# Nonconvex penalties: Algorithms and case studies

Patrick Breheny

March 6, 2025

# Introduction

- In today's lecture, we will discuss the performance of nonconvex penalties with respect to the signal-to-noise ratio of the data-generating process, the most critical factor determining their success relative to the lasso
- We will then turn our attention to the details of model fitting, discussing algorithms for nonconvex penalties as well as the impact of nonconvexity on model-fitting

# Signal to noise ratio

- For linear regression,

$$\begin{aligned}\text{Var}(Y) &= \text{Var}(\text{E}(Y|X)) + \text{E}(\text{Var}(Y|X)) \\ &= \boldsymbol{\beta}^\top \text{Var}(X)\boldsymbol{\beta} + \sigma^2\end{aligned}$$

- The first term in the sum is known as the *signal* and the second term the *noise*
- Thus, we may define the *signal-to-noise ratio*

$$\text{SNR} = \boldsymbol{\beta}^\top \text{Var}(X)\boldsymbol{\beta}/\sigma^2$$

# SNR and $R^2$

- Recall that we have seen this decomposition before, in calculating $R^2$, which is also a function of the signal and noise
- In particular, note that

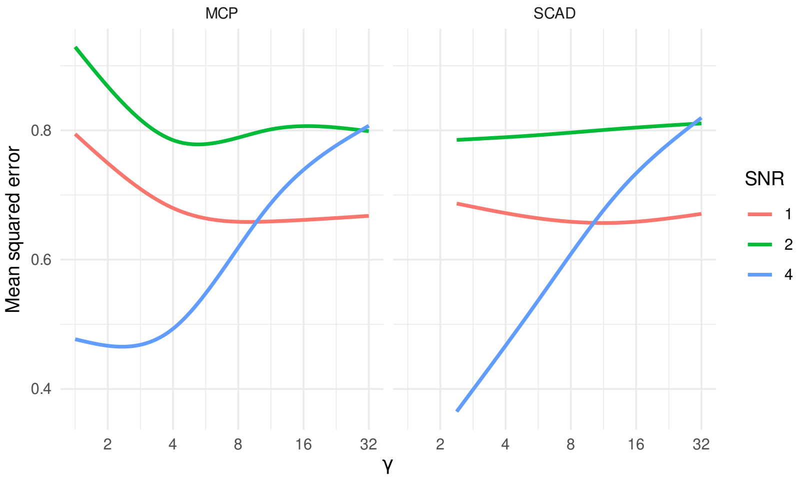
$$R^2 = \frac{\text{SNR}}{1 + \text{SNR}}$$

- As a general piece of advice, I strongly recommend considering the signal-to-noise ratio when designing simulations, and avoiding settings where SNR is, say, 50 ($R^2 = .98$); is this realistic?

# Simulation: Setup

- To see the impact of SNR, let's set $n = 50$, $p = 100$, and let all features $\mathbf{x}_j$ follow independent, standard Gaussian distributions
- In the generating model, we set $\beta_1 = \beta_2 = \beta_3 = \cdots = \beta_6 \neq 0$ and $\beta_7 = \beta_8 = \cdots = \beta_{100} = 0$, varying the nonzero values of $\beta_1$ through $\beta_6$ to produce a range of signal to noise ratios
- For each data set, an independent data set of equal size was generated for the purposes of selecting the regularization parameter

# Simulation: Results



# Remarks

- The motivation of MCP/SCAD/etc. is to eliminate bias for large coefficients; it should not come as little surprise, then, that the advantage of these methods only becomes apparent when some nonzero coefficients are large
- It is also worth noting that $\gamma \approx 3$ is generally a reasonable choice for MCP – its performance was never far from the best
- Also note that the SCAD is somewhat less sensitive to the choice of $\gamma$, in the sense that many values of $\gamma$ produce rather lasso-like estimates

# Algorithm

Letting $\tilde{z} = n^{-1}\mathbf{x}_j^\top \tilde{\mathbf{r}}_j$, $F$ is the firm-thresholding operator, and $T_{\text{SCAD}}$ is the SCAD-thresholding operator, the CD algorithm for MCP/SCAD is

**repeat**

**for** $j = 1, 2, \ldots, p$

$$\tilde{z}_j = n^{-1}\sum_{i=1}^n x_{ij} r_i + \widetilde{\beta}_j^{(s)}$$

$$\widetilde{\beta}_j^{(s+1)} \leftarrow \begin{cases} F(\tilde{z}_j|\lambda,\gamma) & \text{for MCP, or} \\ T_{\text{SCAD}}(\tilde{z}_j|\lambda,\gamma) & \text{for SCAD} \end{cases}$$

$r_i \leftarrow r_i - (\widetilde{\beta}_j^{(s+1)} - \widetilde{\beta}_j^{(s)})x_{ij}$ for all $i$

**until** convergence

The algorithm is identical to our earlier algorithm for the lasso except for the step in which $\widetilde{\beta}_j$ is updated

# Convergence

- Although the MCP and SCAD penalties are not convex functions, $Q(\beta_j|\boldsymbol{\beta}_{-j})$ is still convex
- As a result, the coordinate-wise updates are unique and always occur at the global minimum with respect to that coordinate
- **Proposition:** Let $\{\boldsymbol{\beta}^{(s)}\}$ denote the sequence of coefficients produced at each iteration of the coordinate descent algorithms for SCAD and MCP. For all $s = 0, 1, 2, \ldots$,

$$Q(\boldsymbol{\beta}^{(s+1)}) \leq Q(\boldsymbol{\beta}^{(s)}).$$

  Furthermore, the sequence is guaranteed to converge to a local minimum of $Q(\boldsymbol{\beta})$.

# Local linear approximation

- For MCP and SCAD, one can obtain closed-form coordinate-wise minima and use those solutions as updates
- An alternative approach, particularly useful in penalties that do not yield tidy closed-form solutions, is to construct a local approximation of the penalty about a point $\widetilde{\beta}$:

$$P(|\beta|) \approx P(|\widetilde{\beta}|) + \dot{P}(|\widetilde{\beta}|)(|\beta| - |\widetilde{\beta}|)$$

- Note that with this approximation, the penalty takes on the form of the lasso penalty (with $\dot{P}(|\widetilde{\beta}|)$ playing the role of the regularization parameter) plus a constant

# LLA algorithm

- The approximation is applied in an iterative fashion: at the $s$th iteration, letting $\tilde{\lambda}_j = \dot{P}(|\beta_j^{(s-1)}|)$, the update is given by solving for the value minimizing

$$\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \sum_{j=1}^{p} \tilde{\lambda}_j |\beta_j|$$

- Note that this equation is essentially identical to the one for the adaptive lasso; however, the adaptive lasso weights are assigned in a more or less ad hoc fashion based on an initial estimator, while the LLA modifications to $\lambda$ are explicitly determined by the penalty function $P$

# Remarks

- Like coordinate descent, the local linear approximation (LLA) algorithm is guaranteed to drive the objective function downhill with every iteration and to converge to a local minimum of $Q(\beta)$
- For MCP and SCAD, CD is more efficient, as it avoids the extra approximation introduced by LLA
- However, LLA is still quite efficient, and a valuable alternative when dealing with penalties without a simple solution in the one-dimensional case

# Convexity challenges

- While the objective functions for SCAD and MCP are convex in each coordinate dimension, they are not convex over $\mathbb{R}^p$
- Thus, multiple minima may exist, each satisfying the KKT conditions
- Neither the CD or LLA algorithms are guaranteed to converge to the global minimum in such cases
- As we have discussed earlier, the existence of multiple minima poses problems, both numerically (convergence to an inferior solution) and statistically (increased variance as the solution jumps from one minima to another)

# Global convexity

- It is worth noting that it is possible for the objective function $Q$ to be convex with respect to $\beta$ even though the penalty component is nonconvex
- Letting $c_{\min}$ denote the minimum eigenvalue of $\mathbf{X}^\top\mathbf{X}/n$, the MCP objective function is strictly convex if $\gamma > 1/c_{\min}$, while the SCAD objective function is strictly convex if $\gamma > 1 + 1/c_{\min}$
- In this case, the coordinate descent and LLA algorithms will converge to the unique global minimum of $Q$

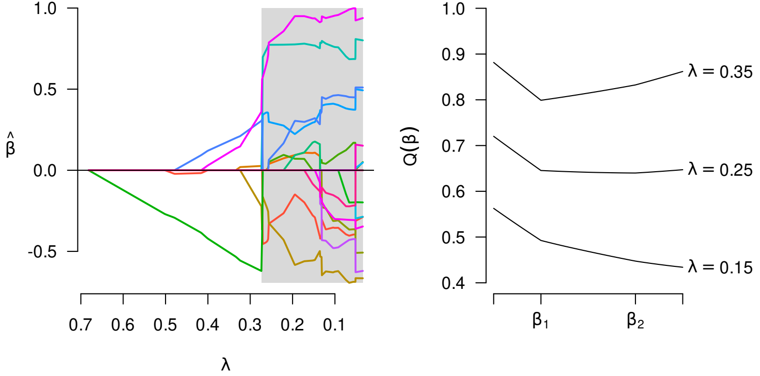
# Is global convexity desirable?

- However, obtaining strict convexity is not always possible or desirable; for example, in high-dimensional settings where $p > n$, $c_{\min} = 0$ and the MCP/SCAD objective functions cannot be globally convex
- Nevertheless, as we saw in the earlier simulations (where $p > n$), convex penalties do not necessarily outperform nonconvex in these scenarios
- For low signal-to-noise ratios there was indeed some benefit to increasing $\gamma$ in an effort to make the objective function more convex; however, for larger SNR values, this strategy diminished estimation accuracy

# Local convexity

- Although $Q(\boldsymbol{\beta})$ may not be convex over the entire $p$-dimensional parameter space (i.e., *globally convex*), it is still convex on many lower-dimensional spaces
- Some authors have advocated choosing solutions in the "locally convex" portion of the solution path (i.e., based on the minimum eigenvalues of the active features)
- Thus, local convexity of the objective function will not be an issue for large $\lambda$, but may cease to hold as $\lambda$ is lowered past some critical value $\lambda^*$

# Convexity diagnostic: Example (MCP)



# Remarks

- As the figure indicates, when $\lambda = 0.35$, $\boldsymbol{\beta}_1$ clearly minimizes the objective function, whereas at $\lambda = 0.15$, $Q(\boldsymbol{\beta}_2) < Q(\boldsymbol{\beta}_1)$
- For $\lambda \approx 0.25$, however, the objective function is very broad and flat, indicating substantial uncertainty about which solution is preferable
- Calculation of the locally convex region (the unshaded region in the earlier figure) can be a useful diagnostic in practice to indicate which regions of the solution path may suffer from multiple local minima and discontinuous paths

# Introduction

- Let us now revisit two high-dimensional studies from the previous topic and analyze them with our new reduced-bias approaches
- First, we consider an adaptive lasso model for the BRCA1 gene expression data
- As our initial estimator, let's use lasso estimates with $\lambda$ chosen according to BIC:

```
fit <- ncvreg(X, y, penalty='lasso')
b <- coef(fit, which=which.min(BIC(fit)))[-1]
```

  (using `ncvreg` for fitting due to its compatibility with `BIC`)
- Cross-validation would of course be a reasonable alternative

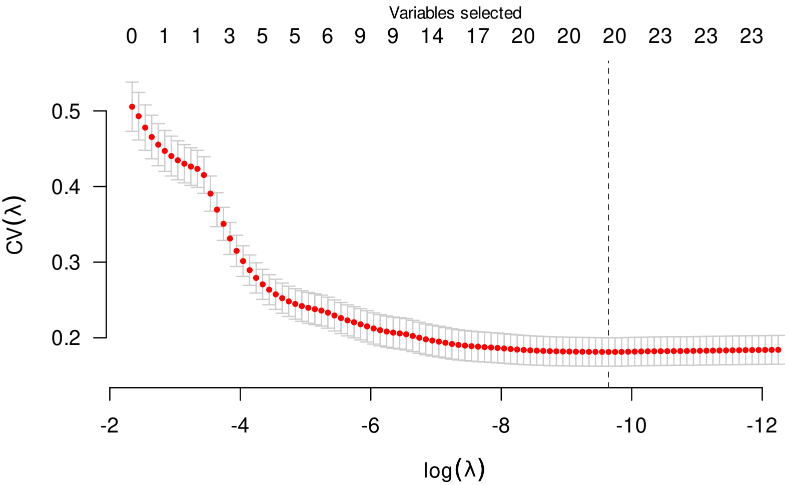
# Adaptive lasso fit

Once we have the initial estimator, it may be tempting to fit an adaptive lasso model as follows:

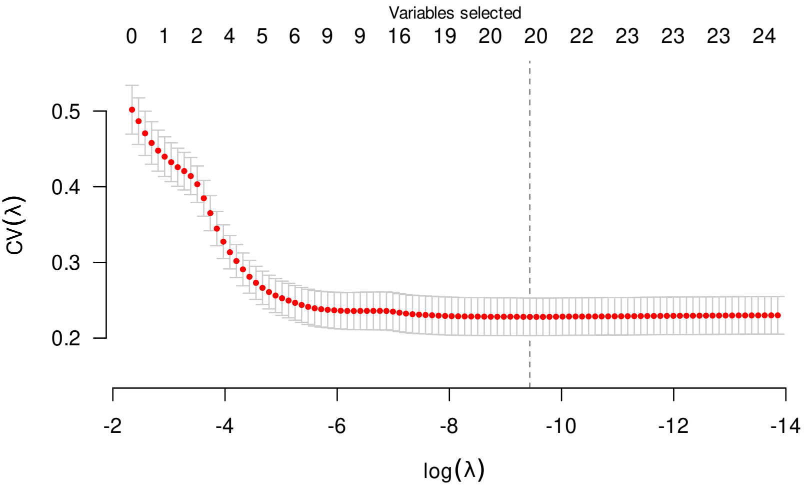
```
w <- abs(b)^(-1)   # Calculate weights
w <- pmin(w, 1e10) # cv.glmnet does not allow
                   # infinite weights
cvfit <- cv.glmnet(X, y, penalty.factor=w,
                   lambda.min=1e-5)
```

but caution is warranted

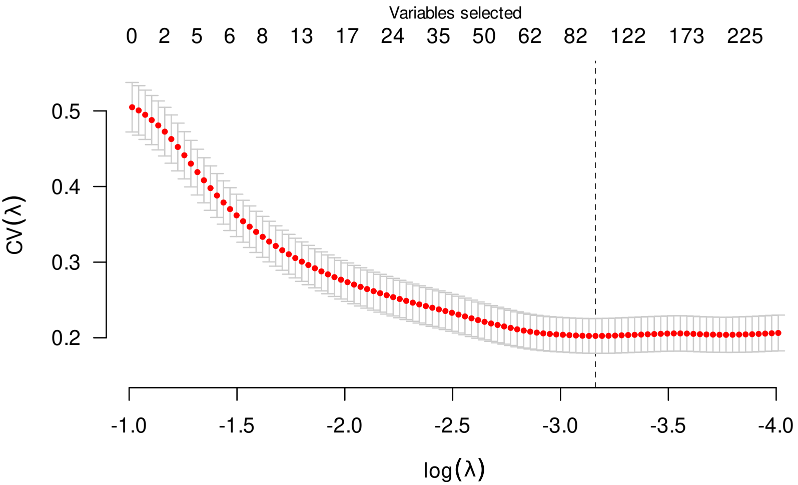
# Adaptive lasso: Cross-validation (biased)



# Adaptive lasso: Cross-validation (unbiased)



# Regular lasso: Cross-validation



# Source of bias

- In the first figure, the CV error is not estimated in an unbiased manner
- The reason is that the left-out fold is not truly external to the fitting procedure, as it was used to obtain an initial estimator
- As a result, prediction error is underestimated
- To obtain an (approximately) unbiased estimate of CV error, one must cross-validate the entire procedure, including the initial estimate

# Remarks

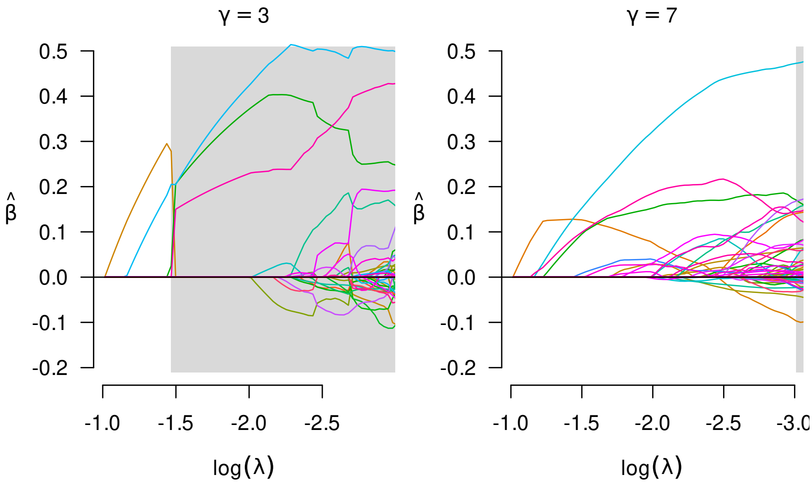
- CV errors:
  - Lasso: 0.20
  - Adaptive lasso (biased): 0.18
  - Adaptive lasso (unbiased): 0.22
- This is an important cautionary example to keep in mind for the adaptive lasso: flexible, two-stage methods have certain advantages in terms of simplicity, but are also easy to make mistakes with
- Unfortunately, while existing `R` packages can be used to fit adaptive lasso models, there are not currently any comprehensive software packages for the adaptive lasso (that I am aware of) that carry out full cross-validation

# MCP analysis

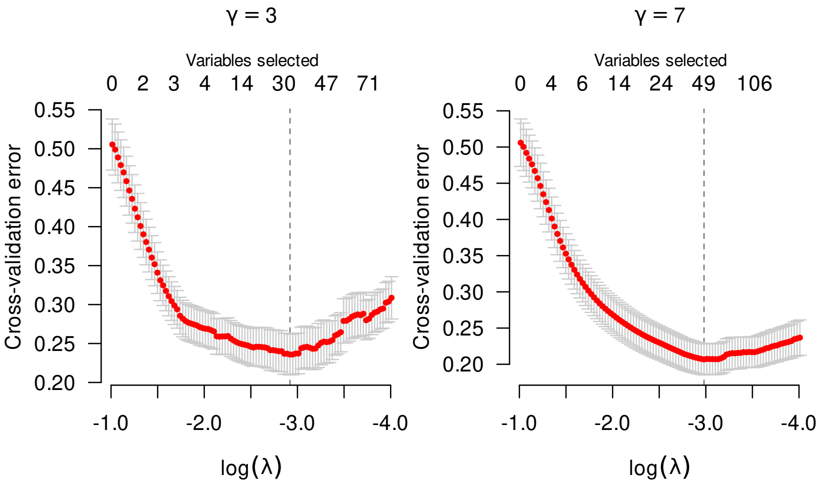
- MCP and SCAD achieve the adaptive lasso's goal of reducing the bias associated with the lasso, but do so in a single step and thus prove a bit more amenable to carrying out inference concerning predictive accuracy using cross-validation
- The `ncvreg` package is a widely used package for fitting MCP/SCAD penalized regression models; its syntax is fairly similar to `glmnet`
- Let's fit two penalized regression models to the BRCA1 data, one with $\gamma = 3$ and the other with $\gamma = 7$:

```
cvfit3 <- cv.ncvreg(X, y) # gam=3 is default
cvfit7 <- cv.ncvreg(X, y, gamma=7)
```

# Results: MCP



# CV Results: MCP



# summary ($\gamma = 3$)

ncvreg provides a useful summary function for fitted CV objects:

```
summary(cvfit3)
# MCP-penalized linear regression with n=536, p=17322
# At minimum cross-validation error (lambda=0.0539):
# -------------------------------------------------
#   Nonzero coefficients: 32
#   Cross-validation error (deviance): 0.24
#   R-squared: 0.53
#   Signal-to-noise ratio: 1.15
#   Scale estimate (sigma): 0.486
```

# summary ($\gamma = 7$)

And the equivalent summary for $\gamma = 7$:

```
summary(cvfit7)
# MCP-penalized linear regression with n=536, p=17322
# At minimum cross-validation error (lambda=0.0508):
# -------------------------------------------------
#   Nonzero coefficients: 49
#   Cross-validation error (deviance): 0.21
#   R-squared: 0.59
#   Signal-to-noise ratio: 1.45
#   Scale estimate (sigma): 0.455
```

# Remarks

- For both models, the minimum error is $\mathrm{CV} = 0.21$; very close to, although slightly larger than the $\mathrm{CV} = 0.20$ achieved by the lasso
- However, the two models select very different numbers of variables, both compared to each other and compared to the lasso, which selected 96 nonzero coefficients
- The most striking difference between the two solution paths is that for MCP with $\gamma = 3$, the the optimal solution occurs in the region that is not locally convex
- As this is real data, we cannot know which estimates are more accurate

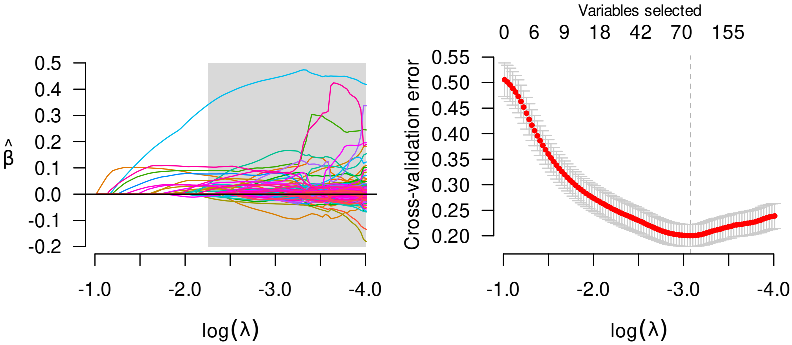
# SCAD

Finally, let us fit a SCAD-penalized regression model to this data ($\gamma = 8$):

```
cvfit_scad <- cv.ncvreg(X, y, gamma=8, penalty='SCAD')
summary(cvfit_scad)
# SCAD-penalized linear regression with n=536, p=17322
# At minimum cross-validation error (lambda=0.0464):
# -------------------------------------------------
#   Nonzero coefficients: 82
#   Cross-validation error (deviance): 0.20
#   R-squared: 0.60
#   Signal-to-noise ratio: 1.53
#   Scale estimate (sigma): 0.448
```

# Results: SCAD ($\gamma = 8$)

The SCAD results are more lasso-like than MCP is (as one would expect since the penalties are more similar)



# Remarks

- This is just one example, but these results seen are fairly representative, in my experience
- The prediction performance (as estimated by cross-validation) is typically similar between MCP/SCAD/lasso, but there can be substantial differences in terms of the estimates themselves
- The main advantage in practice of MCP (or SCAD) is the ability to achieve that prediction performance using fewer features
- Finally, the results of SCAD are almost always in between those of MCP and lasso

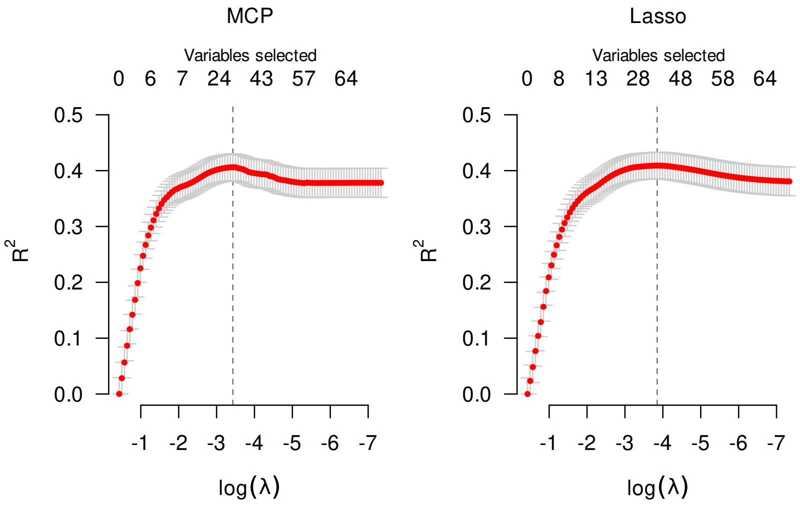
# WHO-ARI: MCP

- Let us also revisit the WHO study of acute respiratory illness, which you have looked at a few times in your homework assignments
- Let us fit an MCP-penalized regression model to this data using $\gamma = 6$ and compare it to the fit of the lasso:

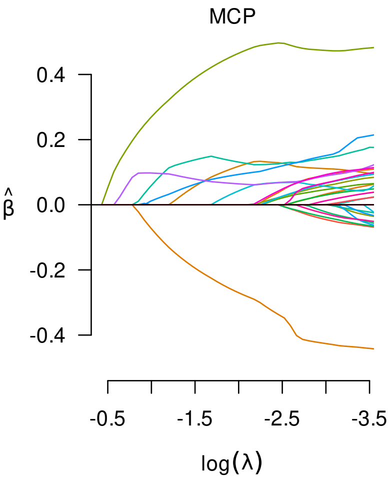
```
fold <- assign_fold(y, 10)
cvfit_mcp <- cv.ncvreg(XX, y, gam=6, fold=fold)
cvfit_las <- cv.ncvreg(XX, y, penalty="lasso",
                       fold=fold)
```

- When making these kinds of comparisons, keep the CV fold assignments the same, otherwise you risk mistaking the effect of different folds for the effect of the penalty

# Results: CV


MCP
Variables selected
0 6 7 24 43 57 64
0.5
0.4
0.3
0.2
0.1
0.0
R2
-1 -2 -3 -4 -5 -6 -7
log(λ)
Lasso
Variables selected
0 8 13 28 48 58 64
0.5
0.4
0.3
0.2
0.1
0.0
R2
-1 -2 -3 -4 -5 -6 -7
log(λ)

# Results: Coefficient path



# Summary: MCP

```
summary(cvfit_mcp)
# MCP-penalized linear regression with n=816, p=66
# At minimum cross-validation error (lambda=0.0324):
# -------------------------------------------------
#   Nonzero coefficients: 26
#   Cross-validation error (deviance): 1.21
#   R-squared: 0.41
#   Signal-to-noise ratio: 0.68
#   Scale estimate (sigma): 1.099
```

# Summary: Lasso

```
summary(cvfit_las)
# lasso-penalized linear regression with n=816, p=66
# At minimum cross-validation error (lambda=0.0213):
# -------------------------------------------------
#   Nonzero coefficients: 39
#   Cross-validation error (deviance): 1.20
#   R-squared: 0.41
#   Signal-to-noise ratio: 0.69
#   Scale estimate (sigma): 1.097
```