A tutorial/walkthrough of basic lasso and ridge model selection techniques

This is a stylized version of a lasso and ridge commands in Stata. The best way to follow along is to download the accompanying "lasso+ridge.do" script which includes on the Stata commands from this file. Be sure to change the file paths to match those of your computer. The PDF of the this document with output is also available in this folder

Introduction

From what little I know about machine learning (ML) is that Stata has been somewhat slow to the game. Version 16, released less than a year ago in the summer of 2019, finally included an in-house command **lasso** that performs many of the functions that were previously found only in community-written packages. I learned using one of these packages called lassopack – which includes the command **lasso2**. This packages is available on SSC and we'll use that package here to demonstrate lasso and ridge algorithms. I don't have much experience with the in-house command built into version 16 and the few times I have used it, I get results that I don't expect. When I have time I'll include a separate script that directly compares the results from **lasso2** and **lasso**. In short, I'm skeptical of **lasso** because the command doesn't produce results that I know I should be getting, but I'm very very open to the possibility that I'm not doing something right. If that's the case, please comment and/or show me how to actually use **lasso** properly.

Getting Started

Anyway, The first thing we need to do is download lassopack from the SSC. We could do so by running the install command from the ssc, which would look like this

```
. ssc install pdslasso
```

However, the authors of this particular package (see https://statalasso.github.io/) note that they update their GitHub-hosted version more frequently than that on the SSC. We'll install directly from GitHub instead and use

```
. net install lassopack, from("https://raw.githubusercontent.com/statalasso/lass
> opack/master/lassopack_v131/") replace
```

We'll also be using the estout package to store results from the lasso commands. You probably know about estout but, if not, I highly recommend familiarizing yourself with it as it will become more and more useful to you over time.

```
. ssc install estout, replace
```

Whereas traditional econometrics cares about making sense of relationships between people, actions, and things in the world, the machine-learning approach doesn't really care about any of that. Instead, machine-learning classifies, predicts, and seeks to build the best model independent of theoretical or historical context. Take Google; they just want to know "yes" or "no". Will you click on the ad or not? They've likely got hundreds of variables on "you" and they're pretty good at predicting the answer to that question, for each of us. The World Bank or J-PAL, residing comfortably in the classical econometrical tradition, care much more about making sure the model makes sense in context: does climate change incite violence? Interestingly, from an econometrics perspective, this question becomes very difficult to answer in the affirmative (even though we all know the meteorological and biological evidence of climate change is immense) since very few researchers have told a convincing story that isolates climate change-induced weather variability from other economic or political factors which also affect intermediate factors – such as food or water insecurity – that may incite violence. (A fantastic paper by Selby et al that examines climate change and the Syrian Revolution is linked here: https://www.sciencedirect.com/science/article/pii/S0962629816301822) Econometrics cares about "why" and machine-learning cares about what the best model looks like; in a way the former is much harder to do well, in my opinion. But it's also pretty easy to have machine-learning go horribly wrong. The rest of this tutorial will hopefully set us up with some basic conceptual and coding understanding to, at least, not totally mess up machine-learning.

Traditional Approach

Now we need a dataset. Let's first use one of Stata's built-in datasets to demonstrate an instance where ML won't be as useful.

```
. sysuse auto, clear
```

Now let's say we want to predict the car's price. An econometric approach might ask what factors predict price (in miles/gallon) by examining the context of the US auto market in the late 1970s. We might surmise that foreign cars, subject to import tariffs, are likely to be more expensive; also, cars that weigh more might indicate less efficiency. We could run lasso or ridge here, but using our rough "theory of change" we can tell a pretty convincing story already: foreign and weight are two strong predictors using OLS.

```
. reg price       foreign weight

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     35.35
       Model |  316859273         2   158429637   Prob > F        =    0.0000
    Residual |  318206123        71  4481776.38   R-squared       =    0.4989
-------------+----------------------------------   Adj R-squared   =    0.4848
       Total |  635065396        73  8699525.97   Root MSE        =      2117

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     foreign |   3637.001    668.583     5.44   0.000     2303.885    4970.118
      weight |   3.320737   .3958784     8.39   0.000     2.531378    4.110096
       _cons |  -4942.844   1345.591    -3.67   0.000    -7625.876   -2259.812
------------------------------------------------------------------------------
```

Lasso

Great, but what happens if we have a dataset about which we know very little? Let's take a look at this one about wine quality from the UCI machine learning repository. When you find a dataset you should always keep in mind what the owners allow you to do with it – here we're ok to use it for research purposes as long as we cite it, so we will: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

```
. import delimited using "http://archive.ics.uci.edu/ml/machine-learning-databas
> es/wine-quality/winequality-red.csv" , clear
(12 vars, 1,599 obs)
```

Great – now let's describe dataset:

```
. describe, short

Contains data
  obs:         1,599
 vars:            12
Sorted by:
     Note: Dataset has changed since last saved.
```

Looks like each observation is a wine sample, and we have 12 variables, including **quality** that tells us how "good" the wine is. Let's see if we can predict quality from the other variables at our disposal. I know nothing about the chemistry of wine, so I can't use my economic background to help me produce any sort of story. We'll use lasso to help us with **model selection**: it will tell us which variables to include minimize the sum of the error term.

We'll use the command called **lasso2** that we downloaded above. We'll focus on model selection here, but I recommend you check out the package website for all that it can do: https://statalasso.github.io/docs/lassopack/. To see the documentation, type

```
. help lasso2
--Break--
r(1);
```

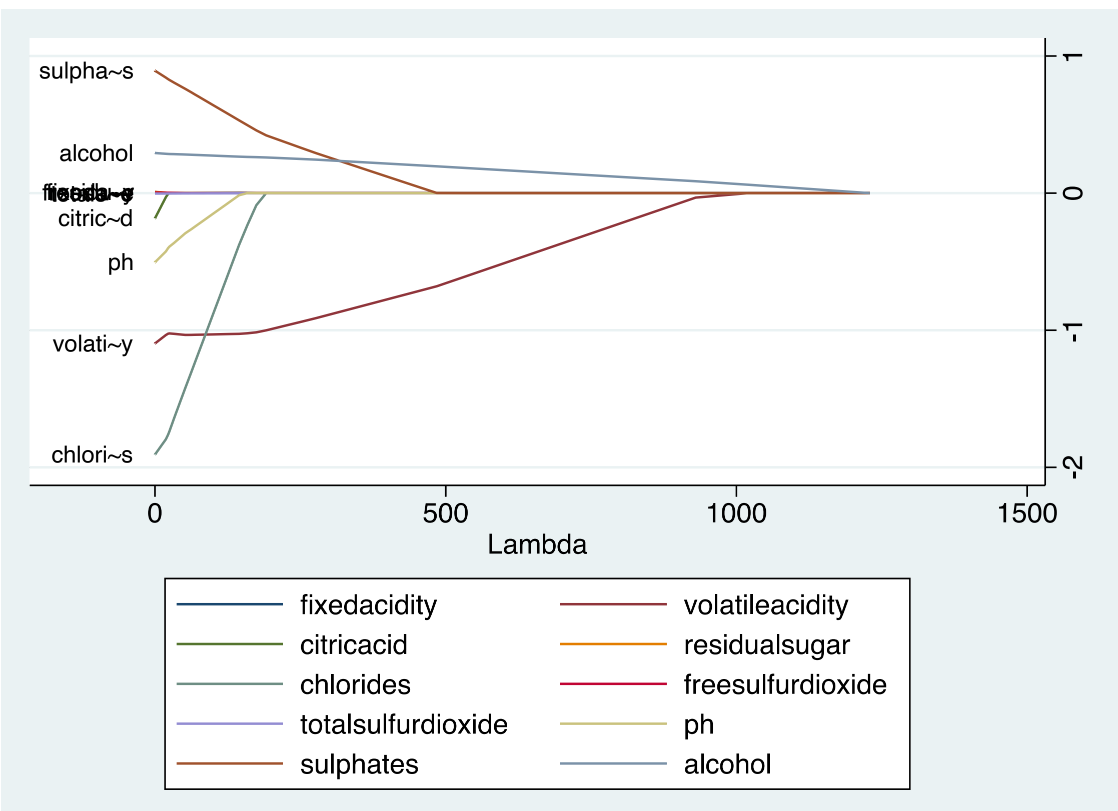
If you look at the help file, you'll see that we need to tell Stata in order to make the command run: 1) a dependent variable, 2) and a list of candidate explanatory variables from which the algorithm will select the "best fit" model. If we browse, we see that **quality** is the main outcome variable. Let's create a global called **winevars** to store all of the potential explanatory variables.

```
. global  winevars        fixedacidity volatileacidity ///
>                                         citricacid residualsugar        ///
>                                         chlorides freesulfurdioxide ///
>                                         totalsulfurdioxide ph sulphates alcoho
> l
```

Now let's run our lasso command. Remember that lasso is an algorithm that selects explanatory variables as a function of lamba, a coefficient "penalty". The one option worth noting with this command is: alpha(1). This tells Stata to run a **lasso** algorithm as opposed to ridge (see more explanation in the theory file). I'll include other options that you can see explained in the do file.

```
. lasso2 quality ${winevars} ///
>                                   , plotpath(lambda) plotlabel ///
>                                   plotvar(${winevars}) ///
>                                   plotopt(legend(on)) alpha(1)
```

```
  Knot|  ID     Lambda     s       L1-Norm        EBIC     R-sq   | Action
------+--------------------------------------------------------+---------------
     1|   1 1229.36594     1       0.00000   -677.11968   0.0000  | Added _cons.
     2|   2 1120.15246     2       0.03206   -732.51296   0.0385  | Added
      |                                                           | alcohol.
     3|   4  929.97025     3       0.11921   -832.91880   0.1012  | Added
      |                                                           | volatileacidit
> y.
     4|  11  484.88726     4       0.87514  -1132.93632   0.2584  | Added
      |                                                           | sulphates.
     5|  18  252.82062     5       1.51373  -1254.76666   0.3159  | Added
      |                                                           | totalsulfurdio
> xide.
     6|  22  174.25945     6       1.82470  -1283.78695   0.3313  | Added
      |                                                           | chlorides.
     7|  23  158.77872     7       2.01970  -1285.43610   0.3351  | Added
      |                                                           | fixedacidity.
     8|  24  144.67325     8       2.21970  -1286.25966   0.3385  | Added ph.
     9|  33   62.62560     7       3.61653  -1330.74797   0.3537  | Removed
      |                                                           | fixedacidity.
    10|  36   47.37397     8       3.87439  -1328.20385   0.3556  | Added
      |                                                           | freesulfurdiox
> ide.
    11|  41   29.75223     9       4.18108  -1326.68990   0.3580  | Added
      |                                                           | residualsugar.
    12|  44   22.50647    10       4.32028  -1321.18095   0.3587  | Added
      |                                                           | citricacid.
    13|  47   17.02532    11       4.46720  -1315.38612   0.3594  | Added
      |                                                           | fixedacidity.
Use long option for full output.
Type e.g. lasso2, lic(ebic) to run the model selected by EBIC.
```



If we take a look at the full table, we notice that as lambda decreases, more variables are added to our model.

(This is because the "pentalty" or adding additional variables decreases as lambda decreases.) We can see this trend by looking at the generated output graph. As lambda increases, the coefficents for our covariates decreases until they reach 0 – that is, when they are dropped from the model all together. But notice how some variables, such as **citricacid** "drop off" quicker than others, such as **volatileacidity**. One might interpret this as an indication that variables that are "slower" to drop off are better predictors of wine quality; this may be a general rule of thumb but is not always true. The iteration higlighted with a **"*"** will indicate the model with the lowest error (as measured by ebic). 7 variables are included in this model. Let's note the value of lambda that gave us our lowest "error": 51.99287. This is our "selected lambda"

Our next command will run standard OLS with the 7 variable selected by the lasso algorithm. It won't give us the full output we're used to with **reg**, but it will at least give us the value of the coefficients:

```
. lasso2, lic(ebic)

Use lambda=51.99287139699468 (selected by EBIC).

---------------------------------------------------
         Selected |           Lasso   Post-est OLS
------------------+--------------------------------
  volatileacidity |      -1.0340199     -1.0381945
        chlorides |      -1.4227042     -2.0022838
totalsulfurdiox~e |      -0.0019225     -0.0023721
               ph |      -0.2931852     -0.4351831
        sulphates |       0.7607917      0.8886801
          alcohol |       0.2817986      0.2906738
------------------+--------------------------------
   Partialled-out*|
------------------+--------------------------------
            _cons |       3.9284465      4.2957318
---------------------------------------------------
```

Cross-Validation

Cross-validation is a means of testing out-of-sample fit. We want to develop a model that is valid not only for our data in the data, but, in our case, for all wine. We'll do this by dividing the dataset into *k* folds, or a certain number of equal divisions. The standard number is 10, so since our dataset has about 1,600 observations, each *fold* will have about 160 wine samples. The algorithm then takes a large majority of the folds and uses this subset to develop its lasso model as we did above. Then it uses the remaning folds to "test" the model. The assumption is that, with large enough datasets, a randomly-divided dataset will generate enough variance between the "training" and "test" divisions that the "test" portion can approximate the differences of out-of-sample data.

A side-note on training and test data

This assumption I just explained about gets at what I see as a fundamental problem of applying machine-learning techniques to development problems: heterogeneity in data availability. Let's use an invented example. Let's say I want to predict likelihood of forest fires based on satellite images. I collect my data using the best publicly available images from NASA or NOAA that I can find, develop my theory of change, and maybe use lasso to fine-tune my model. This is great, but suppose NASA has much better quality images of North America than anywhere else. Even if I train my model on images from around the world, can I really say that my "test" data is good enough to create valid model for terrain that has different tree species, different roof coverings, and other differences that aren't captured in the lesser-quality images?

Back to Cross-validation

Lassopack also includes a cross-validation command called **cvlasso** that works very similarly to **lasso2**. We'll run this command, and also capture the selected varialbes in a global so we can run a better OLS command than last time. Keep in mind that if you type

```
. help cvlasso
```

you'll see that you can ajust the number of k-folds in the options.

```
. cvlasso         quality ${winevars} ///
>                         , plotcv seed(123) lopt alpha(1) postest

K-fold cross-validation with 10 folds. Elastic net with alpha=1.
Fold 1 2 3 4 5 6 7 8 9 10
          |           Lambda           MSPE       st. dev.
----------+---------------------------------------------
         1|       1229.3659      .65077866       .0224136
         2|       1120.1525      .62792033      .02181258
         3|       1020.6412       .6070669      .02094258
         4|       929.97025       .5874103       .0205783
         5|       847.35426      .56397203      .01931562
         6|       772.07764      .54408474       .0181137
         7|       703.48839      .52757376      .01708498
         8|       640.99242      .51386587       .0162064
         9|       584.04842      .50248518      .01545779
        10|       532.16317      .49301384      .01481433
        11|       484.88726      .48500471      .01423636
        12|       441.81121      .47697242       .0138183
        13|       402.56192      .46984836      .01352111
        14|       366.79943      .46393594        .013293
        15|       334.21398      .45902925      .01312089
        16|       304.52333      .45497734       .0129871
        17|       277.47032      .45169216      .01288373
        18|       252.82062      .44860531       .0127395
        19|       230.36073      .44575804      .01266945
        20|       209.89612      .44315179      .01267603
        21|       191.24953      .44115339       .0126832
        22|       174.25945      .43934693      .01269399
        23|       158.77872      .43719499      .01265673
        24|       144.67325      .43526388      .01262962  ^
        25|       131.82088      .43349194       .0126529
        26|       120.11028      .43184632      .01269913
        27|       109.44001      .43039701      .01271566
        28|       99.717662      .42919343       .0127343
        29|       90.859019      .42819391      .01275424
        30|       82.787355      .42736383      .01277485
        31|       75.432754      .42667445      .01279565
        32|       68.731516       .4261019      .01281627
        33|       62.625598      .42562931      .01283761
        34|       57.062112      .42529325      .01287306
        35|       51.992871       .4249987      .01290869
        36|       47.373968      .42468615      .01296141
        37|       43.165395      .42437546      .01300111
        38|       39.330701      .42406713      .01302018
        39|       35.836669      .42382454       .0130389
        40|       32.653039      .42365306       .0130622
```

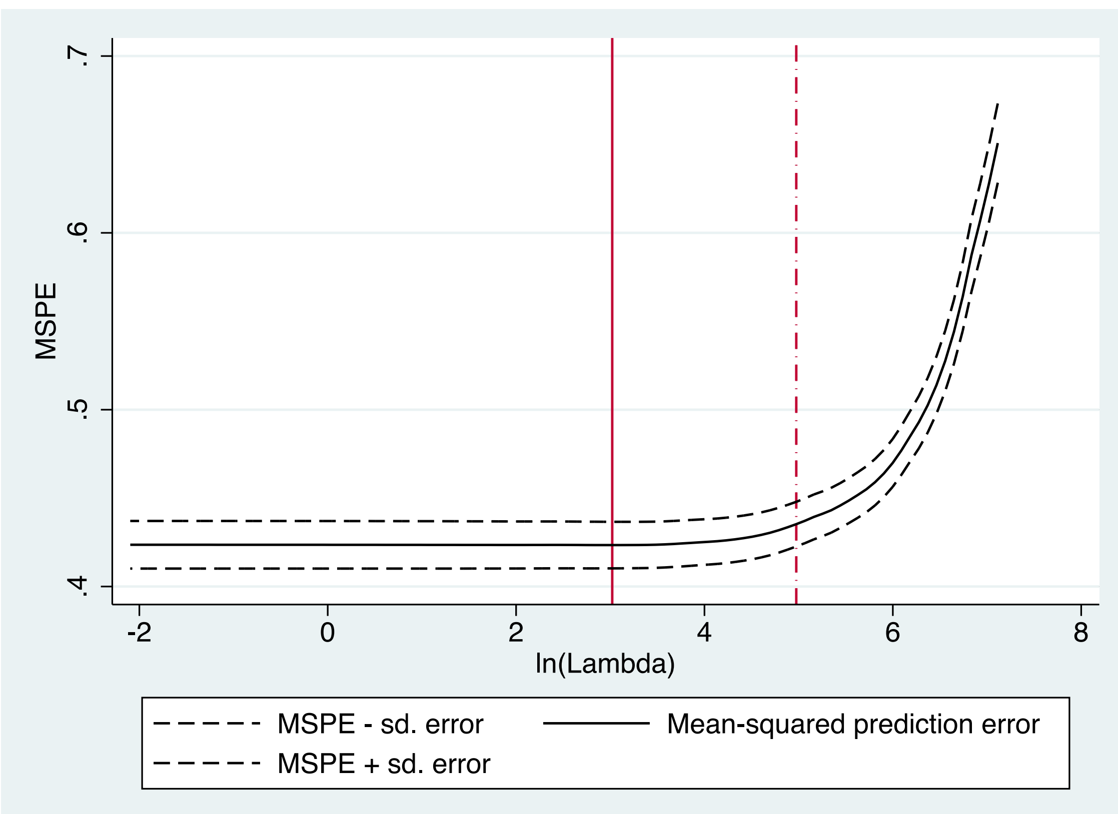
41	29.752233	.42355273	.0130849	
42	27.109126	.42349803	.01310832	
43	24.700826	.42346882	.01313051	
44	22.506473	.42345028	.01315046	
45	20.50706	.42344021	.01316775	*
46	18.685269	.42344336	.01318352	
47	17.025321	.42345655	.01319796	
48	15.512839	.42347742	.01321111	
49	14.134721	.42350378	.01322309	
50	12.879031	.42353391	.01323401	
51	11.734894	.42354737	.01324601	
52	10.692398	.42353933	.01326039	
53	9.7425148	.42353138	.0132751	
54	8.8770167	.42352097	.01328714	
55	8.088407	.42350734	.0132936	
56	7.3698552	.42350264	.01330283	
57	6.7151376	.42350727	.01331551	
58	6.1185833	.42351302	.01332708	
59	5.5750252	.42351955	.01333765	
60	5.0797553	.42352438	.01334827	
61	4.6284838	.42352916	.01335819	
62	4.2173021	.42353426	.01336724	
63	3.8426486	.42353952	.01337549	
64	3.5012783	.42354484	.013383	
65	3.1902344	.42355011	.01338986	
66	2.9068227	.42355527	.0133961	
67	2.6485886	.42356026	.0134018	
68	2.4132953	.42356506	.01340699	
69	2.1989048	.42356963	.01341172	
70	2.0035602	.42357397	.01341603	
71	1.8255694	.42357806	.01341996	
72	1.6633908	.4235819	.01342354	
73	1.5156198	.4235855	.01342681	
74	1.3809763	.42358886	.01342978	
75	1.2582942	.42359199	.01343249	
76	1.1465108	.42359489	.01343496	
77	1.0446579	.42359758	.01343722	
78	.9518534	.42360008	.01343927	
79	.86729337	.42360238	.01344114	
80	.79024541	.4236045	.01344284	
81	.72004219	.42360646	.01344439	
82	.65607563	.42360826	.01344581	
83	.59779168	.42360991	.0134471	
84	.54468551	.42361144	.01344827	
85	.49629715	.42361283	.01344934	
86	.45220748	.42361411	.01345032	
87	.41203461	.42361529	.01345121	
88	.37543059	.42361636	.01345202	
89	.34207838	.42361735	.01345276	
90	.31168908	.42361825	.01345343	
91	.28399948	.42361908	.01345404	
92	.25876975	.42361983	.0134546	
93	.23578136	.42362052	.01345511	
94	.21483519	.42362115	.01345557	
95	.19574982	.42362173	.01345599	
96	.17835995	.42362225	.01345638	
97	.16251494	.42362274	.01345673	
98	.14807756	.42362317	.01345705	
99	.13492276	.42362358	.01345734	
100	.12293659	.42362394	.01345761	

* lopt = the lambda that minimizes MSPE.

```
  Run model: cvlasso, lopt
^ lse = largest lambda for which MSPE is within one standard error of the
        minimal MSPE.
  Run model: cvlasso, lse
Estimate lasso with lambda=20.507 (lopt).

---------------------------------------------------
         Selected |           Lasso   Post-est OLS
------------------+--------------------------------
  volatileacidity |      -1.0303012     -1.0815017
       citricacid |      -0.0182698     -0.1426059
    residualsugar |       0.0030948      0.0093998
        chlorides |      -1.7797219     -1.9615892
freesulfurdioxide |       0.0030169      0.0045912
totalsulfurdiox~e |      -0.0028702     -0.0034134
               ph |      -0.4142290     -0.5465107
        sulphates |       0.8373421      0.8968997
          alcohol |       0.2863251      0.2916526
------------------+--------------------------------
   Partialled-out*|
------------------+--------------------------------
            _cons |       4.2541763      4.6583179
---------------------------------------------------

. global          lassovars = e(selected)
```



Notice how the graph now is different: we see the natural log of lambda (indicated by a red line) that generated the model with the lowest error. Notice that the corss-validation method gave us a different ideal lambda value. Since we stored the variables the cvlasso command selected, we can run OLS with these variables with

```
. reg             quality         ${lassovars}

      Source |       SS           df       MS      Number of obs   =     1,599
```

```
-------------+----------------------------------   F(9, 1589)      =     99.39
       Model |  375.359537         9  41.7066153   Prob > F        =    0.0000
    Residual |  666.805566     1,589  .419638493   R-squared       =    0.3602
-------------+----------------------------------   Adj R-squared   =    0.3565
       Total |   1042.1651     1,598    .6521684   Root MSE        =     .6478

------------------------------------------------------------------------------
     quality |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
volatileaci~y |  -1.081502   .1163884    -9.29   0.000    -1.309793   -.8532108
   citricacid |  -.1426059   .1229263    -1.16   0.246    -.3837208    .0985089
residualsugar |   .0093998   .0120188     0.78   0.434    -.0141746    .0329742
    chlorides |  -1.961589   .4030404    -4.87   0.000    -2.752136   -1.171042
freesulfurd~e |   .0045912   .0021574     2.13   0.033     .0003596    .0088228
totalsulfur~e |  -.0034134   .0006982    -4.89   0.000    -.0047829    -.002044
           ph |  -.5465107    .133194    -4.10   0.000    -.8077651   -.2852563
    sulphates |   .8968997   .1104474     8.12   0.000     .6802618    1.113538
      alcohol |   .2916526   .0172016    16.95   0.000     .2579123    .3253929
        _cons |   4.658318   .4610666    10.10   0.000     3.753955    5.562681
------------------------------------------------------------------------------
```

Now that we have significance values, we can look and see which chemical components are strong predictors of quality.

Ridge

Recall that ridge operates very similarly to lasso. The main differences is that, due to squared constraint term, no coveriates will every be *completely* dropped from the model; “unimportant” covariates will only see their coefficents reduced to near-zero.

Let’s use a new dataset to explore ridge. This one from Prof. Hastie for predicting rates of prostate cancer. (Btw, his website is here for more info and datasets: http://web.stanford.edu/~hastie/pub.htm)

```
. import delimited using "https://web.stanford.edu/~hastie/ElemStatLearn/dataset
> s/prostate.data", clear
```

We’ll follow the steps above, except we’ll tell stata to set *alpha=0* in the options, which indicates a ridge regression. **Lpsa** is our outcome variable of interest, and we’ll group the explanatory variables in a global like last time. Let’s run a lasso before ridge so we can compare.

```
. global          rhsvars lcavol lweight ///
>                                     age lbph svi lcp gleason pgg45

. lasso2          lpsa ${rhsvars} , plotpath(lambda) plotlabel ///
>                                plotvar(${rhsvars}) ///
>                                plotopt(legend(on)) alpha(1) ///
>                                lic(ebic) postresults long

  Knot|  ID     Lambda     s       L1-Norm        EBIC      R-sq   | Action
------+-----------------------------------------------------------+---------------
     1|   1  163.62492     1       0.00000     31.41226    0.0000  | Added _cons.
     2|   2  149.08894     2       0.06390     26.66962    0.0916  | Added lcavol.
      |   3  135.84429     2       0.12213     18.19047    0.1676  |
      |   4  123.77625     2       0.17518     10.54017    0.2307  |
      |   5  112.78031     2       0.22352      3.69568    0.2832  |
```

	6	102.76122	2	0.26757	-2.37828	0.3267	
	7	93.63220	2	0.30770	-7.72701	0.3628	
	8	85.31417	2	0.34427	-12.40327	0.3928	
3	9	77.73509	3	0.40800	-12.63533	0.4221	Added svi.
	10	70.82932	3	0.48389	-17.25324	0.4490	
4	11	64.53704	4	0.60174	-18.31145	0.4801	Added lweight.
	12	58.80375	4	0.71293	-23.37859	0.5066	
	13	53.57979	4	0.81423	-27.79633	0.5285	
	14	48.81991	4	0.90654	-31.62332	0.5468	
	15	44.48288	4	0.99065	-34.91944	0.5619	
	16	40.53114	4	1.06728	-37.74368	0.5745	
	17	36.93047	4	1.13711	-40.15255	0.5849	
	18	33.64967	4	1.20073	-42.19891	0.5936	
	19	30.66032	4	1.25870	-43.93126	0.6008	
	20	27.93654	4	1.31152	-45.39336*	0.6067	
5	21	25.45474	5	1.35340	-42.20238	0.6123	Added pgg45.
6	22	23.19341	6	1.39138	-38.93672	0.6175	Added lbph.
	23	21.13297	6	1.42643	-40.17769	0.6224	
	24	19.25558	6	1.45836	-41.22016	0.6264	
	25	17.54496	6	1.48746	-42.09423	0.6298	
	26	15.98632	6	1.51397	-42.82594	0.6325	
	27	14.56614	6	1.53812	-43.43763	0.6349	
	28	13.27212	6	1.56013	-43.94842	0.6368	
7	29	12.09306	7	1.58269	-39.94418	0.6389	Added age.
	30	11.01875	7	1.61022	-40.80902	0.6421	
	31	10.03987	7	1.63531	-41.53293	0.6448	
	32	9.14796	7	1.65817	-42.13807	0.6470	
	33	8.33528	7	1.67900	-42.64335	0.6488	
	34	7.59480	7	1.69798	-43.06485	0.6503	
8	35	6.92010	8	1.71689	-38.84649	0.6516	Added gleason.
	36	6.30533	8	1.73692	-39.15171	0.6527	
	37	5.74518	8	1.75517	-39.40584	0.6536	
	38	5.23480	8	1.77180	-39.61733	0.6544	
	39	4.76975	8	1.78695	-39.79327	0.6550	
	40	4.34602	8	1.80076	-39.93957	0.6555	
9	41	3.95993	9	1.83346	-35.69248	0.6567	Added lcp.
	42	3.60814	9	1.86831	-36.01479	0.6578	
	43	3.28761	9	1.90006	-36.28320	0.6588	
	44	2.99554	9	1.92900	-36.50659	0.6596	
	45	2.72943	9	1.95536	-36.69246	0.6602	
	46	2.48695	9	1.97938	-36.84703	0.6607	
	47	2.26602	9	2.00126	-36.97555	0.6612	
	48	2.06471	9	2.02120	-37.08238	0.6616	
	49	1.88129	9	2.03937	-37.17116	0.6619	
	50	1.71416	9	2.05593	-37.24493	0.6621	
	51	1.56188	9	2.07101	-37.30622	0.6623	
	52	1.42313	9	2.08476	-37.35713	0.6625	
	53	1.29670	9	2.09728	-37.39942	0.6627	
	54	1.18150	9	2.10869	-37.43454	0.6628	
	55	1.07654	9	2.11909	-37.46371	0.6629	
	56	0.98091	9	2.12856	-37.48793	0.6630	
	57	0.89376	9	2.13720	-37.50804	0.6630	
	58	0.81437	9	2.14506	-37.52475	0.6631	
	59	0.74202	9	2.15223	-37.53862	0.6632	
	60	0.67610	9	2.15876	-37.55013	0.6632	
	61	0.61604	9	2.16471	-37.55969	0.6632	
	62	0.56131	9	2.17013	-37.56763	0.6633	
	63	0.51145	9	2.17507	-37.57422	0.6633	
	64	0.46601	9	2.17957	-37.57970	0.6633	

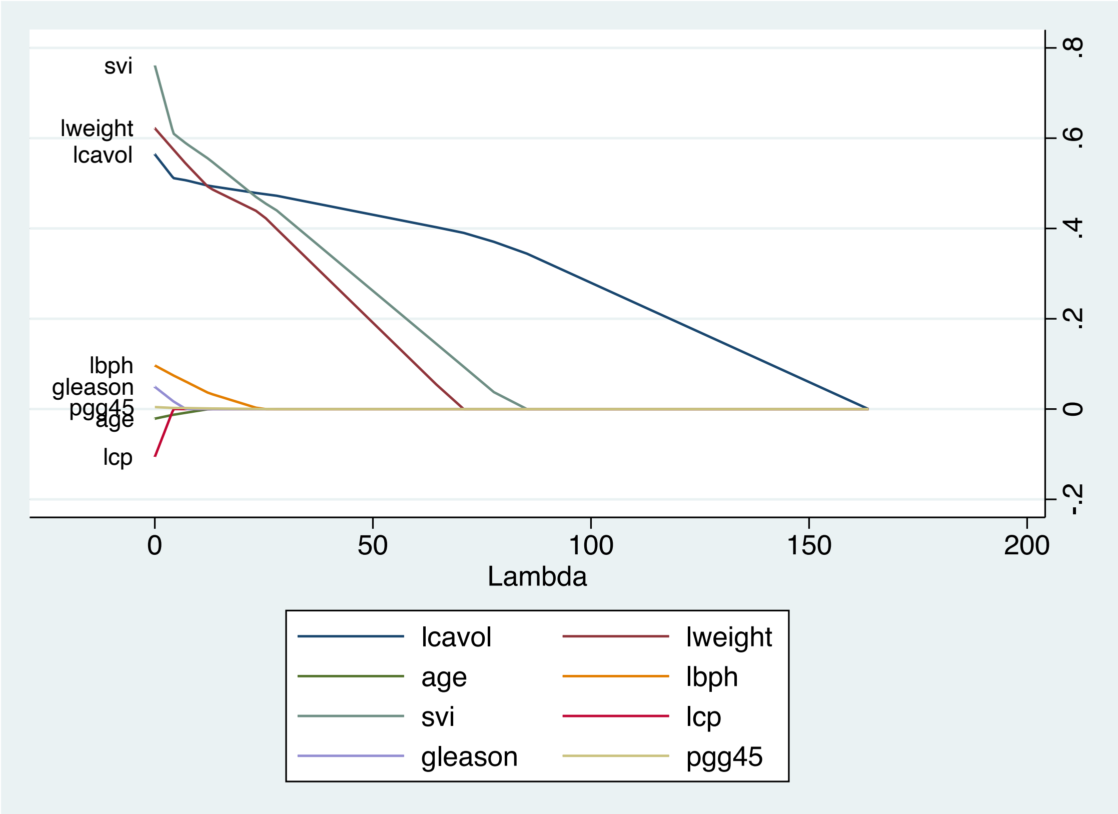
```
      |  65    0.42461     9     2.18367    -37.58424   0.6633  |
      |  66    0.38689     9     2.18741    -37.58801   0.6633  |
      |  67    0.35252     9     2.19081    -37.59115   0.6633  |
      |  68    0.32120     9     2.19391    -37.59375   0.6633  |
      |  69    0.29267     9     2.19674    -37.59590   0.6634  |
      |  70    0.26667     9     2.19932    -37.59770   0.6634  |
      |  71    0.24298     9     2.20166    -37.59919   0.6634  |
      |  72    0.22139     9     2.20380    -37.60042   0.6634  |
      |  73    0.20172     9     2.20575    -37.60145   0.6634  |
      |  74    0.18380     9     2.20752    -37.60230   0.6634  |
      |  75    0.16748     9     2.20914    -37.60301   0.6634  |
      |  76    0.15260     9     2.21062    -37.60359   0.6634  |
      |  77    0.13904     9     2.21196    -37.60408   0.6634  |
      |  78    0.12669     9     2.21318    -37.60448   0.6634  |
      |  79    0.11543     9     2.21430    -37.60482   0.6634  |
      |  80    0.10518     9     2.21531    -37.60510   0.6634  |
      |  81    0.09584     9     2.21624    -37.60533   0.6634  |
      |  82    0.08732     9     2.21708    -37.60552   0.6634  |
      |  83    0.07956     9     2.21785    -37.60568   0.6634  |
      |  84    0.07250     9     2.21855    -37.60581   0.6634  |
      |  85    0.06606     9     2.21919    -37.60592   0.6634  |
      |  86    0.06019     9     2.21977    -37.60602   0.6634  |
      |  87    0.05484     9     2.22030    -37.60609   0.6634  |
      |  88    0.04997     9     2.22078    -37.60615   0.6634  |
      |  89    0.04553     9     2.22122    -37.60621   0.6634  |
      |  90    0.04148     9     2.22162    -37.60625   0.6634  |
      |  91    0.03780     9     2.22199    -37.60629   0.6634  |
      |  92    0.03444     9     2.22232    -37.60632   0.6634  |
      |  93    0.03138     9     2.22262    -37.60634   0.6634  |
      |  94    0.02859     9     2.22290    -37.60636   0.6634  |
      |  95    0.02605     9     2.22315    -37.60638   0.6634  |
      |  96    0.02374     9     2.22338    -37.60639   0.6634  |
      |  97    0.02163     9     2.22359    -37.60640   0.6634  |
      |  98    0.01971     9     2.22378    -37.60641   0.6634  |
      |  99    0.01796     9     2.22395    -37.60642   0.6634  |
      | 100    0.01636     9     2.22411    -37.60643   0.6634  |
*indicates minimum EBIC.

Use lambda=27.93654463358689 (selected by EBIC).

---------------------------------------------------
         Selected |           Lasso   Post-est OLS
------------------+--------------------------------
           lcavol |       0.4725389      0.5258519
          lweight |       0.3989102      0.6617699
              svi |       0.4400748      0.6656665
------------------+--------------------------------
   Partialled-out*|
------------------+--------------------------------
            _cons |       0.2975585     -0.7771568
---------------------------------------------------
Plotting only supported for list of lambda values.
Plotting options ignored.

. global                          lpsalasso  = e(selected)

. graph export            "output/lpsa-lasso-graph.png", replace
(file output/lpsa-lasso-graph.png written in PNG format)
```



Note our selected lambda value, and that we have three covariates with that value. Now let's run the same selection of variables with ridge.

```
. lasso2         lpsa ${rhsvars} , plotpath(lambda) plotlabel ///
>                                plotvar(${rhsvars}) ///
>                                plotopt(legend(on)) alpha(0) ///
>                                lic(ebic) postresults long

  Knot|  ID     Lambda     s      L1-Norm        EBIC      R-sq    | Action
------+------------------------------------------------------------+---------------
     1|   1  1.636e+05     7      0.00550      31.10766    0.0036  | Added lcavol
      |                                                            | lweight lbph
      |                                                            | svi lcp
      |                                                            | gleason _cons.
      |   2  1.491e+05     7      0.00603      31.07807    0.0039  |
      |   3  1.358e+05     7      0.00661      31.04562    0.0043  |
      |   4  1.238e+05     7      0.00726      31.01004    0.0047  |
      |   5  1.128e+05     7      0.00796      30.97101    0.0052  |
      |   6  1.028e+05     7      0.00873      30.92823    0.0057  |
      |   7  9.363e+04     7      0.00958      30.88132    0.0062  |
      |   8  8.531e+04     7      0.01051      30.82989    0.0068  |
      |   9  7.774e+04     7      0.01152      30.77351    0.0075  |
      |  10  7.083e+04     7      0.01264      30.71173    0.0082  |
      |  11  6.454e+04     7      0.01386      30.64401    0.0090  |
      |  12  5.880e+04     7      0.01520      30.56982    0.0099  |
      |  13  5.358e+04     7      0.01666      30.48854    0.0108  |
     2|  14  4.882e+04     8      0.01837      30.37788    0.0121  | Added age.
      |  15  4.448e+04     8      0.02014      30.27832    0.0132  |
      |  16  4.053e+04     8      0.02207      30.16931    0.0145  |
      |  17  3.693e+04     8      0.02419      30.04999    0.0159  |
      |  18  3.365e+04     8      0.02651      29.91940    0.0174  |
     3|  19  3.066e+04     9      0.02916      29.56364    0.0212  | Added pgg45.
      |  20  2.794e+04     9      0.03194      29.38726    0.0232  |
      |  21  2.545e+04     9      0.03498      29.19446    0.0254  |
```

	22	2.319e+04	9	0.03831	28.98378	0.0277	
	23	2.113e+04	9	0.04195	28.75367	0.0303	
	24	1.926e+04	9	0.04592	28.50245	0.0332	
	25	1.754e+04	9	0.05025	28.22832	0.0362	
	26	1.599e+04	9	0.05498	27.92935	0.0396	
	27	1.457e+04	9	0.06014	27.60349	0.0432	
	28	1.327e+04	9	0.06575	27.24856	0.0471	
	29	1.209e+04	9	0.07187	26.86224	0.0514	
	30	1.102e+04	9	0.07853	26.44207	0.0560	
	31	1.004e+04	9	0.08577	25.98547	0.0610	
	32	9147.95888	9	0.09364	25.48976	0.0664	
	33	8335.27943	9	0.10217	24.95213	0.0723	
	34	7594.79618	9	0.11143	24.36965	0.0785	
	35	6920.09542	9	0.12146	23.73936	0.0853	
	36	6305.33321	9	0.13232	23.05820	0.0925	
	37	5745.18479	9	0.14405	22.32310	0.1003	
	38	5234.79841	9	0.15671	21.53101	0.1086	
	39	4769.75334	9	0.17035	20.67891	0.1174	
	40	4346.02160	9	0.18503	19.76390	0.1268	
	41	3959.93302	9	0.20080	18.78321	0.1368	
	42	3608.14349	9	0.21771	17.73433	0.1473	
	43	3287.60596	9	0.23581	16.61501	0.1585	
	44	2995.54411	9	0.25515	15.42341	0.1702	
	45	2729.42823	9	0.27576	14.15813	0.1825	
	46	2486.95335	9	0.29768	12.81831	0.1954	
	47	2266.01927	9	0.32095	11.40377	0.2087	
	48	2064.71236	9	0.34557	9.91503	0.2226	
	49	1881.28900	9	0.37157	8.35342	0.2369	
	50	1714.16047	9	0.39895	6.72119	0.2516	
	51	1561.87918	9	0.42769	5.02152	0.2667	
	52	1423.12614	9	0.45777	3.25861	0.2821	
	53	1296.69954	9	0.48918	1.43767	0.2977	
	54	1181.50432	9	0.52185	-0.43509	0.3135	
	55	1076.54274	9	0.55575	-2.35244	0.3294	
	56	980.90565	9	0.59079	-4.30628	0.3453	
	57	893.76469	9	0.62690	-6.28767	0.3611	
	58	814.36510	9	0.66398	-8.28701	0.3768	
	59	742.01915	9	0.70195	-10.29418	0.3923	
	60	676.10021	9	0.74068	-12.29875	0.4075	
	61	616.03733	9	0.78006	-14.29014	0.4223	
	62	561.31027	9	0.81997	-16.25788	0.4368	
	63	511.44502	9	0.86029	-18.19177	0.4509	
	64	466.00965	9	0.90088	-20.08212	0.4644	
	65	424.61064	9	0.94161	-21.91988	0.4775	
	66	386.88940	9	0.98236	-23.69681	0.4900	
	67	352.51921	9	1.02300	-25.40561	0.5020	
	68	321.20238	9	1.06340	-27.03996	0.5134	
	69	292.66764	9	1.10345	-28.59459	0.5242	
4	70	266.66786	8	1.14310	-30.06520	0.5345	Removed age.
5	71	242.97782	9	1.18294	-31.44891	0.5442	Added age.
	72	221.39234	9	1.22213	-32.74329	0.5533	
	73	201.72445	9	1.26063	-33.94720	0.5619	
	74	183.80381	9	1.29835	-35.06032	0.5700	
	75	167.47519	9	1.33521	-36.08310	0.5776	
	76	152.59715	9	1.37114	-37.01671	0.5847	
	77	139.04084	9	1.40608	-37.86296	0.5914	
	78	126.68884	9	1.43999	-38.62419	0.5976	
	79	115.43415	9	1.47280	-39.30319	0.6033	
	80	105.17930	9	1.50449	-39.90316	0.6087	
	81	95.83546	9	1.53503	-40.42759	0.6137	
	82	87.32170	9	1.56438	-40.88022	0.6183	

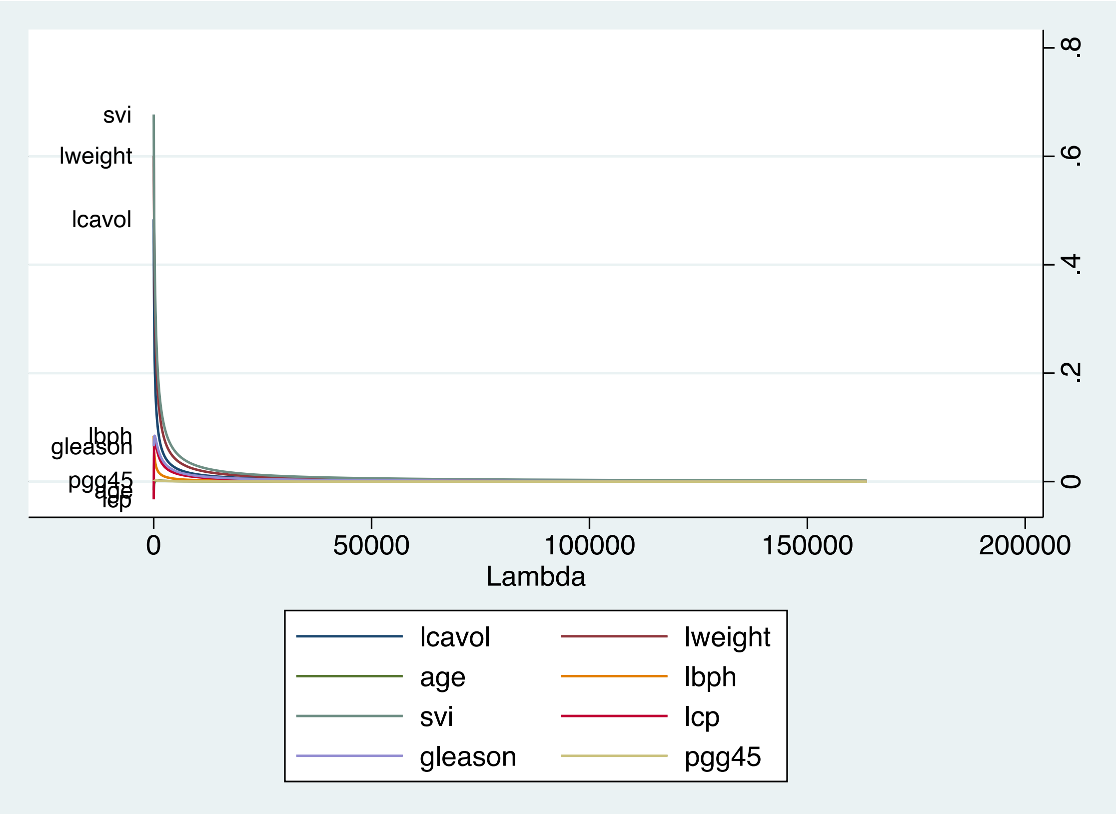
```
      |  83   79.56428     9     1.59254    -41.26496   0.6226  |
      |  84   72.49601     9     1.61950    -41.58586   0.6266  |
      |  85   66.05566     9     1.64525    -41.84705   0.6302  |
      |  86   60.18746     9     1.66979    -42.05269   0.6336  |
      |  87   54.84057     9     1.69314    -42.20694   0.6366  |
      |  88   49.96869     9     1.71530    -42.31395   0.6395  |
      |  89   45.52961     9     1.73628    -42.37780   0.6421  |
      |  90   41.48488     9     1.75613    -42.40252*  0.6444  |
      |  91   37.79948     9     1.77484    -42.39203   0.6465  |
      |  92   34.44148     9     1.79247    -42.35016   0.6485  |
      |  93   31.38179     9     1.80902    -42.28063   0.6502  |
      |  94   28.59392     9     1.82770    -42.18699   0.6518  |
      |  95   26.05372     9     1.85317    -42.07267   0.6532  |
      |  96   23.73917     9     1.87753    -41.94094   0.6545  |
      |  97   21.63025     9     1.90079    -41.79487   0.6556  |
      |  98   19.70868     9     1.92293    -41.63735   0.6566  |
      |  99   17.95782     9     1.94395    -41.47105   0.6575  |
      | 100   16.36249     9     1.96387    -41.29846   0.6583  |
*indicates minimum EBIC.

Use lambda=41.48488213793183 (selected by EBIC).

---------------------------------------------------
         Selected |           Ridge   Post-est OLS
------------------+--------------------------------
           lcavol |       0.4092614      0.5643413
          lweight |       0.5623717      0.6220198
              age |      -0.0113673     -0.0212482
             lbph |       0.0731408      0.0967125
              svi |       0.6032902      0.7616733
              lcp |       0.0204348     -0.1060509
          gleason |       0.0734774      0.0492279
            pgg45 |       0.0027821      0.0044575
------------------+--------------------------------
   Partialled-out*|
------------------+--------------------------------
            _cons |      -0.0872316      0.1815609
---------------------------------------------------
Plotting only supported for list of lambda values.
Plotting options ignored.

. global                          lpsaridge  = e(selected)

. graph export            "output/lpsa-ridge-graph.png", replace
(file output/lpsa-ridge-graph.png written in PNG format)
```



Notice how the ridge graph coefficent paths all approach zero as lambda approaches infinity, but never actually reach it. Likewise, if we compare the two following regressions with the lasso- and ridge-selected variables, we notice that the ridge regression includes all variables.

```
. eststo lasso: reg lpsa ${lpsalasso}

      Source |       SS           df       MS      Number of obs   =        97
-------------+----------------------------------   F(3, 93)        =     54.15
       Model |  81.3492223         3  27.1164074   Prob > F        =    0.0000
    Residual |  46.5684363        93  .500735874   R-squared       =    0.6359
-------------+----------------------------------   Adj R-squared   =    0.6242
       Total |  127.917659        96  1.33247561   Root MSE        =    .70763

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lcavol |   .5258519   .0748632     7.02   0.000     .3771884    .6745154
     lweight |   .6617699   .1756352     3.77   0.000     .3129933    1.010547
         svi |   .6656665   .2070898     3.21   0.002     .2544271    1.076906
       _cons |  -.7771568   .6229995    -1.25   0.215     -2.01431    .4599967
------------------------------------------------------------------------------

. eststo ridge: reg lpsa ${lpsaridge}

      Source |       SS           df       MS      Number of obs   =        97
-------------+----------------------------------   F(8, 88)        =     21.68
       Model |  84.8592398         8   10.607405   Prob > F        =    0.0000
    Residual |  43.0584188        88  .489300213   R-squared       =    0.6634
-------------+----------------------------------   Adj R-squared   =    0.6328
       Total |  127.917659        96  1.33247561   Root MSE        =     .6995

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
```

lcavol	.5643413	.0878335	6.43	0.000	.3897908	.7388918
lweight	.6220198	.2008967	3.10	0.003	.2227799	1.02126
age	-.0212482	.0110841	-1.92	0.058	-.0432755	.0007791
lbph	.0967125	.0579127	1.67	0.098	-.0183768	.2118018
svi	.7616733	.2411757	3.16	0.002	.2823873	1.240959
lcp	-.1060509	.089868	-1.18	0.241	-.2846446	.0725427
gleason	.0492279	.1553407	0.32	0.752	-.259479	.3579349
pgg45	.0044575	.0043653	1.02	0.310	-.0042177	.0131327
_cons	.1815609	1.320568	0.14	0.891	-2.442791	2.805913