

Quick start guide for the `grpreg` package

Patrick Breheny

July 11, 2016

This guide is intended to briefly demonstrate the basic usage of `grpreg`. For more details, see the other vignettes, documentation for individual functions, and the references.

`grpreg` comes with an example data set, `Birthwt`. The outcome, `Birthwt$bwt`, records the birth weights (in kg) of 189 babies. The following predictors are available:

```
> data(Birthwt)
> head(Birthwt$X, n=3)
```

	age1	age2	age3	lwt1	lwt2
[1,]	-0.05833434	0.011046300	0.02956182	0.12446282	-0.02133871
[2,]	0.13436561	0.055245529	-0.09690705	0.06006722	-0.06922831
[3,]	-0.04457006	-0.009415469	0.04508877	-0.05918388	0.03746349

	lwt3	white	black	smoke	ptl1	ptl2m	ht	ui	ftv1	ftv2	ftv3m
[1,]	-0.130731102	0	1	0	0	0	0	1	0	0	0
[2,]	-0.033348413	0	0	0	0	0	0	0	0	0	1
[3,]	0.004618178	1	0	1	0	0	0	0	1	0	0

This is a design matrix derived from the original data set, in which several terms have been expanded. For example, there are multiple indicator functions for race (“other” being the reference group) and several continuous factors such as age have been expanded using polynomial contrasts (splines would give a similar structure). Hence, the columns of the design matrix are *grouped*; this is what `grpreg` is designed for. The grouping information is encoded as follows:

```
> Birthwt$group
```

[1]	age	age	age	lwt	lwt	lwt	race	race	smoke	ptl	ptl
[12]	ht	ui	ftv	ftv	ftv						

Levels: age lwt race smoke ptl ht ui ftv

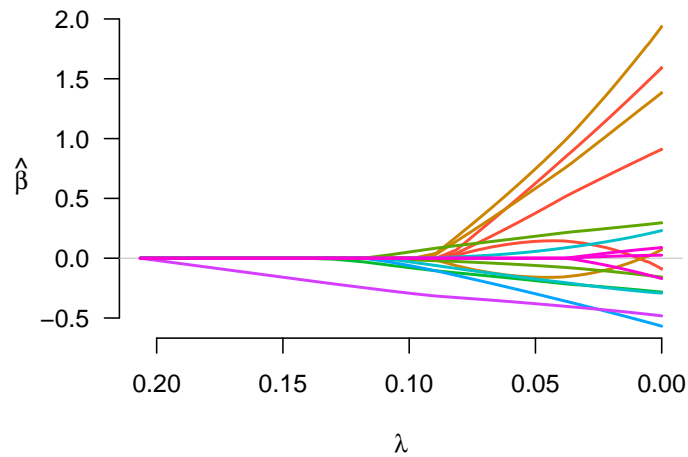
Here, groups are given as a factor; unique integer codes (which are essentially unlabeled factors) and character vectors are also allowed¹. To fit a group lasso model to this data:

```
> X <- Birthwt$X
> y <- Birthwt$bwt
> group <- Birthwt$group
> fit <- grpreg(X, y, group, penalty="grLasso")
```

We can then plot the coefficient paths with

¹Character vectors are not an ideal choice, however, as the order of the groups is left unspecified, which can lead to ambiguity if you also try to set the `group.multiplier` option.

```
> plot(fit)
```



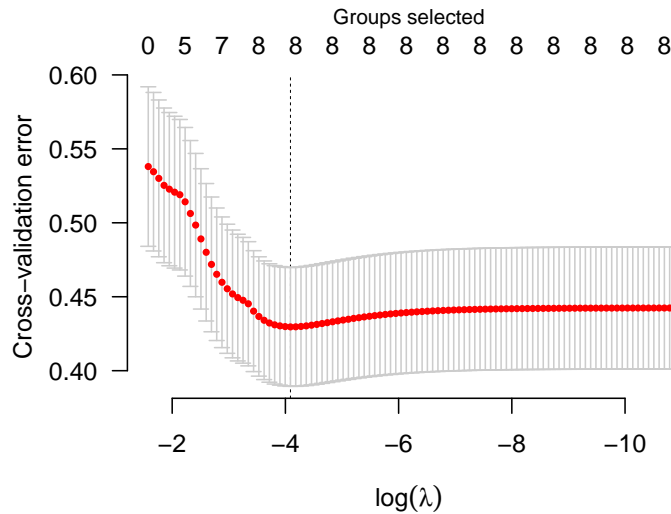
Notice that when a group enters the model (e.g., the green group), all of its coefficients become nonzero; this is what happens with group lasso models. To see what the coefficients are, we could use the `coef` function:

```
> coef(fit, lambda=0.05)
```

(Intercept)	age1	age2	age3	lwt1	lwt2
3.02894394	0.14043439	0.62607918	0.37679709	0.74715011	-0.15840434
lwt3	white	black	smoke	ptl1	ptl2m
0.58299107	0.18338287	-0.06109341	-0.18774399	-0.17425432	0.05710309
ht	ui	ftv1	ftv2	ftv3m	
-0.29775401	-0.38050515	0.00000000	0.00000000	0.00000000	

Note that the number of physician's visits (`ftv`) is not included in the model at $\lambda = 0.05$. Typically, one would carry out cross-validation for the purposes of carrying out inference on the predictive accuracy of the model at various values of λ .

```
> cvfit <- cv.grpreg(X, y, group, penalty="grLasso")
> plot(cvfit)
```



The coefficients corresponding to the value of λ that minimizes the cross-validation error can be obtained via `coef`:

```
> coef(cvfit)

(Intercept)      age1      age2      age3      lwt1      lwt2
 3.04031026  0.05717046  1.25373904  0.74067171  1.49289670 -0.06702824
      lwt3      white      black      smoke      ptl1      ptl2m
 1.10635750  0.25892223 -0.11827031 -0.25089892 -0.25912825  0.15649092
      ht      ui      ftv1      ftv2      ftv3m
-0.47299817 -0.44390536  0.05384583  0.01801390 -0.08269463
```

Predicted values can be obtained via `predict`, which has a number of options:

```
> predict(cvfit, X=head(X))

[1] 2.553149 3.020204 3.035466 2.568439 2.580925 3.080860

> predict(cvfit, type="ngroups")

[1] 8
```

Note that the original fit (to the full data set) is returned as `cvfit$fit`; it is not necessary to call both `grpreg` and `cv.grpreg` to analyze a data set. Several other penalties are available, as are methods for logistic regression and Cox proportional hazards regression.