:

```
Hs.all <- cbind(matrix(1, nrow=n.all), hsa.all)
Hs <- Hs.all
GP <- c(1/10, 1)
sfInit(TRUE, k.cpus)
MSS.snow(FUN.source, NA, p.ndx.ls, f.d, sds.mx=sds.mx,
  k.glob, k.loc.coef=7, X = NULL)
sfStop()
```

Add incident solar area (ISA) to the space-time covariates:

```
Hst.ls.all <- H.Earth.solar(locs[ , 2], locs[ , 1], dateDate)
Hst.ls <- Hst.ls.all
```

```
GP <- c(1/10, 1)
sfInit(TRUE, k.cpus)
MSS.snow(FUN.source, NA, p.ndx.ls, f.d, sds.mx=sds.mx,
  k.glob, k.loc.coef=7, X = NULL)
sfStop()
```

Add the ISA-elevation interaction to the space-time covariates:

```
Hst.ls.all2 <- list()
for(tt in 1:tau) {
      Hst.ls.all2[[tt]] <- cbind(Hst.ls.all[[tt]], Hst.ls.all[[tt]]*hsa.all)
      colnames(Hst.ls.all2[[tt]]) <- c("ISA", "ISAxElev")
  }
Hst.ls <- Hst.ls.all2
GP <- c(1/10, 1)
sfInit(TRUE, k.cpus)
MSS.snow(FUN.source, NA, p.ndx.ls, f.d, sds.mx=sds.mx,
  k.glob, k.loc.coef=7, X = NULL)
sfStop()
```

Let's look at the "effective prediction errors" for the ALS partial slopes:

```
Hals.ses(Z, Hs, Ht, Hst.ls, GP[1], GP[2], b.lag, test.rng)
```

Does standardizing the covariates help?:

```
Z <- Z.all
Hst.ls <- stnd.Hst.ls(Hst.ls.all2)$sHst.ls
Hs <- stnd.Hs(Hs.all)$sHs
Ht <- stnd.Ht(Ht, nrow(Hs))
GP <- c(1/10, 1)
sfInit(TRUE, k.cpus)
MSS.snow(FUN.source, NA, p.ndx.ls, f.d, sds.mx=sds.mx,
  k.glob, k.loc.coef=7, X = NULL)
sfStop()
```

How about standardizing the response?:

```
z.mean <- mean(Z.all)
z.sd <- sd(as.vector(Z.all))
Z <- (Z.all - z.mean) / z.sd
GP <- c(1/10, 1)
sfInit(TRUE, k.cpus)
MSS.snow(FUN.source, NA, p.ndx.ls, f.d, sds.mx=sds.mx,
  k.glob, k.loc.coef=7, X = NULL)
sfStop()
our.cost * z.sd
```

3 WIDALS Examples

Now, WIDALS.

We'll start by fitting using cross-validation (the same `rm.ndx` list we've been using with ALS above). This will take a few minutes.

```
Z <- Z.all
Hst.ls <- Hst.ls.all2
Hs <- Hs.all
Ht <- Ht.original
FUN.source <- fun.load.widals.a
d.alpha.lower.limit <- 0
GP <- c(1/1000, 1, 0.01, 3, 1)
cv <- 2
lags <- c(0)
b.lag <- 0
sds.mx <- seq(2, 0.01, length=k.glob) * matrix(1, k.glob, length(GP))
ltco <- -10
```

```
stnd.d <- TRUE
sfInit(TRUE, k.cpus)
FUN.source()
MSS.snow(FUN.source, NA, p.ndx.ls, f.d, sds.mx=sds.mx,
  k.glob, k.loc.coef=7, X = NULL)
sfStop()
```

Try a fast WIDALS fit using pseudo CV (set `cv` to -2, set `b.lag` to -1).

```
rm.ndx <- 1:n
Z <- Z.all
Hst.ls <- Hst.ls.all2
Hs <- Hs.all
Ht <- Ht.original
FUN.source <- fun.load.widals.a
d.alpha.lower.limit <- 0
GP <- c(1/1000, 1, 0.01, 3, 1)
cv <- -2
lags <- c(0)
b.lag <- -1
sds.mx <- seq(2, 0.01, length=k.glob) * matrix(1, k.glob, length(GP))
ltco <- -10
stnd.d <- TRUE
sfInit(TRUE, k.cpus)
FUN.source()
MSS.snow(FUN.source, NA, p.ndx.ls, f.d, sds.mx=sds.mx,
  k.glob, k.loc.coef=7, X = NULL)
sfStop()
```

```
Hals.ses(Z, Hs, Ht, Hst.ls, GP[1], GP[2], b.lag, test.rng)
```

Plot interpolation:

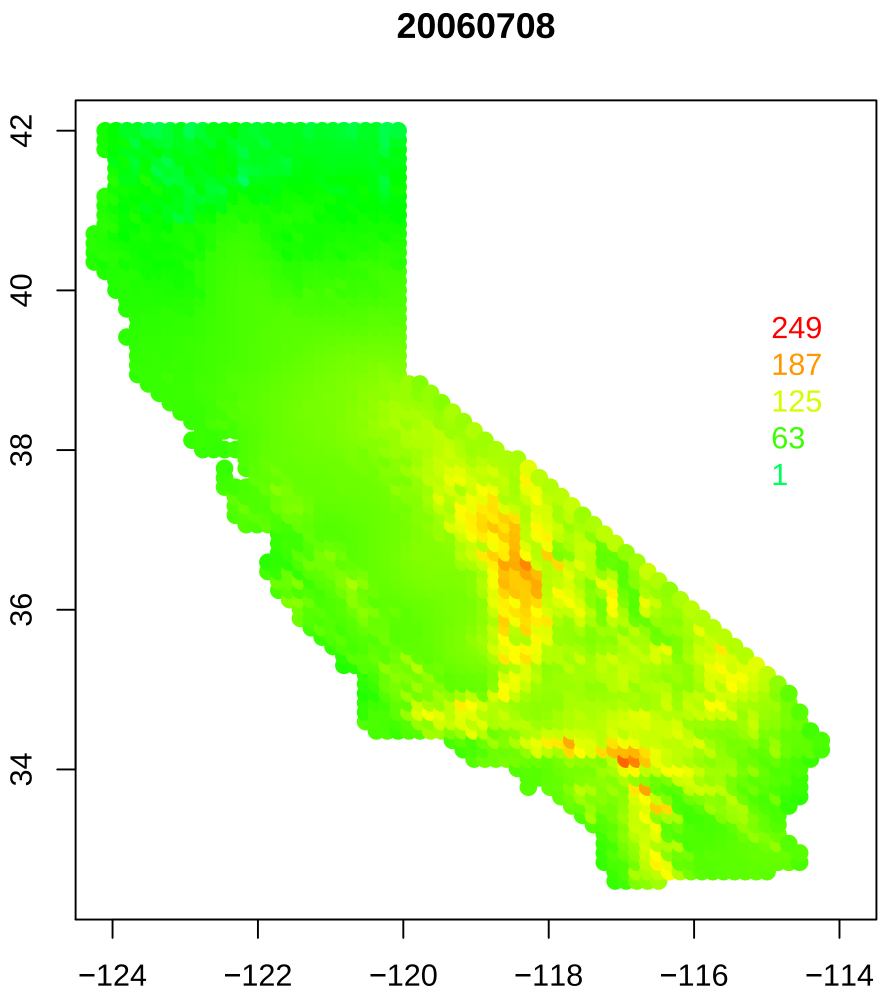
```
Z <- Z.all
Hst.ls <- Hst.ls.all2
```

```
Hs <- Hs.all
Ht <- Ht.original
Hs0 <- cbind(matrix(1, length(O3$helevs0)), O3$helevs0)
Hst0isa.ls <- H.Earth.solar(O3$locs0[ , 1], O3$locs0[ , 2], dateDate)
Hst0.ls <- list()
for(tt in 1:tau) {
          Hst0.ls[[tt]] <- cbind(Hst0isa.ls[[tt]], Hst0isa.ls[[tt]]*O3$helevs0)
          colnames(Hst0.ls[[tt]]) <- c("ISA", "ISAxElev")
  }
locs0 <- O3$locs0[ , c(2,1)]
Z0.hat <- widals.predict(Z, Hs, Ht, Hst.ls, locs, lags, b.lag, Hs0, Hst0.ls,
  locs0=locs0, geodesic=xgeodesic, wrap.around=NULL, GP, stnd.d, ltco)[10:tau, ]
ydate <- xdate[10:tau]
#xcol.vec <- heat.colors(max(round(Z0.hat)))
#xcol.vec <- rev(rainbow(max(round(Z0.hat))))
xcol.vec <- rev(rainbow(630)[1:max(round(Z0.hat))])
xleg.vals <- round( seq(1, max(Z0.hat)-1, length=(5)) / 1 ) * 1
xleg.cols <- xcol.vec[xleg.vals+1]



for(tt in 1:nrow(Z0.hat)) {
      plot(0, 0, xlim=c(-124.1, -113.9), ylim=c(32.5, 42), type="n", main=ydate[tt])
      ## points(locs0[ , c(2,1)], cex=Z0.hat[tt,] / 30) ## uncomment to see sites
      this.zvec <- round(Z0.hat[tt,])
      this.zvec[ this.zvec < 1] <- 1
      this.color <- xcol.vec[ this.zvec ]
      points(locs0[ , c(2,1)], cex=1.14,  col=this.color, pch=19 )
      #points(locs[ , c(2,1)], cex=Z[tt,] / 30, col="red")
      legend(-116, 40, legend=rev(xleg.vals), fill=FALSE, col=rev(xleg.cols),
      border=NA, bty="n", text.col=rev(xleg.cols))
          Sys.sleep(0.1)
  }
```

```
ydate <- xdate[10:tau]
tt <- 180
plot(0, 0, xlim=c(-124.1, -113.9), ylim=c(32.5, 42), type="n", main=ydate[tt],
  xlab="", ylab="")
## points(locs0[ , c(2,1)], cex=Z0.hat[tt,] / 30) ## uncomment to see sites
this.zvec <- round(Z0.hat[tt,])
this.zvec[ this.zvec < 1] <- 1
this.color <- xcol.vec[ this.zvec ]
points(locs0[ , c(2,1)], cex=1.14,  col=this.color, pch=19 )
#points(locs[ , c(2,1)], cex=Z[tt,] / 30, col="red")
legend(-116, 40, legend=rev(xleg.vals), fill=FALSE, col=rev(xleg.cols), border=NA,
  bty="n", text.col=rev(xleg.cols))
```



References

[1] California Air Resources Board. California Air Resources Board DVD-ROM. http://www.arb.ca.gov/aqd/aqdcd/aqdcd.htm, 2011. [Online; last accessed 2012-11-04].