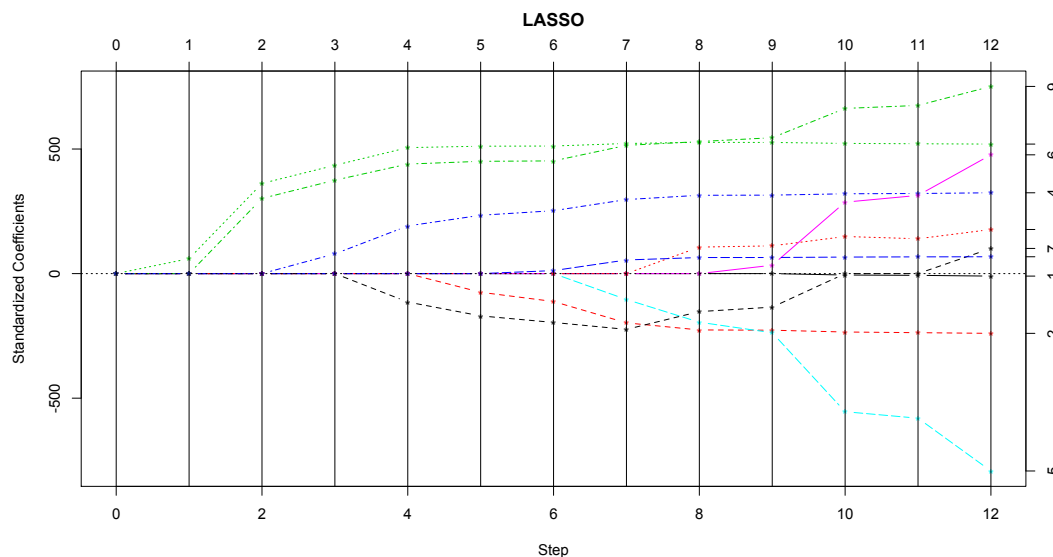


Homework 9 stepped you through the lasso modification of the LARS algorithm, based on the papers by Efron, Hastie, Johnstone, and Tibshirani (2004) (= the LARS paper) and by Rosset and Zhu (2007).

I made a few small changes to the algorithm in the LARS paper. I used the *diabetes* data set in R (the data used as an illustration in the LARS paper) to test my modification:

```
library(lars)
data(diabetes) # load the data set
LL = lars(diabetes$x,diabetes$y,type="lasso")
plot(LL,xvar="step") # one variation on the usual plot
```

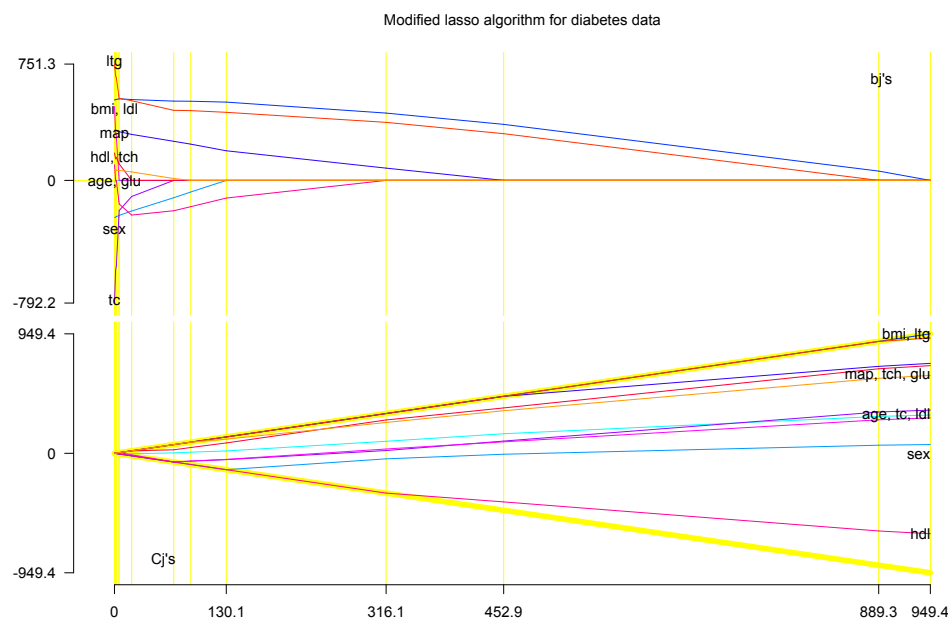


My modified algorithm *getlasso()* produces the same sequence of coefficients as the R function *lars()* when run on the *diabetes* data set:

```
J= getlasso(diabetes$y,diabetes$x)
round(LL$bet[-1,]-t(J$output["coef.end",,]),6)# gives all zeros
```

I am fairly confident that my changes lead to the same sequences of lasso fits. I find my plots, which are slightly different from those produced by the *lars* library, more helpful for understanding the algorithm:

```
show2(J) # my plot
```



[For higher resolution, see the pdf files attached to this handout.]

1 The Lasso problem

The problem is: given an $n \times 1$ vector y and an $n \times p$ matrix X , find the $\hat{b}(\lambda)$ that minimizes

$$L_\lambda(b) = \|y - Xb\|^2 + 2\lambda \sum |b_j|$$

for each $\lambda \geq 0$. [The extra factor of 2 eliminates many factors of 2 in what follows.] The columns of X will be assumed to be standardized to have zero means and $\|X_j\| = 1$. I also seem to need linear independence of various subsets of columns of X , which would be awkward if p were larger than n .

Remark. I thought the vector y was also supposed to have a zero mean. That is not the case for the *diabetes* data set in R. It does not seem to be needed for the algorithm to work.

I will consider only the “one at a time” case (page 417 of the LARS paper), for which the “active set” of predictors X_j changes only by either addition or deletion of a single predictor.

2 Directional derivative

The function L_λ has derivative at b in the direction u defined by

$$\begin{aligned} L_\lambda^\bullet(b, u) &= \lim_{t \downarrow 0} \frac{L_\lambda(b + tu) - L_\lambda(b)}{t} \\ &= 2 \sum_j \lambda D(b_j, u_j) - (y - Xb)' X_j u_j \\ &\quad \text{where } D(b_j, u_j) = u_j \{b_j > 0\} - u_j \{b_j < 0\} + |u_j| \{b_j = 0\}. \end{aligned} \tag{<1>}$$

By convexity, a vector b minimizes L_λ if and only if $L_\lambda^\bullet(b, u) \geq 0$ for every u . Equivalently, for every j and every u_j the j th summand in <1> must be non-negative. [Consider u vectors with only one nonzero component to establish this equivalence.] That is, b minimizes L_λ if and only if

$$\lambda D(b_j, u_j) \geq (y - Xb)' X_j u_j \quad \text{for every } j \text{ and every } u_j$$

When $b_j \neq 0$ the inequalities for $u_j = \pm 1$ imply an equality; for $b_j = 0$ we get only the inequality. Thus b minimizes L_λ if and only if

$$\begin{cases} \lambda = X_j' R & \text{if } b_j > 0 \\ \lambda = -X_j' R & \text{if } b_j < 0 \\ \lambda \geq |X_j' R| & \text{if } b_j = 0 \end{cases} \quad \text{where } R := y - Xb$$

The LARS/lasso algorithm recursively calculates a sequence of break-points $\infty > \lambda_1 > \lambda_2 > \dots \geq 0$ with $\hat{b}(\lambda)$ linear for each interval $\lambda_{k+1} \leq \lambda \leq \lambda_k$. Define “residual” vector and “correlations”

$$R(\lambda) := y - X\hat{b}(\lambda) \quad \text{and} \quad C_j(\lambda) := X_j' R(\lambda).$$

Remark. To get a true correlation we would have to divide by $\|R(\lambda)\|$, which would complicate the constraints.

The algorithm will ensure that

$$\begin{cases} \lambda = C_j(\lambda) & \text{if } \hat{b}_j(\lambda) > 0 & \text{(constraint } \oplus) \\ \lambda = -C_j(\lambda) & \text{if } \hat{b}_j(\lambda) < 0 & \text{(constraint } \ominus) \\ \lambda \geq |C_j(\lambda)| & \text{if } \hat{b}_j(\lambda) = 0 & \text{(constraint } \odot) \end{cases} \tag{<2>}$$

That is, for the minimizing $\hat{b}(\lambda)$ each $(\lambda, C_j(\lambda))$ needs to stay inside the region $\mathcal{R} = \{(\lambda, c) \in \mathbb{R}_+ \times \mathbb{R} : |c| \leq \lambda\}$, moving along the top boundary ($c = \lambda$) when $\hat{b}_j(\lambda) > 0$ (constraint \oplus), along the lower boundary ($c = -\lambda$) when $\hat{b}_j(\lambda) < 0$ (constraint \ominus), and being anywhere in \mathcal{R} when $\hat{b}_j(\lambda) = 0$ (constraint \odot).

3 The algorithm

The solution $\hat{b}(\lambda)$ is continuous in λ and linear on intervals defined by change points $\infty = \lambda_0 > \lambda_1 > \lambda_2 > \dots > 0$. The construction proceeds in steps, starting with large λ and working towards $\lambda = 0$. The vector of fitted values $\hat{f}(\lambda) = X\hat{b}(\lambda)$ is also piecewise linear. Within each interval $(\lambda_{k+1}, \lambda_k)$ only the “active subset” $A = A_k = \{j : \hat{b}_j(\lambda) \neq 0\}$ of the coefficients changes; the inactive coefficients stay fixed at zero.

3.1 Some illuminating special cases

It helped me to work explicitly through the first few steps before thinking about the equations that define a general step in the algorithm.

Start with $A_0 = \emptyset$ and $\hat{b}(\lambda) = 0$ for $\lambda \geq \lambda_1 := \max |X'_j y|$. Constraint \odot is satisfied on $[\lambda_1, \infty)$.

Step 1.

Constraint \odot would be violated if we kept $\hat{b}(\lambda)$ equal to zero for $\lambda < \lambda_1$, because we would have $\max_j |C_j(\lambda)| > \lambda$. The $\hat{b}(\lambda)$ must move away from zero as λ decreases below λ_1 .

We must have $|C_j(\lambda_1)| = \lambda_1$ for at least one j . For convenience of exposition, suppose $C_1(\lambda_1) = \lambda_1 > |C_j(\lambda_1)|$ for all $j \geq 2$. The active set now becomes $A = \{1\}$.

For $\lambda_2 \leq \lambda < \lambda_1$, with λ_2 to be specified soon, keep $\hat{b}_j(\lambda) = 0$ for $j \geq 2$ but let

$$\hat{b}_1(\lambda) = 0 + v_1(\lambda_1 - \lambda)$$

for some constant v_1 . To maintain the equalities

$$\begin{aligned} \lambda &= C_1(\lambda) = X'_1(y - X_1\hat{b}_1(\lambda)) \\ &= C_1(\lambda_1) - X'_1X_1v_1(\lambda_1 - \lambda) = \lambda_1 - v_1(\lambda_1 - \lambda) \end{aligned}$$

we need $v_1 = 1$. This choice also ensures that $\hat{b}_1(\lambda) > 0$ for a while, so that \oplus is the relevant constraint for \hat{b}_1 .

For $\lambda < \lambda_1$, with $v_1 = 1$ we have $R(\lambda) = y - X_1(\lambda_1 - \lambda)$ and

$$C_j(\lambda) = C_j(\lambda_1) - a_j(\lambda_1 - \lambda) \quad \text{where } a_j := X'_jX_1.$$

Notice that $|a_j| < 1$ unless $X_j = \pm X_1$. Also, as long as $\max_{j \geq 2} |C_j(\lambda)| \leq \lambda$ the other \hat{b}_j 's still satisfy constraint \odot .

We need to end the first step at λ_2 , the largest λ less than λ_1 for which $\max_{j \geq 2} |C_j(\lambda)| = \lambda$. Solve for $C_j(\lambda) = \pm\lambda$ for each fixed $j \geq 2$:

$$\begin{aligned}\lambda &= \lambda_1 - (\lambda_1 - \lambda) = C_j(\lambda_1) - a_j(\lambda_1 - \lambda) \\ -\lambda &= -\lambda_1 + (\lambda_1 - \lambda) = C_j(\lambda_1) - a_j(\lambda_1 - \lambda)\end{aligned}$$

if and only if

$$\begin{aligned}\lambda_1 - \lambda &= (\lambda_1 - C_j(\lambda_1)) / (1 - a_j) \\ \lambda_1 - \lambda &= (\lambda_1 + C_j(\lambda_1)) / (1 + a_j)\end{aligned}$$

Both right-hand sides are strictly positive. Thus $\lambda_2 = \lambda_1 - \Delta\lambda$ where

$$<3> \quad \Delta\lambda := \min_{j \geq 2} \min \left(\frac{\lambda_1 - C_j(\lambda_1)}{1 - a_j}, \frac{\lambda_1 + C_j(\lambda_1)}{1 + a_j} \right)$$

Second step.

We have $C_1(\lambda_2) = \lambda_2 = \max_{j \geq 2} |C_j(\lambda_2)|$, by construction. For convenience of exposition, suppose $|C_2(\lambda_2)| = \lambda_2 > |C_j(\lambda_2)|$ for all $j \geq 3$. The active set now becomes $A = \{1, 2\}$.

To emphasize a subtle point it helps to consider separately two cases. Write s_2 for $\text{sign}(C_2(\lambda_2))$, so that $C_2(\lambda_2) = s_2\lambda_2$.

case $s_2 = +1$:

For $\lambda_3 \leq \lambda < \lambda_2$ and a new v_1 and v_2 (*Note the recycling of notation.*), define

$$\begin{aligned}\hat{b}_1(\lambda) &= \hat{b}_1(\lambda_2) + (\lambda_2 - \lambda)v_1 \\ \hat{b}_2(\lambda) &= 0 + (\lambda_2 - \lambda)v_2\end{aligned}$$

with all other \hat{b}_j 's still zero. Write Z for $[X_1, X_2]$. The new C_j 's become

$$\begin{aligned}C_j(\lambda) &= X_j' \left(y - X_1 \hat{b}_1(\lambda) - X_2 \hat{b}_2(\lambda) \right) \\ &= C_j(\lambda_2) - (\lambda_2 - \lambda) X_j' Z v \quad \text{where } v' = (v_1, v_2).\end{aligned}$$

We keep $C_1(\lambda) = C_2(\lambda) = \lambda$ if we choose v to make $X_1' Z v = 1 = X_2' v$. That is, we need

$$v = (Z'Z)^{-1} \mathbf{1} \quad \text{with } \mathbf{1} = (1, 1)'.$$

Of course we must assume that X_1 and X_2 are linearly independent for $Z'Z$ to have an inverse.

case $s_2 = -1$:

For $\lambda_3 \leq \lambda < \lambda_2$ and a new v_1 and v_2 , define

$$\begin{aligned}\hat{b}_1(\lambda) &= \hat{b}_1(\lambda_2) + (\lambda_2 - \lambda)v_1 \\ \hat{b}_2(\lambda) &= 0 - (\lambda_2 - \lambda)v_2 \quad (\text{note the change of sign})\end{aligned}$$

with all other \hat{b}_j 's still zero. Write Z for $[X_1, -X_2]$. The tricky business with the signs ensures that

$$X\hat{b}(\lambda) - X\hat{b}(\lambda_2) = X_1(\lambda_2 - \lambda)v_1 - X_2(\lambda_2 - \lambda)v_2 = (\lambda_2 - \lambda)Zv.$$

The new C_j 's become

$$C_j(\lambda) = X_j' \left(y - X_1\hat{b}_1(\lambda) - X_2\hat{b}_2(\lambda) \right) = C_j(\lambda_2) - (\lambda_2 - \lambda)X_j'Zv.$$

We keep $C_1(\lambda) = -C_2(\lambda) = \lambda$ if we choose v to make $X_1'Zv = 1 = -X_2'v$. That is, again we need $v = (Z'Z)^{-1}\mathbf{1}$.

Remark. If we had assumed $C_1(\lambda_1) = -\lambda_1$ then X_1 would be replaced by $-X_1$ in the Z matrix, that is, $Z = [s_1X_1, s_2X_2]$ with $s_1 = \text{sign}(C_1(\lambda_1))$ and $s_2 = \text{sign}(C_2(\lambda_2))$.

Because $\hat{b}_1(\lambda_2) > 0$, the correlation $C_1(\lambda)$ stays on the correct boundary for the \oplus constraint. If $s_2 = +1$ we need $v_2 > 0$ to keep $\hat{b}_2(\lambda) > 0$ and $C_2(\lambda) = \lambda$, satisfying \oplus . If $s_2 = -1$ we also need $v_2 > 0$ to keep $\hat{b}_2(\lambda) < 0$ and $C_2(\lambda) = -\lambda$, satisfying \ominus . That is, in both cases we need $v_2 > 0$.

Why do we get a strictly positive v_2 ? Write ρ for $s_2X_2'X_1$. As the $Z'Z$ matrix is nonsingular we must have $|\rho| < 1$ so that

$$v = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}^{-1} \mathbf{1} = (1 - \rho^2)^{-1} \begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix} \mathbf{1}$$

and $v_2 = v_1 = (1 - \rho)/(1 - \rho^2) > 0$.

If no further $C_j(\lambda)$'s were to hit the $\pm\lambda$ boundary, step 2 could continue all the way to $\lambda = 0$. More typically, we would need to create a new active set at the largest λ_3 strictly smaller than λ_2 for which $\max_{j \geq 3} |C_j(\lambda)| = \lambda$.

For the general step there is another possible event that would require a change to the active set: one of the $\hat{b}_j(\lambda)$'s in the active set might hit zero, threatening to change sign and leave the corresponding $C_j(\lambda)$ on the wrong boundary.

I could pursue these special cases further, but it is better to start again for the generic step in the algorithm.

3.2 The general algorithm

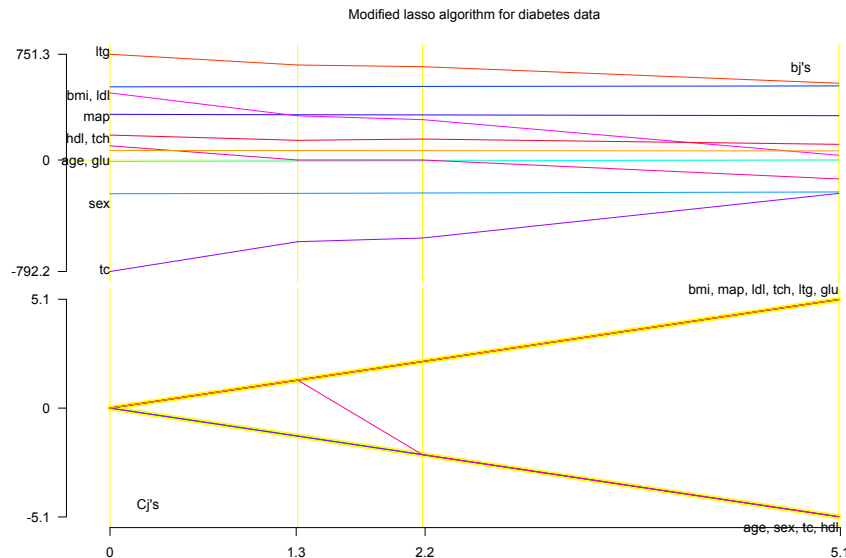
Once again, start with $A_0 = \emptyset$ and $\hat{b}(\lambda) = 0$ for $\lambda \geq \lambda_1 := \max |X'_j y|$. Constraint \odot is satisfied on $[\lambda_1, \infty)$.

At each λ_k a new active set A_k is defined. During the k th step the parameter λ decreases from λ_k to λ_{k+1} . For all j 's in the active set A_k , the coefficients $\hat{b}_j(\lambda)$ change linearly and the $C_j(\lambda)$'s move along one of the boundaries of the feasible region: $C_j(\lambda) = \lambda$ if $\hat{b}_j(\lambda) > 0$ and $C_j(\lambda) = -\lambda$ if $\hat{b}_j(\lambda) < 0$. For each inactive j the coefficient $\hat{b}_j(\lambda)$ remains zero throughout $[\lambda_{k+1}, \lambda_k]$.

Step k ends when either an inactive $C_j(\lambda)$ hits a $\pm\lambda$ boundary or if an active $\hat{b}_j(\lambda)$ becomes zero: λ_{k+1} is defined as the largest λ less than λ_k for which either of these conditions holds:

- (i) $\max_{j \notin A_k} |C_j(\lambda)| = \lambda$. In that case add the new $j \in A_k^c$ for which $|C_j(\lambda_{k+1})| = \lambda_{k+1}$ to the active set, then proceed to step $k + 1$.
- (ii) $\hat{b}_j(\lambda) = 0$ for some $j \in A_k$. In that case, remove j from the active set, then proceed to step $k + 1$.

For the *diabetes* data, the alternative (ii) caused the behavior shown below for $1.3 \leq \lambda \leq 2.2$.



I will show how the $C_j(\lambda)$'s and the $\hat{b}_j(\lambda)$'s can be chosen so that the conditions $\langle 2 \rangle$ are always satisfied.

At the start of step k (with $\lambda = \lambda_k$), define $Z = [s_j X_j : j \in A_k]$, where $s_j := \text{sign}(C_j(\lambda))$. That is, Z has a column for each active X_j predictor, with the sign flipped if the corresponding $C_j(\lambda)$ is moving along the $-\lambda$ boundary. Define $v := (Z'Z)^{-1}\mathbf{1}$. For $\lambda_{k+1} \leq \lambda \leq \lambda_k$ the active coefficients change linearly,

$$<4> \quad \hat{b}_j(\lambda) = \hat{b}_j(\lambda_k) + (\lambda_k - \lambda)v_j s_j \quad \text{for } j \in A_k.$$

The “fitted vector”,

$$\hat{f}(\lambda) := X\hat{b}(\lambda) = \hat{f}(\lambda_k) + \sum_{j \in A_k} X_j \left(\hat{b}_j(\lambda) - \hat{b}_j(\lambda_k) \right) = \hat{f}(\lambda_k) + (\lambda_k - \lambda)Zv,$$

the “residuals”,

$$R(\lambda) := y - \hat{f}(\lambda) = R(\lambda_k) - (\lambda_k - \lambda)Zv,$$

and the “correlations”,

$$<5> \quad C_j(\lambda) := X_j' R(\lambda) = C_j(\lambda_k) - (\lambda_k - \lambda)a_j \quad \text{where } a_j := X_j' Zv,$$

also change linearly. In particular, for $j \in A_k$,

$$C_j(\lambda) = s_j \lambda_k - (\lambda_k - \lambda)s_j Z_j' Zv = s_j \lambda \quad \text{because } Z_j' Z(Z'Z)^{-1}\mathbf{1} = 1.$$

The active $C_j(\lambda)$ ’s move along one of the boundaries $\pm\lambda$ of \mathcal{R} .

Remember that I am assuming the “one at a time” condition: the active set changes only by addition of one new predictor in case (i) or by dropping one predictor in case (ii).

Suppose index α enters the active set via (i) when λ equals λ_{k+1} then leaves it via (ii) when λ equals $\lambda_{\ell+1}$. (If α never leaves the active set put $\lambda_{\ell+1}$ equal to 0.) Throughout the interval $(\lambda_{\ell+1}, \lambda_{k+1})$ both $\text{sign}(C_\alpha(\lambda))$ and $\text{sign}(\hat{b}_\alpha(\lambda))$ stay constant. To ensure that all constraints are satisfied it is enough to prove:

- (a) For λ only slightly smaller than λ_{k+1} , both $C_\alpha(\lambda)$ and $\hat{b}_\alpha(\lambda)$ have the same sign.
- (b) For λ only slightly smaller than $\lambda_{\ell+1}$, constraint \odot is satisfied, that is, $|C_\alpha(\lambda)| \leq \lambda$.

The analyses are similar for the two cases. They both depend on a neat formula for inversion of symmetric block matrices. Suppose A is an $m \times m$ nonsingular, symmetric matrix and d is an $m \times 1$ vector for which $\kappa := 1 - d'A^{-1}d \neq 0$. Then

$$<6> \quad \begin{pmatrix} A & d \\ d' & 1 \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + ww'/\kappa & | & -w/\kappa \\ -w'/\kappa & | & 1/\kappa \end{pmatrix} \quad \text{where } w := A^{-1}d .$$

This formula captures the ideas involved in Lemma 4 of the LARS paper.

For simplicity of notation, I will act in both cases (a) and (b) as if the α is larger than all the other j 's that were in the active set. (More formally, I could replace X in what follows by XE for a suitable permutation matrix E .) The old and new Z matrices then have the block form shown in $<6>$.

Case (a): α enters the active set at λ_{k+1}

As for the analysis that led to the expression in $<3>$, the solutions for $\lambda = |C_j(\lambda)|$ with $j \in A_k^c$ are given by

$$\begin{aligned} (\lambda_k - \lambda)(1 - a_j) &= \lambda_k - C_j(\lambda_k) && \text{to get } \lambda = C_j(\lambda) \\ (\lambda_k - \lambda)(1 + a_j) &= \lambda_k + C_j(\lambda_k) && \text{to get } -\lambda = C_j(\lambda) \end{aligned}$$

Index α is brought into the active set because $|C_\alpha(\lambda_{k+1})| = \lambda_{k+1}$. Thus

$$(\lambda_k - \lambda'_{k+1})(1 - s_\alpha a_\alpha) = \lambda_k - s_\alpha C_\alpha(\lambda_k) \quad \text{where } s_\alpha := \text{sign}(C_\alpha(\lambda'_{k+1})).$$

Note that $|C_\alpha(\lambda_k)| < \lambda_k$ because α was not active during step k . It follows that the right-hand side of the last equality is strictly positive, which implies

$$<7> \quad 1 - s_\alpha a_\alpha > 0.$$

Throughout a small neighborhood of λ_{k+1} the sign s_α of $C_\alpha(\lambda)$ stays the same. Continue to write Z for the active matrix $[s_j X_j; j \in A_k]$ for λ slightly larger than λ_{k+1} and denote by

$$\tilde{Z} = [s_j X_j; j \in A_{k+1}] = [Z, s_\alpha X_\alpha]$$

the new active matrix for λ slightly small than λ_{k+1} . Then

$$\tilde{Z}'\tilde{Z} = \begin{pmatrix} Z'Z & | & d \\ d' & | & 1 \end{pmatrix} \quad \text{where } d := s_\alpha Z'X_\alpha .$$

Notice that

$$1 - \kappa = d' A^{-1} d = X'_\alpha Z (Z' Z)^{-1} Z' X_\alpha = \|H X_\alpha\|^2,$$

where $H = Z(Z'Z)^{-1}Z'$ is the matrix that projects vectors orthogonally onto $\text{span}(Z)$. If $X_\alpha \notin \text{span}(Z)$ then $\|H X_\alpha\| < \|X_\alpha\| = 1$ so that $\kappa > 0$. From <6>,

$$<8> \quad (\tilde{Z}' \tilde{Z})^{-1} = \left(\begin{array}{c|c} (Z'Z)^{-1} + ww'/\kappa & -w/\kappa \\ \hline -w'/\kappa & 1/\kappa \end{array} \right) \quad \text{where } w := s_\alpha (Z'Z)^{-1} Z' X_\alpha.$$

The α th coordinate of the new $\tilde{v} = (\tilde{Z}' \tilde{Z})^{-1} \mathbf{1}$ equals $\tilde{v}_\alpha = \kappa^{-1} (1 - w' \mathbf{1})$, which is strictly positive because

$$\begin{aligned} 1 - w' \mathbf{1} &= 1 - s_\alpha X'_\alpha Z (Z'Z)^{-1} \mathbf{1} \\ &= 1 - s_\alpha X'_\alpha \tilde{Z} v \\ &= 1 - s_\alpha a_\alpha \quad \text{by <5>} \\ &> 0 \quad \text{by <7>}. \end{aligned}$$

By <4>, the new $\hat{b}_\alpha(\lambda) = (\lambda_{k+1} - \lambda) \tilde{v}_\alpha s_\alpha$ has the same sign, s_α , as $C_\alpha(\lambda)$.

Case (b): α leaves the active set at $\lambda = \lambda_{\ell+1}$

The roles of $Z = [s_j X_j : 1 \leq j < \alpha]$ and $\tilde{Z} = [Z, s_\alpha X_\alpha]$ are now reversed. For λ slightly larger than $\lambda_{\ell+1}$ the active matrix is \tilde{Z} , and both $C_\alpha(\lambda)$ and $\hat{b}_\alpha(\lambda)$ have sign s_α .

Index α leaves the active set because $\hat{b}_\alpha(\lambda_{\ell+1}) = 0$. Thus

$$0 = \hat{b}_\alpha(\lambda_\ell) + (\lambda_\ell - \lambda_{\ell+1}) \tilde{v}_\alpha s_\alpha$$

where $s_\alpha := \text{sign}(C_\alpha(\lambda_\ell))$ and $\tilde{v} = (\tilde{Z}' \tilde{Z})^{-1} \mathbf{1}$. The active $\hat{b}_\alpha(\lambda_\ell)$ also had sign s_α . Consequently, we must have

$$<9> \quad \tilde{v}_\alpha < 0.$$

For λ slightly smaller than $\lambda_{\ell+1}$ the active matrix is Z , and, by <5>,

$$\begin{aligned} s_\alpha C_\alpha(\lambda) &= \lambda_{\ell+1} - (\lambda_{\ell+1} - \lambda) s_\alpha a_\alpha \quad \text{where } a_\alpha := X'_\alpha Z v \\ &= \lambda + (\lambda_{\ell+1} - \lambda) (1 - s_\alpha a_\alpha) \end{aligned}$$

We also know that

$$0 > \kappa \tilde{v}_\alpha = 1 - w' \mathbf{1} = 1 - s_\alpha a_\alpha.$$

Thus $s_\alpha C_\alpha(\lambda) < \lambda$, and hence $|C_\alpha(\lambda)| < \lambda$, for λ slightly less than $\lambda_{\ell+1}$. The new $C_\alpha(\lambda)$ satisfies constraint \oplus as it heads off towards the other boundary.

Remark. Clearly there is some sort of duality accounting for the similarities in the arguments for cases (a) and (b), with $\langle 6 \rangle$ as the shared mechanism. If we think of λ as time, case (b) is a time reversal of case (a). For case (a) the defining condition (C_α hits the boundary) at λ_{k+1} gives $1 - s_\alpha v_\alpha > 0$, which implies $v_\alpha > 0$. For case (b), the defining condition (\hat{b}_α hits zero) at $\lambda_{\ell+1}$ gives $\tilde{v}_\alpha < 0$, which implies $1 - s_\alpha a_\alpha < 0$.

Is there some clever way to handle both cases by a duality argument? Maybe I should read more of the LARS paper.

4 My getlasso() function

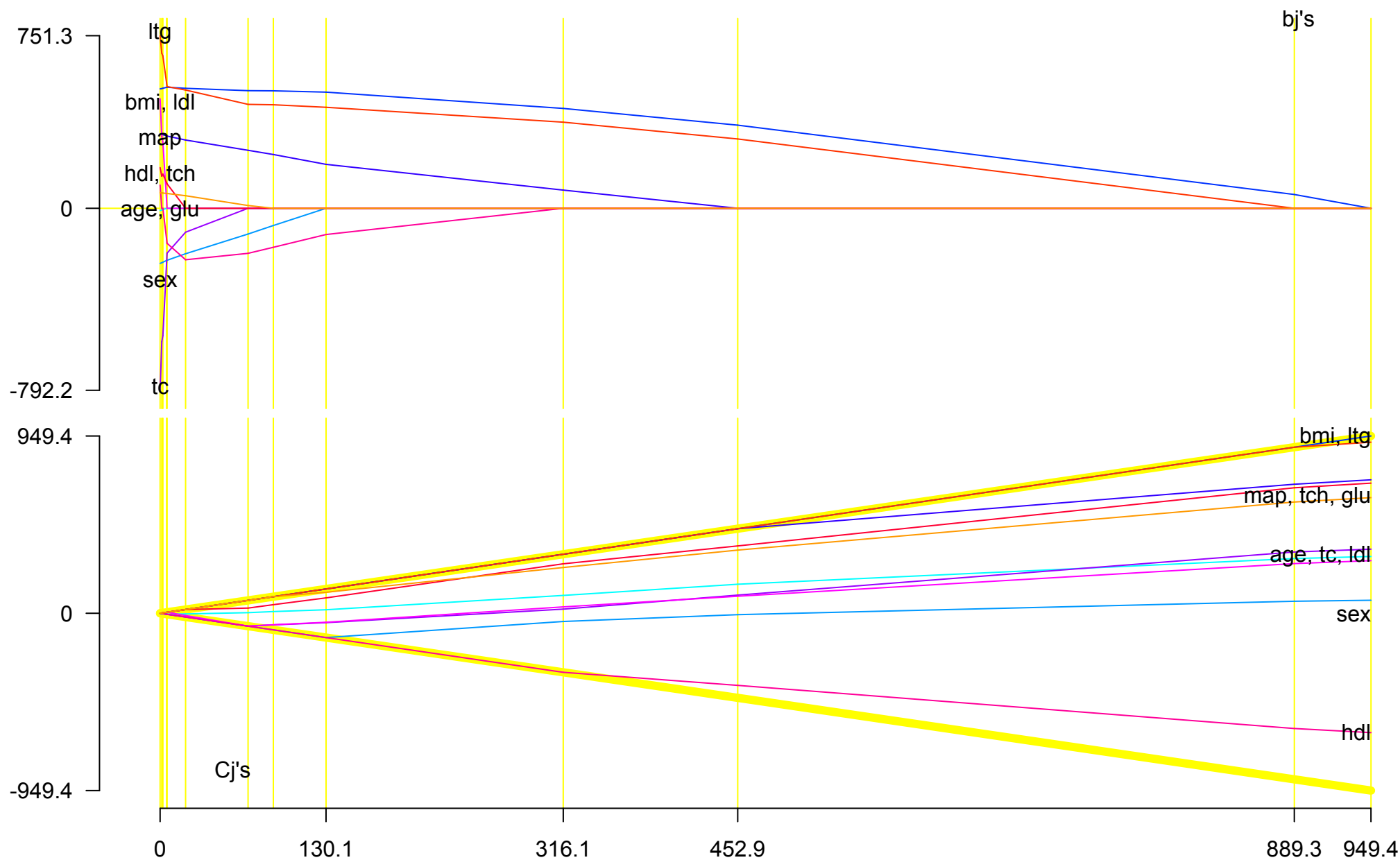
I wrote the R code in the file `lasso.R` to help me understand the algorithm described in the LARS paper. The function is not particularly elegant or efficient. I wrote in a way that made it easy to examine the output from each step. If `verbose=T`, lots of messages get written to the console and the function pauses after each step. I also used some calls to `browser()` while tracking down some annoying bugs related to case (b).

d.p. 1 December 2010

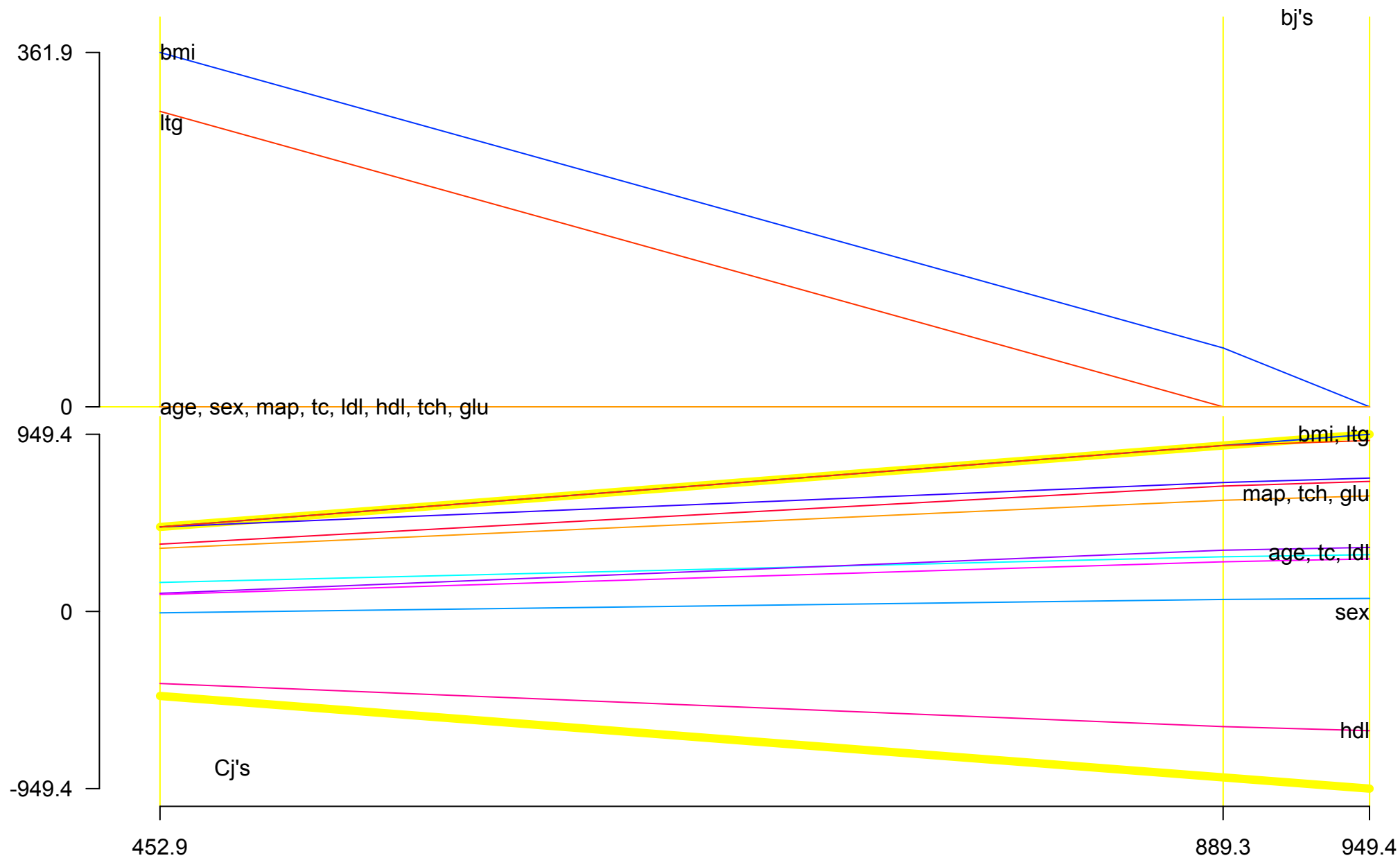
References

- Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani (2004). Least angle regression. *The Annals of Statistics* 32(2), pp. 407–451.
- Rosset, S. and J. Zhu (2007). Piecewise linear regularized solution paths. *Annals of Statistics* 35(3), 1012–1030.

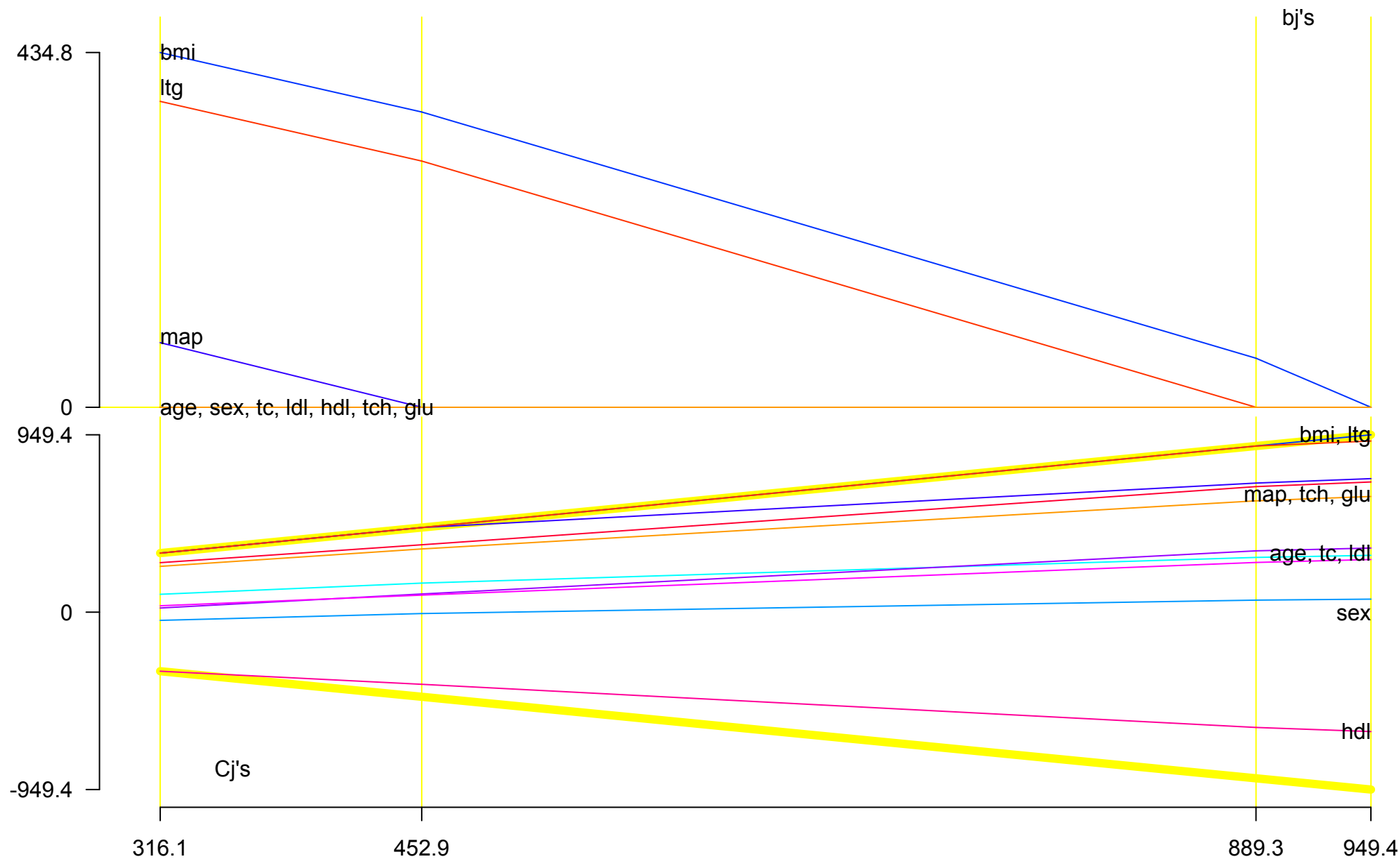
Modified lasso algorithm for diabetes data



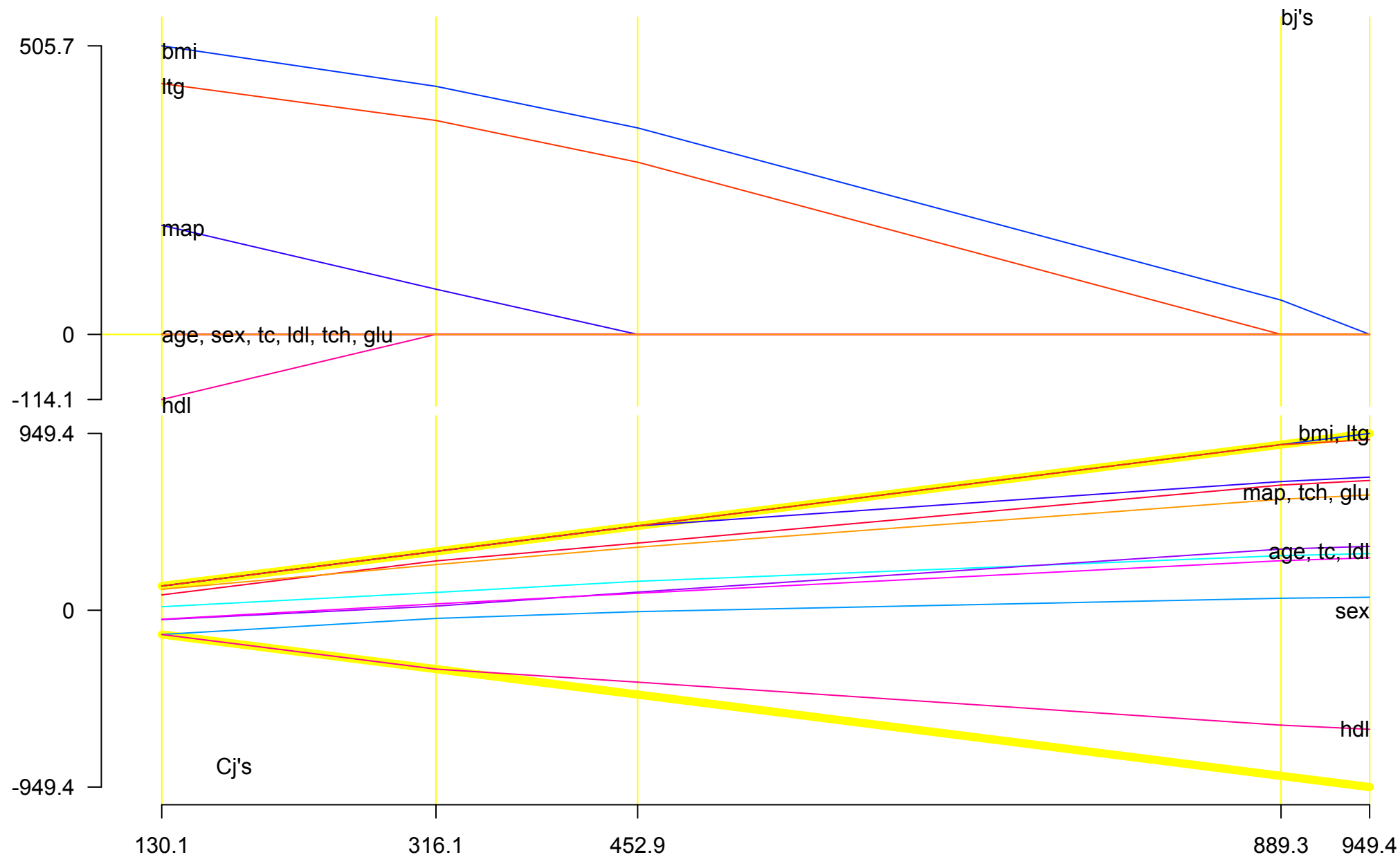
Modified lasso algorithm for diabetes data



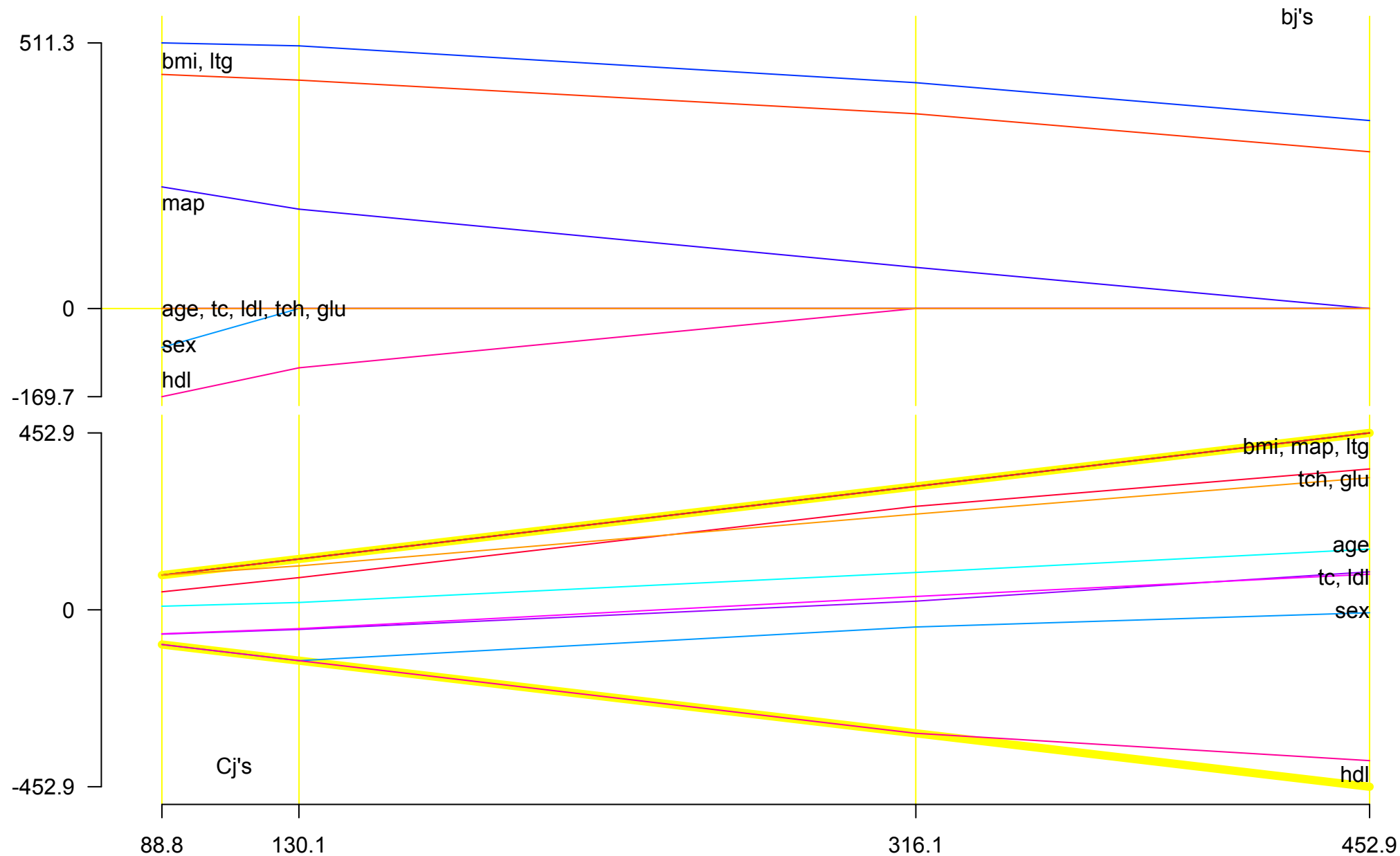
Modified lasso algorithm for diabetes data



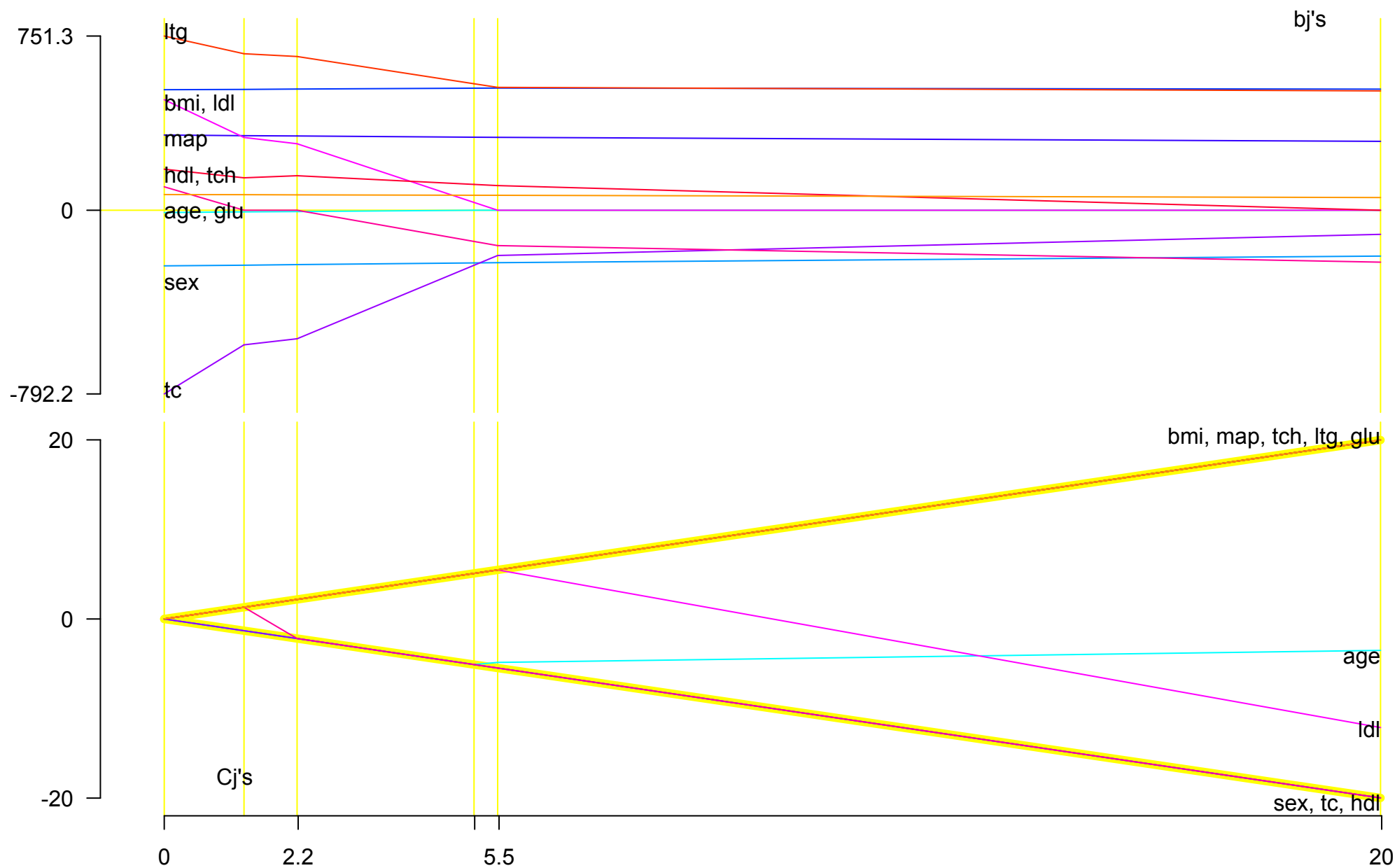
Modified lasso algorithm for diabetes data



Modified lasso algorithm for diabetes data



Modified lasso algorithm for diabetes data



Modified lasso algorithm for diabetes data

